Resampling vs Recombination: a Statistical Run Time Estimation

Tobias Friedrich
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Timo Kötzing
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Francesco Quinzan
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Andrew M. Sutton
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

ABSTRACT

Noise is pervasive in real-world optimization, but there is still little understanding of the interplay between the operators of randomized search heuristics and explicit noise-handling techniques, such as statistical resampling. In this paper, we report on several statistical models and theoretical results that help to clarify this reciprocal relationship for a collection of randomized search heuristics on noisy functions.

We consider the optimization of pseudo-Boolean functions under additive posterior Gaussian noise and explore the trade-off between noise reduction and the computational cost of resampling. We first perform experiments to find the optimal parameters at a given noise intensity for a mutation-only evolutionary algorithm, a genetic algorithm employing recombination, an estimation of distribution algorithm (EDA), and an ant colony optimization algorithm. We then observe how the optimal parameter depends on the noise intensity for the different algorithms. Finally, we locate the point where statistical resampling costs more than it is worth in terms of run time. We find that the EA requires the highest number of resamples to obtain the best speed-up, whereas crossover reduces both the run time and the number of resamples required. Most surprisingly, we find that EDA-like algorithms require no resampling, and can handle noise implicitly.

Keywords

Evolutionary Algorithm; Genetic Algorithm; Crossover; Estimation of Distribution Algorithm; Ant Colony Optimization; Robustness; Noise

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

FOGA '17, January 12 - 15, 2017, Copenhagen, Denmark

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-4651-1/17/01…$15.00

DOI: http://dx.doi.org/10.1145/3040718.3040723

1. INTRODUCTION

In many practical optimization problems, the objective function has some kind of stochastic component that arises out of different factors, such as measurement error, simulation nonlinearities, the finite precision of Monte Carlo sampling, or other environmental effects. In these scenarios, the direct evaluation of the objective function is not very reliable, and optimization algorithms must employ some kind of noise-handling strategy.

The most common type of noise-handling technique is statistical resampling. In this scenario, an algorithm estimates the true value of a function at a specific point by repeatedly sampling the corresponding value, to increase the signal to noise ratio. This approach comes at a computational cost, as the extra function evaluations must count toward the total run time of the algorithm. The idea of statistical resampling to address noise in the context of genetic algorithms has been studied as far back as 1988. In a paper by Fitzpatrick and Grefenstette [9], it was argued that the implicit parallelism of a GA is a sufficient mechanism for handling noise. They found that the amount of explicit resampling can be reduced in a GA by increasing the population size.

In the context of Evolution Strategies (ES) Arnold and Beyer [3] also found that increased population sizes are preferable to resampling as long as mutation strength is optimally adapted. Goldberg et al. [12] studied the influence of GA population size in the presence of noisy functions, but also with more general sources of noise (such as noisy operators). Arnold and Beyer [4] also noticed that the same algorithm may react in different ways to different types of posterior noise distributions.

The issue of resampling as a noise-handling technique has been approached from many different perspectives over the last few decades. Aizawa and Wah [1] proposed a detailed adaptive strategy for modelling the underlying noise, in order to determine the appropriate number of samples to be drawn from each individual. Stagge [23] recognized that noise can be reduced by repeated sampling, even at the cost of a higher number of function evaluations. However, he argues that computational effort can be saved by focusing only on resampling for the *best* individuals in the population. Many other detailed sampling frameworks have been

proposed, such as ones based on *selection races* from the machine learning community [15, 22].

Branke and Schmidt [5, 6] also recognised that resampling strategies produce a trade-off between noise reduction and computational effort. They consider a number of different sampling procedures that attempt to characterize the error probability during the selection step. They also raise the interesting point that sometimes noise can be *beneficial* to stochastic search algorithms (e.g., for promoting objective space exploration), and therefore attempting to eliminate noise completely may not always be the best strategy.

Akimoto et al. [2] explicitly study the run time effect of resampling on various noise models to derive the extra cost incurred by performing enough resampling to ensure the underlying optimization algorithm "sees" a noiseless function. For Gaussian noise, they show the existence of a resampling scheme such that any optimization algorithm that requires $r(\delta)$ function evaluations to optimize a noise-free function f with probability $1-\delta$ requires $\mathcal{O}(r(\delta)\max\{1, \sigma^2 \log(r\delta)/\delta\})$ evaluations to optimize $f+\mathcal{N}(0,\sigma^2)$ with probability $(1-\delta)^2$ under their resampling scheme.

Recently, Friedrich et al. [10] proved that an Estimation of Distribution Algorithm (EDA) called the Compact Genetic Algorithm (cGA) can *scale gracefully* with noise: its runtime on a noisy OneMax function is bounded by a polynomial in the variance, regardless of noise intensity. The cGA does not explicitly keep a population in memory, but only an array of so-called *allele frequencies* that represent the product distribution of alleles in an implicit population (cf. Goldberg et al. [14]). Offspring are then generated by drawing from this product distribution, and this has the same effect as *gene pool recombination*: all members of the population participate in recombination (rather than, e.g., two) to produce an offspring. On the other hand, they also proved that a mutation-only evolutionary algorithm exhibits superpolynomial runtime due to its reliance on hill-climbing a gradient that becomes obscured in the presence of heavy noise.

The array of allele frequencies employed by the cGA is similar to the pheromone values stored by Ant Colony Optimization (ACO) algorithms. In the latter, the update rule is distinct, but we should still expect similar protection against noise, which has also been recently noted [11].

None of these cases employ resampling, but instead rely on the implicit mechanisms of the GA/EDA/ACO to somehow "filter" the noise. Prügel-Bennett, Rowe and Shapiro [20] proved that a generational evolutionary algorithm using uniform crossover finds the optimum in $\mathcal{O}(\sigma^2 \log^2(\sigma^2))$ fitness evaluations, on OneMax with additive Gaussian noise of standard deviation σ. This result is in line with various studies on the effectiveness of population-based heuristics to deal with noise (Hellwig [17]). Dang and Lehre observe that for some cumulative distributions, the 2-tournament EA gives positive result on OneMax and LeadingOnes with additive posterior noise (cf. Dang [7]). Similarly, Hellwig and Beyer experimentally show that a population size controlled ES performs well in noisy environments. On the other hand, it has been argued that for some instances constant resampling operators are not useful in dealing with additive posterior gaussian noise (cf. Qian [21]).

In the context of continuous optimization, Hansen et. al. developed a heuristic to adapt the resampling based on the noise level of the rank (cf. Hansen [13]). They argue that if there are many rank changes when reevaluating a point, than the number of resampling should be geometrically increased. This suggests that in some cases non-constant resampling is beneficial. A similar idea has been discussed more formally in the form of Bernstein-Races (cf. Heidrich-Meisner [16]). However, the trade-off between resampling, recombination, and implicit methods is still not clear.

In this paper we want to take a first look at the interplay between resampling and implicit noise-handling exhibited by genetic algorithms employing crossover and estimation of distribution algorithms. We perform extensive statistical analysis on the run time of two genetic algorithms and two EDAs, to determine their asymptotic behavior.

