On Robustness of Parallel Evolutionary Algorithms: Experimental Analysis

Brahim Aboutaib
Univ. Littoral Côte d'Opale, France
Mohammed V University in Rabat, Morocco
Host Supervisor: Andrew M. Sutton
University of Minnesota Duluth, Duluth, MN, USA

***Abstract*—We are interested in *rigorously* analyzing the robustness of parallel and distributed Evolutionary Algorithms (EAs) to noise. This is a report in which we describe our results towards such objective. As a first step, we report experimental results on some models of noise, and three parallel EAs solving a pseudo-boolean black-box noisy optimization problem. We consider posterior additive noise. That is the noise occurring in the fitness function. As algorithms, we consider mutation-based and crossover-based algorithms implemented in the single receiver model. We focus on the average optimization time to find the optimum of the considered problem to characterize the performances.**

Our experimental analysis indicates some advantages of the majority vote crossover based parallel EA to deal with the considered noisy problem compared with the uniform crossover operator and mutation only based parallel EA. Nevertheless, our results indicate some chances to make uniform and mutation-only based parallel EAs resilient to noise, compared to majority vote crossover, by allowing more computation power. Also, our results indicate that, at some noise levels, the optimization time of the considered parallel EA using a majority vote crossover with a large number of parents matches the optimization time of this algorithm under no noise.

***Index Terms*—Parallel Evolutionary Algorithms, Pseudo-Boolean Black-Box Optimization, Robustness.**

I. Introduction

Evolutionary Algorithms (EAs) are a generic class of randomized heuristics that mimic natural evolution, applied to different computational problems, mainly combinatorial optimization problems. They come with the advantage of requiring little about the problem at hand, being derivative-free, and can offer satisfactory results in a reasonable amount of time. Another advantage of these algorithms is their algorithmic structure, which makes devising parallel versions of these algorithms quite *straightforward*. That is, for example, by evaluating a population in parallel. Parallel EAs (p-EAs) are practical and was shown competitive results to many difficult optimization problems [ALN13].

Applying EAs to real-life optimization problems require, often, dealing with different sources of uncertainty [JB05]. Jin and Branke, [JB05], detailed four main causes and sources of uncertainty that may face an EA. The first source is the noise that occurs during solutions evaluation (e.g., simulation-based optimization). The second source of uncertainty is the design variables that can be modified. The third source is scenarios when the fitness is not computed but approximated (e.g., surrogate-based optimization). The fourth source is when the fitness function is time-dependent. In this work, we focus on the first and the second sources of uncertainty.

To the best of our knowledge, parallel and distributed EAs are not yet considered for theoretical analysis when tackling noisy problems. Though there exists a growing and active body of literature on performances of EAs under noise [Sud21], [GK16], it is limited to the serial case. Moreover, many problems where noise occurs are based on some simulation, which may be expensive to compute. Thus, it is reasonable to aim for understanding p-EAs for noisy problems so that they can be applied wisely in such situations. Another motivation to study the behavior of these algorithms under noise (the prior noise model) is situations of software or hardware failures or attacks that can alter the candidate solutions p-EAs evolves. This can be of high interest if one aims at providing EAs based optimization software in a distributed environment, where the reliability of provided solutions would be critical.

Our work towards understanding p-EAs performances under noise can be summarized in the following points.

- We study the optimization time performances of a p-EA based on the single receiver model solving a noisy problem.
- We consider the posterior noise model based on the Gaussian distribution with 0 mean and variance σ^2.
- We compare the impact of using recombination *vs.* mutation only operators in p-EAs.
- We show that the majority vote under specific tuning and up to some noise limit can make the performances of the single receiver algorithm matches its performances under no noise. We also characterize the noise level after which the single receiver model fails to solve the optimization problem under the given budget.

The rest of this report is organized as fellow. We first review the literature of EAs for noisy problems and state of the art of theoretical analysis of p-EAs in section II. Section III presents the considered p-EA and the optimization problem. The experimental protocol and results are reported in section IV. Section V closes the report by laying out the next step towards our initial objective.

II. Related Work

In this section, we review existing work related to the performances of EAs under noise. We also review the existing literature related to theoretical work on p-EAs.

A. Robustness of EAs to Noise

Research on understanding and improving the performances of EAs in a noisy environment has been carried since the early days of evolutionary computation [FG88]. Though there is a considerable amount of empirical study on this subject, we focus our review on mathematical work. The review by Jin and Branke [JB05], though quite dated, gives a good overview of the field. Two main noise models have been considered by theoreticians when analyzing EAs. The additive noise [FKKS15] and the prior model [Dro04]. We shall note that other forms of noise are also considered. For example, Dang and Lehre [DL16] studied the case of partial computation of the fitness function and its impact on the optimization time of a non-elitist EA.

Gieben and Kotzing [GK16] analyzed a set single-solution and population-based EAs under the prior noise. They proved that $(1+1)$-EA solves the noisy OneMax under the prior noise, in which one bit is flipped with a probability q in $\Theta(nlog(n))$ if q is $O(1/n)$. This time become polynomial $poly(n)$ when q is $O(log(n)/n)$, and exponential $2^{\omega(log(n))}$ when $q \in \omega(log(n)/n)$. The authors also considered the $(\mu+1)$-EA. They showed that this algorithm takes $O(\mu nlog(n))$ on the noisy OneMax, given that $\mu > 