2. SETTING AND ALGORITHMS

2.1 The test function

All tests are carried out by performing a global optimum search of the following function:

$$\text{OneMax}\colon (x_1, \ldots, x_n) \longmapsto \sum_{i=1}^{n} x_i$$

with $x_i \in \{0,1\}$ for all $i = 1, \ldots n$, and with objective space consisting of all pseudo-Boolean strings of fixed size n. We add posterior Gaussian noise in order to simulate an environment where errors of controlled standard deviation are produced. In other words, if we denote with σ the posterior noise standard deviation, the fitness function is

$$\text{OneMax} + \mathcal{N}(0, \sigma^2)$$

with $\mathcal{N}(0,\sigma^2)$ the centered Gaussian distribution. In some cases we simulate a resampling operator: we compute the fitness function r times, and take the average. Thus the test function in its generic form is:

$$\frac{1}{r}\sum_{j=1}^{r}\left(\text{OneMax} + \mathcal{N}(0,\sigma^2)\right)$$

$$= \text{OneMax} + \frac{1}{r}\sum_{j=1}^{r}\mathcal{N}(0,\sigma^2)$$

for a given $r > 0$. The OneMax function has the advantage of being symmetric, well-understood, and extensively studied. The goal of the hereby presented experiments is to statistically infer the asymptotic trend of some algorithms in view of a broader theoretical setting.

2.2 Algorithms

We consider four algorithms, namely $(\mu+1)$-EA, $(\mu+1)$-GA, cGA, and λ-MMASib. $(\mu+1)$-EA and $(\mu+1)$-GA are search heuristics inspired by the process of natural selection (cf. Algorithm 1 and Algorithm 2).

Typically, they require as input a *population* of strings of fixed length n. After an *offspring* is generated, a mutation factor is introduced, to ensure full objective space exploration. The fitness is then computed, and the less desirable result is discarded. The $(\mu+1)$-EA and the $(\mu+1)$-GA differ in how the offspring is generated. In the first case, the offspring is selected u.a.r. from the input population, while in the latter case a crossover operation is performed on two

Algorithm 1: $(\mu+1)$-EA

$t \leftarrow 0$;
Choose a population $P_0 \subseteq \{0,1\}^n$ s.t. $|P_0| = \mu$ u.a.r;
while *convergence criterion not met* **do**
  Select a parent $x \in P_t$ u.a.r;
  Generate offspring y by flipping each bit of x w/p $\frac{1}{n}$;
  Choose $z \in P_t \cup \{y\}$ s.t. $f(z) = \max_{x \in P_t} f(x)$;
  Define population $P_{t+1} \leftarrow (P_t \cup \{y\}) \setminus \{z\}$;
  $t \leftarrow t+1$;

Algorithm 2: $(\mu+1)$-GA

$t \leftarrow 0$;
Choose population $P_0 \subseteq \{0,1\}^n$ s.t. $|P_0| = \mu$ u.a.r;
while *convergence criterion not met* **do**
  Select parents $x_1, x_2 \in P_t$ u.a.r;
  Generate offspring y by recombining x_1 and x_2 u.a.r;
  Choose $z \in P_t \cup \{y\}$ s.t. $f(z) = \max_{x \in P_t} f(x)$;
  Define population $P_{t+1} \leftarrow (P_t \cup \{y\}) \setminus \{z\}$;
  $t \leftarrow t+1$;

u.a.r. chosen parents. We use uniform crossover, which consists of assembling a new element by choosing coefficients of one parent or the other with probability $p = 0.5$.

The cGA is a search heuristic similar to $(\mu+1)$-EA and $(\mu+1)$-GA. As shown in Algorithm 3, this process consists of sampling two *individuals* with given probability distribution, and swapping them according to the fitness evaluation. At each step, the distribution by which individuals are chosen is updated according to the fitness gain, and proportionally to a parameter K. The offspring generation procedure of the Compact Genetic Algorithm is equivalent to a concrete population with the same allele frequencies engaging in *gene pool recombination* introduced by Mühlenbein and Paaß [19]. In gene pool recombination, all members of the population participate as parents in uniform recombination. The correspondence between EDAs and models of sexually recombining populations has already been noted (cf. Mühlenbein and Paaß [18]).

The λ-MMASib is an ant colony optimization method (cf. Algorithm 4). Algorithms of this kind can be described in terms of λ ants exploring given *paths*, which correspond to pseudo-Boolean arrays of length n. The probability distribution by which ants choose one path over another is called *pheromone* vector, and it is updated at each step according to the fitness evaluation, and proportionally to a parameter ρ. Both cGA and λ-MMASib are an estimation of distribution algorithms (EDAs).

2.3 Statistical and experimental setting

For each set of experiments we look at the sample mean, sample standard deviation, and infer the trend toward asymptotic behaviour via model regression. All samples considered are of size $N \geq 10^2$; the exact size and relevant information is given in the description of each experiment. Statistical models with different properties are considered:

- polynomial models: $X \sim \alpha x^k + \beta$;
- rational models: $X \sim \frac{1}{\alpha x^k + \beta}$;

Algorithm 3: cGA

$t \leftarrow 0$;
$p_{1,t} \leftarrow p_{2,t} \leftarrow \cdots \leftarrow p_{n,t} \leftarrow 0.5$;
while *convergence criterion not met* **do**
  for $i = 1 \dots n$ **do**
    $x_i \leftarrow 1$ w/ prob. $p_{i,t}$, $x_i \leftarrow 0$ w/ prob. $1 - p_{i,t}$;
    $y_i \leftarrow 1$ w/ prob. $p_{i,t}$, $y_i \leftarrow 0$ w/ prob. $1 - p_{i,t}$;
  if $f(x_1, \dots, x_n) < f(y_1, \dots, y_n)$ **then**
    Swap $(x_1, \dots, x_n)$ with $(y_1, \dots, y_n)$;
  for $i = 1 \dots n$ **do**
    if $x_i > y_i$ **then**
      $p_{i,t+1} \leftarrow \min\left(\max\left(p_{i,t} + \frac{1}{K}, \frac{1}{n}\right), 1 - \frac{1}{n}\right)$;
    else if $x_i < y_i$ **then**
      $p_{i,t+1} \leftarrow \min\left(\max\left(p_{i,t} - \frac{1}{K}, \frac{1}{n}\right), 1 - \frac{1}{n}\right)$;
    else
      $p_{i,t+1} \leftarrow p_{i,t}$;
  $t \leftarrow t+1$;

Algorithm 4: λ-MMASib

$t \leftarrow 0$;
$p_{1,t} \leftarrow p_{2,t} \leftarrow \cdots \leftarrow p_{n,t} \leftarrow 0.5$;
while *convergence criterion not met* **do**
  for $i = 1 \dots \lambda$ **do**
    Generate x_i w/ prob. $p_t = (p_{1,t}, \dots, p_{n,t})$
  Choose $z \in \{x_1, \dots, x_\lambda\}$ s.t. $f(z) = \min_i\{f(x_i)\}$;
  for $i = 1 \dots n$ **do**
    if $z_i = 1$ **then**
      $p_{i,t+1} \leftarrow \min\left(p_{t,i}(1-\rho) + \rho, 1 - \frac{1}{n}\right)$;
    else
      $p_{i,t+1} \leftarrow \max\left(p_{t,i}(1-\rho), \frac{1}{n}\right)$;
  $t \leftarrow t+1$;

- square-root models: $X \sim \alpha\sqrt{x} + \beta$;
- square-root exponential models: $X \sim \alpha e^{\beta\sqrt{x}}$;
- any linear combination of the above;

In all cases, tests on the predictions made by the fitting models are performed. For a given experiment described by pairs $\{(x_i, y_i)\}_{i \in I}$, denote with $\bar{y}$ the sample mean, and let $\{f_i\}_{i \in I}$ be the predictions of a given model. Consider the quantities

$$SS_{res} := \sum_{i \in I} (y_i - f_i)^2$$

$$SS_{tot} := \sum_{i \in I} (f_i - \bar{y})^2$$

and consider the coefficient of determination

$$R^2 := 1 - \frac{SS_{res}}{SS_{tot}}$$

We assume that the model is valid if $R^2 > 0.95$. This choice is intuitively motivated by the fact that R^2 is the "percent of variance explained" by the model.

We perform a Student's t-Test on the each model, to determine whether it outperforms "random noise" as a predictor. We look at the corresponding p-value, and consider the

algorithm	parameter	description
cGA	K	distribution parameter
$(\mu+1)$-EA	μ	population size
$(\mu+1)$-GA	μ	population size
λ-MMASib	ρ	evaporation factor
λ-MMASib	λ	number of ants

Table 1: Parameters for the four algorithms.

model valid only for p-value < 0.05. We accept variables such that the probabilities $p_{|t|}$ of obtaining a corresponding value outside the confidence interval are $p_{|t|} < 10^{-5}$. Thus all variables have a very high level of significance.

All tests are carried out on MacBook Pro (13" Retina, Beginning 2015), with operating system Mac OS X Version 10.11.1, processor 2.7GHz dual-core Intel Core i5 (Turbo Boost up to 3.1GHz) with 3MB shared L3 cache, and memory 8GB of 1866MHz LPDDR3. All algorithms are implemented in C++ on Xcode Version 7.3.1 (7D1014), and implemented as OSX command line executables.The fitting was performed using least-square methods implemented by the 'lm' command of R 3.2.2 GUI 1.66 Mavericks build (6996). Code is available upon request.

Pseudo-random numbers are generated with the Mersenne Twister generator (64 bit) with a state size of 19937 bits, implemented with the 'std::mt19937_64' template instantation in the <random> library of C++11. The engine has an internal state sequence of n integer elements, which is filled with a pseudo-random series generated on construction or by calling member function seed. The internal state sequence becomes the source for n elements: when the state is advanced, the engine alters the state sequence by twisting the current value. The random numbers thusly generated have a period equivalent to the Mersenne number $2^{(n-1)*w} - 1$.

3. EXPERIMENTAL RESULTS

3.1 Parameter tuning

Each algorithm depends on one or two parameters, as shown in Table 1. The goal of parameter tuning is to reduce the expected number of fitness evaluations until the optimum search point is found. We refer to the optimal configuration by which the expected number of fitness evaluations is minimal as the *sweet spot*.

For every algorithm, parameter tuning is performed by brute force. We go through a list of possible input parameters and count the number of fitness calls that the algorithm needs with a given parameter; we make 10^2 independent runs and take the arithmetic mean. Note that since we count fitness evaluations and not wall clock time, this tuning is independent of the underlying machine. To perform parameter calibration for λ-MMASib, pairs of optimal choices for ρ, λ are tested, thus taking into account a possible correlation between the two. We give exemplary plots for the case of $\sigma^2 = 10$ in this section. The problem size is always fixed at $n = 10^2$.

Figure 1 shows the dependence of the optimization time of the $(\mu+1)$-EA and $(\mu+1)$-GA on the parameter μ. Note that both axes use a logarithmic scale. We see that optimization is slow for very small population sizes μ, quickly improves to best performance, and then slowly worsens again. This

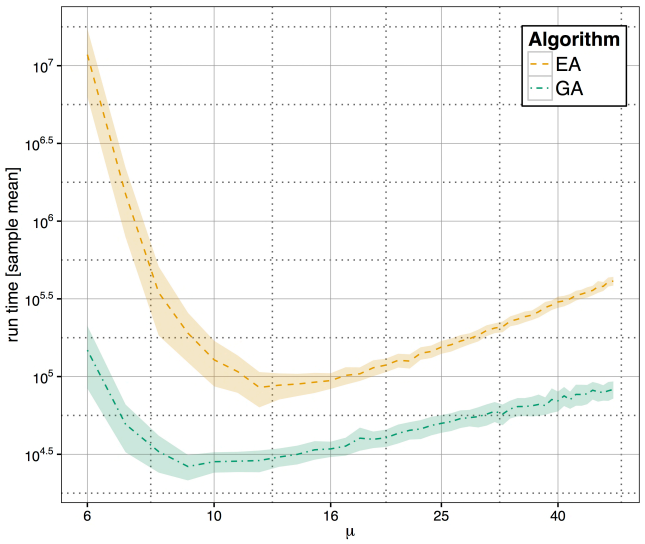

Figure 1: Number of fitness evaluations (run time) for a given parameter choice (μ) for the $(\mu+1)$-EA and $(\mu+1)$-GA. The problem size is fixed at $n = 10^2$.The run time is the average over 10^2 runs, and the shading is proportional to the sample standard deviation. Note that both axes follow a logarithmic scale. In both cases we see a unimodal trend. For the $(\mu+1)$-EA we see a sweet spot for $\mu = 12$, and for the $(\mu+1)$-GA the sweet spot is $\mu = 9$.

is in contrast to the well-known fact that, in the absence of noise, a choice of $\mu = 1$ is optimal for the $(\mu+1)$-EA.

In Figure 2 we can see a similar trend for the cGA. In this case, however, we observe a slightly different behaviour. In fact, higher K gives better worst-case and worse average-case. This observation has been already framed theoretically (cf. Droste [8]). For the cGA without boundaries on the distribution adjustment, there is a non-zero probability that the algorithm converges in infinite time ($p(+\infty) > 0$). In our case, we put boundaries on the distribution adjustment (cf. Algorithm 3), and we expect $p(+\infty) = 0$. However, during some runs the algorithm still may take a very long time to reach the optimum.

The case of λ-MMASib is more involved, since we have to optimize two parameters in parallel. Again we approach this problem with brute force and display here only an interesting selection of parameters ρ. As we can see in Figure 3, an evaporation factor slightly below 0.05 with a number of ants of around 5 is optimal.

In all cases there exists a sweet spot for the choice of parameters, at which the algorithm performs best. The difference between the sweet spot and the optimal parameter with absence of noise depends on the algorithm's stability to fitness evaluation errors.

We run statistical analysis on the parameter tuning experiments for the $(\mu+1)$-EA, and $(\mu+1)$-GA, as described in Section 2.3. In both cases we find that the parameter tuning trend is best described in term of rational function, as displayed below.

OBSERVATION 1. *Let $\mathcal{A}(\mu)$ be one of $(\mu+1)$-EA, $(\mu+1)$-GA, with μ the μ-parameter. Let $\tau_{\mu,\sigma}$ the corresponding random variable that returns the number of fitness evaluations*

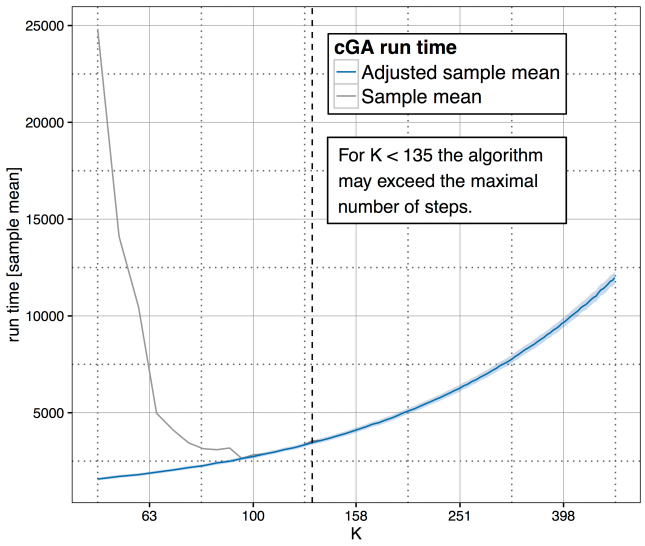

Figure 2: Number of fitness evaluations (run time) for the parameter K for the cGA. The problem size is fixed at $n = 10^2$. The run time is the average over 10^3 runs, and the shading is proportional to the sample standard deviation. We see a unimodal trend, and a sweet spot for $K = 135$. The adjusted sample mean does not consider the points at which the algorithm hits the maximal number of steps allowed, and returns a wrong result.

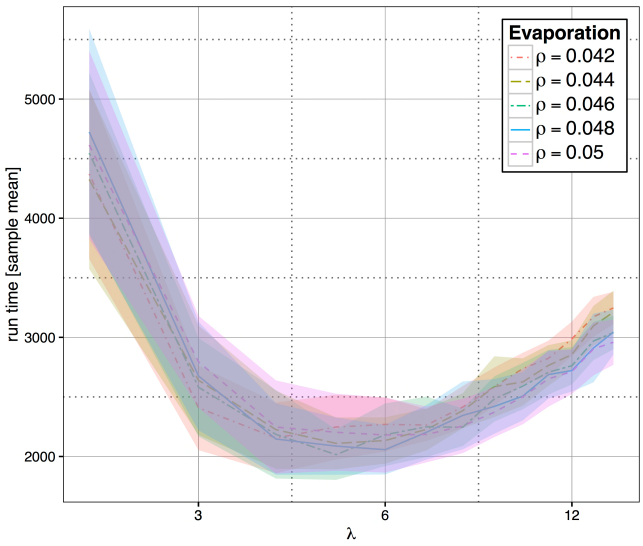

Figure 3: Number of fitness evaluations (run time) for a given pair of parameters (λ, ρ) for the λ-MMASib. The problem size is fixed at $n = 10^2$. Note that the λ-axis follows a logarithmic scale. The run time is the average over 10^2 runs, and the shading is proportional to the sample standard deviation. We see a unimodal trend, and a sweet spot for $\lambda = 5$ and $\rho = 0.046$.

until convergence, for a given posterior noise standard deviation σ. Consider the map:

$$f_{\mathcal{A}} : \mu \longmapsto \mathbb{E}[\tau_{\mu,\sigma}]$$

Then there exist $c_1, c_2, c_3 \in \mathbb{R}_{>0}$ s. t.

$$f_{\mathcal{A}} \sim c_1 - \frac{c_2}{\mu} + \frac{c_3}{\mu^2}$$

We perform nonlinear regression in the case of the cGA. We find that the most suitable model is

$$f_{\text{cGA}} \sim c_2 K^{\frac{2}{3}} + \frac{c_3}{K^7} - c_1$$

for $c_1, c_2, c_3 \in \mathbb{R}_{>0}$. This model seems to be more unnatural than the corresponding one for $(\mu + 1)$-EA or $(\mu + 1)$-GA. This is probably due to the fact that the outcome of the cGA parameter tuning depends on the user-defined maximal number of steps allowed. Thus this model, which is statistically valid, may differ slightly from the underlying true equation. In the case of λ-MMASib we find that the model is

$$f_{\lambda\text{-MMASib}} \sim c_1 + c_2\lambda + c_3\rho + \frac{c_4}{\lambda^2} + \frac{c_5}{\rho^2}$$

with λ the number of ants, ρ the evaporation factor, and $c_1, c_2, c_3, c_4, c_5 \in \mathbb{R}_{>0}$ constants. Again, the model is statistically valid, and all parameters are chosen to be very significant. However, the experimental data for λ-MMASib are a bit more sparse than in the other cases, probably because the parameters are tuned concurrently. Therefore, for the cGA and λ-MMASib sweet spot experiments, we claim a weaker result:

Observation 2. *Let $\mathcal{A}(x)$ be one of cGA, λ-MMASib, with x a parameter. Let $\tau_{x,\sigma}$ be the corresponding random variable that returns the number of fitness evaluation until convergence, and consider the function*

$$f_{\mathcal{A}} : x \longmapsto \mathbb{E}[\tau_{x,\sigma}]$$

Then there exists a parameter choice x_* such that $x_* = \min_{x<+\infty}\{f_{\mathcal{A}}(x)\}$. Moreover, $f_{\mathcal{A}}(y) \geq f_{\mathcal{A}}(y + \epsilon)$ for $y \leq x$ and $f_{\mathcal{A}}(y) \leq f_{\mathcal{A}}(y + \epsilon)$ for $y \geq x$, for all $\epsilon > 0$.

The next set of experiments focuses on finding the optimal parameters in dependence on the standard deviation σ of the noise. We exploit the unimodal shape of the dependence on the parameter, and perform a descent with a local search. The performance of this approach depends on finding a good starting value, preferably near the optimum. We increase σ slowly, and start the search where the optimum of the previous σ was found. Since a small change in the standard deviation results only in a small change of optimal parameter, this leads to faster search for the optimum.

The optimal parameters dependent on the noise are depicted in Figure 4 and 5. We can see a polynomial relation (roughly linear) for the $(\mu + 1)$-EA, $(\mu + 1)$-GA, and cGA. The λ-MMASib uses a decreasing evaporation rate while keeping the number of ants constant. In all cases we perform model regression:

Observation 3. *Let $\mathcal{A}(x)$ be one of the four tested algorithms, with x a parameter, and let $f_{\mathcal{A}}$ be as above. Consider the function*

$$min_{x<+\infty}\{f_{\mathcal{A}}(x)\} : \sigma \longmapsto x_*$$

that returns the optimal parameter for a given posterior noise standard deviation. Then

$$min_{x<+\infty}\{f_{\mathcal{A}}(x)\} \sim P(\sigma)$$

for a polynomial P.

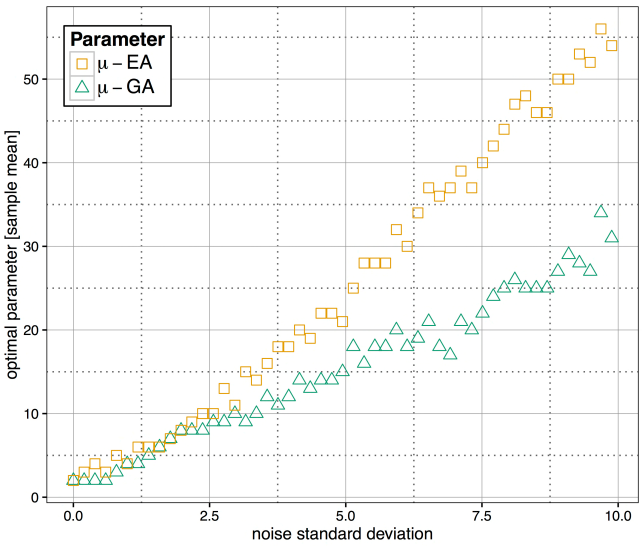

Figure 4: Optimal parameter choice for the $(\mu+1)$-EA and $(\mu+1)$-GA, for a given posterior noise standard deviation σ. The problem size is $n = 10^2$. In both cases, the optimal choice of the μ parameter increases polynomially with respect to the noise standard deviation. Each point is the sample mean of 10^2 runs. The run time for both algorithms with this parameters choice is presented in Figure 7.

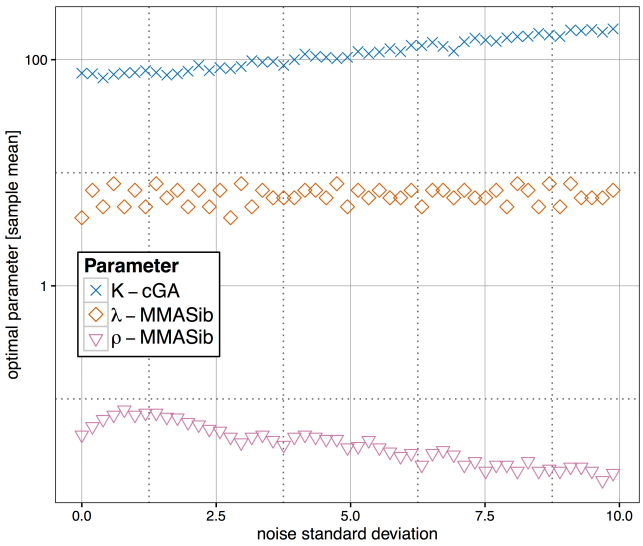

Figure 5: Optimal parameter choice for the cGA and λ-MMASib, for a given posterior noise standard deviation σ. The problem size is fixed at $n = 10^2$. Each point is the sample mean of 10^2 runs. Note that the optimal parameter axis is in logarithmic scale. The K and λ parameters linearly increase for increasing posterior noise standard deviation, while the ρ parameter linearly decreases. The run time for both algorithms with this parameters choice is presented in Figure 7.

3.2 Run time estimation

For each algorithm we approximate the run time function in dependence of the standard deviation and optimal parameter choice. This function returns the number of fitness evaluations to reach the optimum, with optimal parameters tuning, posterior error of fixed standard deviation, and no re-sampling. Let $\mathcal{A}$ be one of cGA, $(\mu+1)$-EA, $(\mu+1)$-GA, λ-MMASib. For a given algorithm and given standard deviation σ, we let

$$\sigma \longmapsto \mathbb{E}[\tau_\sigma]$$

be the expected run time of $\mathcal{A}$ with posterior noise standard deviation σ and optimal parameters choice as in Section 3.1. This set of experiments aims at giving an estimate of $\mathbb{E}[\tau_\sigma]$. The problem size is always fixed at $n = 10^2$, unless otherwise specified. Every observation is the arithmetic mean of 10^2 runs (see Figure 7).

The run time plot in Figure 7 shows that the $(\mu+1)$-EA quickly becomes inefficient, when increasing posterior noise standard deviation. In the case of the $(\mu+1)$-GA, results seem to be slightly better. However, we still can see that this algorithm significantly worsens with increasing posterior noise standard deviation. In the case of the λ-MMASib we see further improvement. The results in Figure 7 show that it reacts much better to increasing posterior noise standard deviation. Still, we observe a polynomial trend, with degree clearly greater than one (cf. Figure 7). We see a similar trend for the cGA.

As in the case of the parameter tuning experiments, we perform statistical regression:

Observation 4. *Let τ_σ be the runtime of $(\mu+1)$-EA or $(\mu+1)$-GA, for a given posterior noise standard deviation. Then*

$$\mathbb{E}[\tau_\sigma] \sim c_1 e^{c_2\sqrt{\sigma}}$$

for $c_1, c_2 \in \mathbb{R}_{>0}$ constants.

These results confirm the generic fact that, without employing any additional operator, the $(\mu+1)$-EA and $(\mu+1)$-GA perform poorly with noise. According to our statistical models, however, the $(\mu+1)$-GA has coefficient c_2 lower than the $(\mu+1)$-EA, while c_1 is the same for both algorithms. Thus the difference of the runtime in expected value is again exponential. Therefore, it seems that the recombination operator employed by the $(\mu+1)$-GA helps in dealing with posterior noise.

The cGA and λ-MMASib are much more stable under an increase in the posterior noise standard deviation. This is clear when looking at the data (cf. Figure 7), and it can be formalised through statistical analysis:

Observation 5. *Let τ_σ be the run time for cGA or λ-MMASib, for a given posterior noise standard deviation σ. Then*

$$\mathbb{E}[\tau_\sigma] \sim c_1\sigma^2 + c_2$$

for $c_1, c_2 \in \mathbb{R}_{>0}$ constants.

Recently, attempts have been made to perform a theoretical analysis of the run time of the hereby presented algorithms, solving OneMax with additive posterior gaussian noise. Particularly useful in this direction was the definition of graceful scaling (cf. Friedrich et al. [10]). Let F be a family of pseudo-Boolean functions $(F_n)_{n\in\mathbb{N}}$, with F_n a set of functions $f : \{0,1\}^n \longrightarrow \mathbb{R}$. Let D be a

family of distributions $(D_v)_{v\in\mathbb{R}}$ such that for all $D_v \in D$, $\mathbb{E}[D_v] = 0$. We define F with additive posterior D-noise as the set $F[D] := \{f_n + D_v : f_n \in F_n, D_v \in D\}$.

Definition 1. *An algorithm $\mathcal{A}$ scales gracefully with noise on $F[D]$ if there is a polynomial q such that, for all $g_{n,v} = f_n + D_v \in F[D]$, there exists a parameter setting x such that $\mathcal{A}(x)$ finds the optimum of f_n using at most $q(n, v)$ calls to $g_{n,v}$.*

Even though this definition does not give detailed information regarding the run time, we can safely assume that an algorithm that does not scale gracefully without noise shows experimental run time trend significantly worse than one that scales gracefully with noise. From this point of view, it seems that the $(\mu + 1)$-EA and $(\mu + 1)$-GA do not scale gracefully with noise, while both cGA and λ-MMASib do. In fact, partial theoretical results in this direction have been proven.

Theorem 1 (Theorem 11 in Friedrich et al. [10]). *Consider the cGA optimizing* OneMax *with additive posterior Gaussian noise of variance σ^2, and problem size n. If $K = \omega(\sigma^2\sqrt{n}\log n)$, then the cGA finds the optimum after $\mathcal{O}(K\sigma^2\sqrt{n}logKn)$ steps, with probability $1 - o(1)$.*

This result is consistent with the model given in Observation 5. A similar, negative result holds for $(\mu + 1)$-EA. In this case, if we look at the μ parameter, it can be proven that it does not scale polynomially with respect to the problem size. Since our unit of measurement is the number of fitness evaluations, this implies that the $(\mu + 1)$-EA does not scale gracefully with noise.

Theorem 2 (Theorem 5 in Friedrich et al. [10]). *Let $\mu \geq 1$ and let D be a distribution on R. Let Y be the random variable describing the minimum over μ independent copies of D. Suppose*

$$Pr(Y > D + n) \geq \frac{1}{2(\mu + 1)}$$

Consider optimisation of OneMax with reevaluated additive posterior noise from D by $(\mu + 1)$-EA. Then, for μ bounded from above by a polynomial, the optimum will not be evaluated after polynomially many iterations w.h.p.

Again, this result validates the exponential trend given in Observation 4. Weather the $(\mu + 1)$-GA or the λ-MMASib scale gracefully with noise is still an open question. Based on Theorem 1 and Theorem 2, it has been argued that recombination helps dealing with additive posterior Gaussian noise. This generic idea is confirmed by our observations: the $(\mu + 1)$-GA, which employs recombination, performs better then the $(\mu + 1)$-EA. The fact, however, that they both perform significantly worse than the EDA counterparts, leads us to claim that there are other structural elements which come into play when dealing with posterior noise.

We conclude with a description of the run time trend for the cGA and λ-MMASib, for increasing problem size and posterior noise standard deviation fixed at $\sigma = \sqrt{10}$. The results are displayed in Figure 6. The experiments show that both algorithms have very similar run time. The cGA seems to perform slightly better then the λ-MMASib, probably due to the fact that it uses an implicit recombination operator,

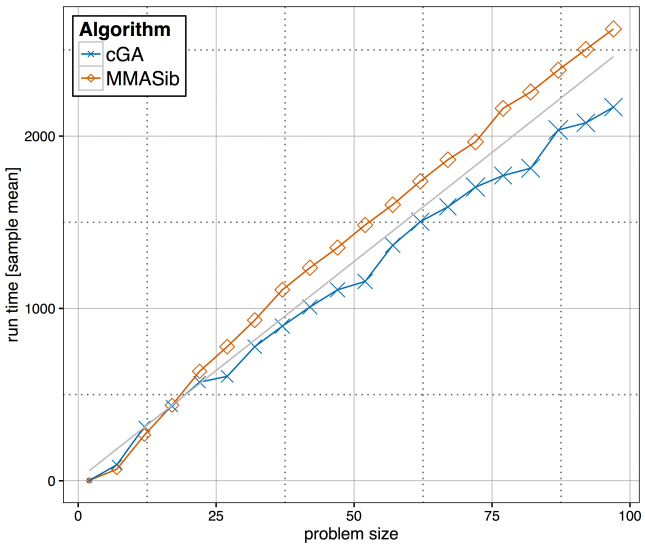

Figure 6: Number of fitness evaluations (run time) for a given problem size, taken to be the arithmetic mean of 10^2 runs, with optimally tuned parameters. Posterior noise standard deviation is fixed at $\sigma = \sqrt{10}$. The size of each point is proportional to the standard deviation. We see that the cGA performs slightly better then the λ-MMASib, even though they give very similar results. A linear approximation of the midpoint between the respective run time functions is highlighted in gray.

that may positively affect its performance in the presence of posterior gaussian noise. As in the other cases, we perform statistical analysis on the hereby presented data:

Observation 6. *The run time of the cGA, solving* OneMax $+\mathcal{N}(0, \sigma^2)$*, for a fixed posterior noise standard deviation σ is $\tau_n = \mathcal{O}(n\log(n))$. Similarly, the run time of the λ-MMASib is $\tau_n = \mathcal{O}(n^2\log(n))$. Moreover, the difference between the respective run times is at least linear for increasing n.*

Note that the run time of the cGA is consistent with the results presented in Theorem 1.

3.3 Resampling

We show that there exists a sweet spot for the resampling operator, as in the case of the parameter tuning experiments, for fixed problem size $n = 10^2$. This time, we do not need to compute directly the optimal setting for the resampling operator: there exists a clear relationship between the run time with and without resampling, which can be effectively described theoretically. In fact, the following equations hold:

$$\mathrm{Var}\left(\frac{1}{r}\sum_{k=1}^{r}\mathcal{N}(0,\sigma^2)\right) = \frac{1}{r^2}\mathrm{Var}\left(\mathcal{N}(0, r\sigma^2)\right) = \frac{\sigma^2}{r}$$

Let $\tau_{\sigma,r}$ be the random variable that returns the run time of an algorithm $\mathcal{A}$, for a given posterior noise standard deviation and number of resampling r. Due to the property of the noise variance mentioned above, we can approximate

$$\mathbb{E}[\tau_{\sigma,r}] = r\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r}}}\right] \quad (1)$$

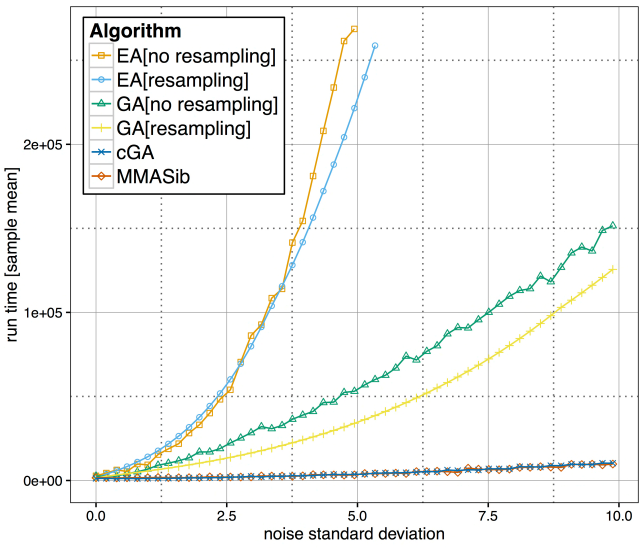

Figure 7: Number of fitness evaluations (run time) for the$(\mu+1)$-EA and $(\mu+1)$-GA with optimal resampling, and the cGA, $(\mu+1)$-EA, $(\mu+1)$-GA and λ-MMASib with no resampling, for a given posterior noise standard deviation. The problem size is fixed at $n = 10^2$. The $(\mu+1)$-EA and $(\mu+1)$-GA with optimal number of samples are derived from Equation 2, while in the other cases we performed direct experiments, with optimal parameter choice as in Figure 4 and Figure 5, and run time given by the arithmetic mean of a sample of 10^2 runs.

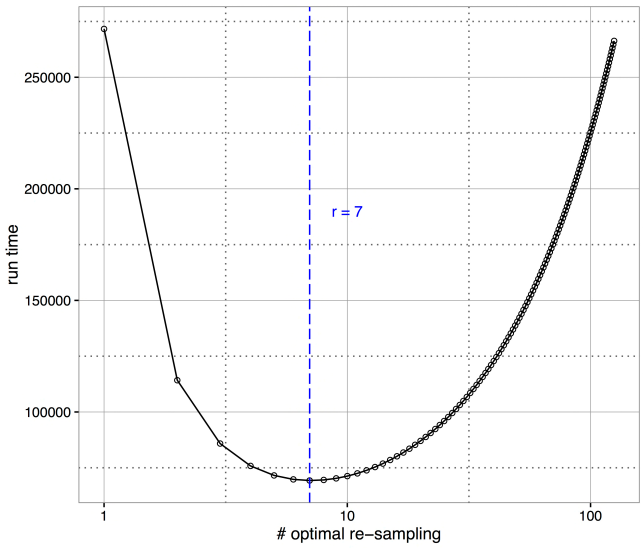

Figure 8: Number of fitness evaluations (run time) for a given number of samples. The problem size is fixed at $n = 10^2$. Posterior noise standard deviation is fixed at $\sigma = 5.0$, and optimal parameter choice is implicitly given in Equation 2. Note that the axis for the optimal number of samples is in logarithmic scale. We observe a unimodal trend, with sweet spot at $r = 7$. In Figure 9 we display the trend of the optimal r for a given posterior noise standard deviation.

from which we obtain the following result:

$$r_\sigma = \min_{1\leq r<+\infty} \left\{ r\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r}}}\right]\right\} \tag{2}$$

with r_σ the optimal amount of resampling for a given posterior noise standard deviation σ.

Using Equation 2, we can readily visualise the behaviour and trend of the optimal r_σ. We only consider the case of the $(\mu+1)$-EA: due to the fact that this algorithm performs poorly and the trend of r_σ is clearer than in other cases, even for low σ. Much like all of the other parameters, we see a unimodal trend and a sweet spot $r_\sigma < +\infty$. In Figure 9 we see the trend of the optimal number of samples for a given posterior noise standard deviation, and we can observe a linear increase in σ.

In Figure 7 we compare the run time trend of the $(\mu+1)$-EA, with the run time trend of $(\mu+1)$-GA, both given when the resampling operator is used and when it is not. The $(\mu+1)$-EA and $(\mu+1)$-GA with resampling are given by means of the regression model described in Observation 4. For both algorithms we observe improvement. However, for posterior noise standard deviation $\sigma \leq 10$, it seems that the $(\mu+1)$-EA with resampling still performs worse than the $(\mu+1)$-GA without resampling.

Additional experiments show that the resampling sweet-spot for the cGA and λ-MMASib is $r_\sigma = 1$. In Figure 7, we compare the $(\mu+1)$-EA and $(\mu+1)$-GA with resampling, with the cGA, and λ-MMASib without resampling. We see that the cGA and λ-MMASib perform better then the $(\mu+1)$-EA and $(\mu+1)$-GA with resampling. This confirms the fact that the benefit of resampling is limited, and noticeable only with algorithms that perform particularly bad under noisy environments. We can frame these ideas rigorously:

LEMMA 1. *Let $\tau_{\sigma,r}$ be the run time of an algorithm $\mathcal{A}$, for a given posterior noise standard deviation σ, and number of samples r. Define $\tau_\sigma = \tau_{\sigma,1}$ and let r_σ be the optimal number of samples. Suppose that*

$$\mathbb{E}[\tau_\sigma] \sim c_1 e^{c_2\sqrt{\sigma}}$$

for $c_1, c_2 \in \mathbb{R}_{>0}$. Then for σ sufficiently large, we have that

$$\mathbb{E}[\tau_{\sigma,r_\sigma}] \sim c_3 \left(\frac{16\sigma}{c_2^2}\right)^2$$

for a constant $c_3 \in \mathbb{R}$.

PROOF. We know from Equation 2 that

$$r_\sigma = \min_{1\leq r<+\infty} \left\{ r\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r}}}\right]\right\}$$

Thus we study the minima of $r\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r}}}\right]$. To do so, we consider the natural extension of this function over the non-negative real numbers, which is continuous and differentiable at each point. The derivative in r is

$$\frac{d}{dr}\left(r\mathbb{E}[\tau_{\frac{\sigma}{\sqrt{r}}}]\right) = \frac{e^{c_2\sqrt{\frac{\sigma}{\sqrt{r}}}+c_1}\left(4\sqrt{r}\sqrt{\frac{\sigma}{\sqrt{r}}} - c_2\sigma\right)}{4\sqrt{r}\sqrt{\frac{\sigma}{\sqrt{r}}}}$$

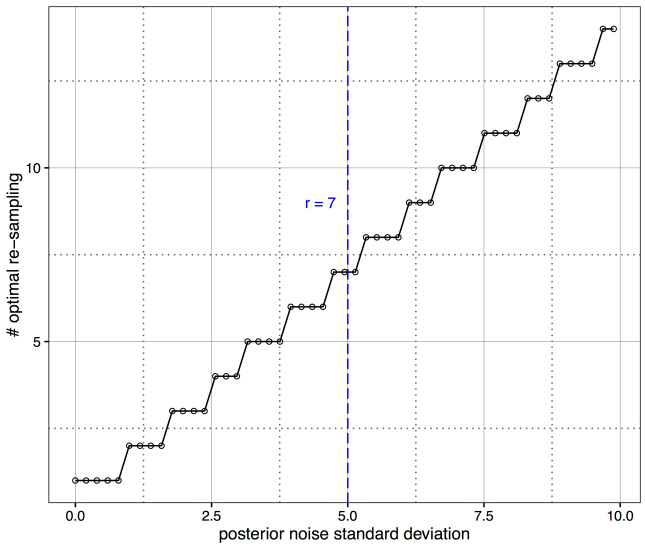

Figure 9: Optimal number of samples for a given posterior noise standard deviation. The optimal parameter choice is given implicitly in Equation 2. We observe a linear increase. For noise standard deviation $\sigma = 5.0$, we obtain the sweet spot displayed in Figure 8 ($r = 7$).

Therefore, all local optima of $r\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r}}}\right]$ in $(0, +\infty) \subseteq \mathbb{R}_{\geq 0}$ satisfy the equation

$$\frac{e^{c_2\sqrt{\frac{\sigma}{\sqrt{r}}}+c_1}\left(4\sqrt{r}\sqrt{\frac{\sigma}{\sqrt{r}}} - c_2\sigma\right)}{4\sqrt{r}\sqrt{\frac{\sigma}{\sqrt{r}}}} = 0$$

Standard calculations give us only one non-negative solution, namely $r_* = \frac{c_2^4\sigma^2}{4^4}$. Moreover,

$$\frac{d}{dr}\left(r\mathbb{E}[\tau_{\frac{\sigma}{\sqrt{r}}}]\right) < 0 \text{ for } 0 < r < r_*$$

$$\frac{d}{dr}\left(r\mathbb{E}[\tau_{\frac{\sigma}{\sqrt{r}}}]\right) > 0 \text{ for } r > r_*$$

Therefore, r_* is the absolute minimum of $r\mathbb{E}[\tau_{\frac{\sigma}{\sqrt{r}}}]$ for $r \in (0, +\infty)$. The sweet-spot for the resampling operator is given by the integer that best approximates this value, i.e.

$$r_\sigma \sim r_* = \frac{c_2^4\sigma^2}{4^4}$$

It follows that

$$\frac{\sigma}{\sqrt{r_\sigma}} \sim \left(\frac{4}{c_2}\right)^2$$

from which it follows that $\mathbb{E}\left[\tau_{\sigma,r_\sigma}\right] = c_3$, for $c_3 \in \mathbb{R}$ constant. Therefore, we obtain that

$$\mathbb{E}\left[\tau_{\sigma,r_\sigma}\right] = r_\sigma\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r_\sigma}}}\right] \sim c_3\left(\frac{16\sigma}{c_2^2}\right)^2$$

where we used the approximation given in Equation 1. □

From Lemma 1 it follows that, if we consider valid the statistical models given in Observation 4 and Observation 5,

algorithm	run time no resampling	resampling sweet spot	run time w/ resampling
$(\mu+1)$-EA	$\mathcal{O}(e^{c\sqrt{\sigma}})$	$\mathcal{O}(\sigma^2)$	$\mathcal{O}(\sigma^2)$
$(\mu+1)$-GA	$\mathcal{O}(e^{c\sqrt{\sigma}})$	$\mathcal{O}(\sigma^2)$	$\mathcal{O}(\sigma^2)$
cGA	$\mathcal{O}(\sigma^2)$	$\mathcal{O}(1)$	$\mathcal{O}(\sigma^2)$
λ-MMASib	$\mathcal{O}(\sigma^2)$	$\mathcal{O}(1)$	$\mathcal{O}(\sigma^2)$

Table 2: Run time with and without resampling for the four algorithms, with respect to posterior noise standard deviation σ.

then the $(\mu+1)$-EA and $(\mu+1)$-GA with resampling have a run time $\mathcal{O}(\sigma^2)$. Therefore, for very large σ, $(\mu+1)$-EA and $(\mu+1)$-GA with resampling perform at least as well as cGA and λ-MMASib without resampling. It is, thus, interesting to understand weather with the latter algorithms we can further improve performance. In this sense, the following lemma holds:

LEMMA 2. *Let τ_σ, $\tau_{\sigma,r}$, and r_σ be as above. Suppose that*

$$\mathbb{E}[\tau_\sigma] \sim c_1\sigma^2 + c_2$$

for $c_1, c_2 \in \mathbb{R}_{>0}$ constants. Then the optimal configuration for the resampling operator is always $r_\sigma = 1$.

PROOF. The proof is very similar to the one given in Lemma 1: we study the minima of the function

$$r\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r}}}\right]$$

We observe, however, that this function is linear in r, and

$$\frac{d}{dr}\left(r\left(c_1\left(\frac{\sigma}{\sqrt{r}}\right)^2 + c_2\right)\right) = c_2$$

for a constant $c_2 > 0$. Therefore, the function $r\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r}}}\right]$ is strongly monotonic non-decreasing and

$$r_\sigma = \min_{1\leq r<+\infty}\left\{r\mathbb{E}\left[\tau_{\frac{\sigma}{\sqrt{r}}}\right]\right\} = 1$$

□

Combining this result together with Theorem 1 we obtain the following

COROLLARY 1. *The resampling operator is redundant for the cGA optimizing* OneMax *with additive posterior gaussian noise $\mathcal{N}(0, \sigma^2)$. In particular, the cGA reaches the optimum in $\mathcal{O}(\sigma^2)$ fitness evaluations.*

If we assume that the models presented in Observation 5 are correct, then we conclude that the resampling operator is redundant for λ-MMASib as well, regardless of the posterior noise standard deviation. Therefore, when optimizing OneMax with additive posterior gaussian noise $\mathcal{N}(0, \sigma^2)$, we cannot hope to achieve a run time with order of complexity smaller than $\mathcal{O}(\sigma^2)$, for any of the hereby tested algorithms (cf. Table 2). These results are not in contrast with the experiments displayed in Figure 7. In fact, the resampling operator gives a faster run time for $(\mu + 1)$-GA and $(\mu + 1)$-EA, but its effect is noticeable only for very large posterior noise standard deviation σ. In the case of $\sigma \leq 10$, the benefit of recombination overwhelms the effect of the resampling operator. Nevertheless, for very large σ, we expect

the $(\mu + 1)$-EA with resampling to give better performance then the $(\mu + 1)$-GA without resampling.

4. CONCLUSIONS

We compare empirically the run time behavior of four different bio-inspired search heuristics: $(\mu+1)$-EA, $(\mu+1)$-GA, cGA, and λ-MMASib. Our testbed is the fitness function OneMax + $\mathcal{N}(0, \sigma^2)$, which is a simple unimodal function with posterior noise. All algorithms have local parameters (cf. Table 1), which influence their run time (cf. Figures 1–2). We study the dependence of the parameter on the amount of noise σ and empirically determine for each algorithm the optimal parameter setting depending on σ (cf. Figures 4–5). We give statistical predictions for each parameter's asymptotic behavior.

We are then able to compare the algorithms with optimal parameter settings depending on the level of noise. We observe a strict hierarchy in how well the algorithms can deal with noise (cf. Figure 7). From worst to best this is $(\mu + 1)$-EA, $(\mu+1)$-GA, λ-MMASib, and cGA. We show statistically that the $(\mu + 1)$-EA and $(\mu + 1)$-GA have run time trend of $\mathcal{O}(e^{c\sqrt{\sigma}})$, and that the λ-MMASib and cGA have run time trend of $\mathcal{O}(\sigma^2)$. We observe that cGA and λ-MMASib have very similar run time, depending on the problem size.

A common technique to deal with noisy fitness functions is resampling. We therefore also study the optimal number of samples for a given noise level (cf. Figures 8–9). With optimal resampling we observe improved run times for the $(\mu + 1)$-EA and $(\mu + 1)$-GA, which scaled least graceful with noise. However, with optimal resampling both of them have run time $\mathcal{O}(\sigma^2)$. We prove that resampling is redundant for any algorithm with run time $\mathcal{O}(\sigma^2)$, thus showing that cGA and λ-MMASib do not benefit from this operator (cf. Lemma 1-2). Therefore, all four algorithms reach run time complexity of $\mathcal{O}(\sigma^2)$ at most (cf. Table 2).

Overall, this study shows that resampling is more beneficial than crossover, for algorithms that perform poorly in noisy environments. By far the best scaling behaviour was achieved with EDAs, suggesting that such algorithms can handle noise implicitly. We plan to validate all hereby presented statistical models analytically in the future.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 618091 (SAGE) and from the German Science Foundation (DFG) under grant agreement FR 2988 (TOSU).

References

[1] A. Aizawa and B. W. Wah. Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation*, 2(2):97–122, 1994.

[2] Y. Akimoto, S. Astete-Morales, and O. Teytaud. Analysis of runtime of optimization algorithms for noisy functions over discrete codomains. *Theoretical Computer Science*, 605:42–50, 2015.

[3] D. Arnold and H. G. Beyer. Efficiency and mutation strength adaptation of the (μ, μ_I, λ)-ES in a noisy environment. In *Proc. of PPSN '00*, pages 39–48, 2000.

[4] V. D. Arnold and G. H. Beyer. A general noise model and its effects on evolution strategy performance. *IEEE Trans. Evolutionary Computation*, 10(4):380–391, 2006.

[5] J. Branke and C. Schmidt. Selection in the presence of noise. In *Proc. of GECCO '03*, pages 766–777, 2003.

[6] J. Branke and C. Schmidt. Sequential sampling in noisy environments. In *Proc. of PPSN '04*, pages 202–211, 2004.

[7] D.-C. Dang and P. K. Lehre. Efficient optimisation of noisy fitness functions with population-based evolutionary algorithms. In *Proc. of FOGA '15*, pages 62–68, 2015.

[8] S. Droste. A rigorous analysis for the compact genetic algorithm for linear functions. *Natural Computing*, 5(4):257–283, 2006.

[9] M. J. Fitzpatrick and J. J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 3(2):101–120, 1988.

[10] T. Friedrich, T. Kötzing, M. S. Krejca, and A. M. Sutton. The benefit of recombination in noisy evolutionary search. In *Proc. of ISAAC '15*, pages 140–150, 2015.

[11] T. Friedrich, T. Kötzing, M. S. Krejca, and A. M. Sutton. Robustness of ant colony optimization to noise. In *Proc. of GECCO '15*, pages 17–24, 2015.

[12] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

[13] N. Hansen, A. S. P. Niederberger, L. Guzzella, and P. Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Trans. Evolutionary Computation*, pages 180–197, 2009.

[14] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. pages 523–528, 1998.

[15] V. Heidrich-Meisner and C. Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *Proc. of ICML '09*, pages 401–408, 2009.

[16] V. Heidrich-Meisner and C. Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *Proc. of ICML '09*, pages 401–408, 2009.

[17] M. Hellwig and H.-G. Beyer. Evolution under strong noise: A self-adaptive evolution strategy can reach the lower performance bound - the pccmsa-es. In *Proc. of PPSN '16*, pages 26–36, 2016.

[18] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In *Proc. of PPSN '96*, pages 178–187. Springer-Verlag, 1996.

[19] H. Mühlenbein and H. Voigt. Gene pool recombination in genetic algorithms. In *Meta-Heuristics*, pages 53–62. Springer US, 1996.

[20] A. Prügel-Bennett, J. Rowe, and J. Shapiro. Run-time analysis of population-based evolutionary algorithm in noisy environments. In *Proc. of FOGA '15*, pages 69–75. ACM, 2015.

[21] C. Qian, Y. Yu, Y. Jin, and Z.-H. Zhou. On the effectiveness of sampling for evolutionary optimization in noisy environments. In *Proc. of PPSN '14*, pages 302–311, 2014.

[22] P. Rolet and O. Teytaud. Bandit-based estimation of distribution algorithms for noisy optimization: Rigorous runtime analysis. In *Proc. of LION '10*, pages 97–110, 2010.

[23] P. Stagge. Averaging efficiently in the presence of noise. In *Proc. of PPSN '98*, pages 188–200, 1998.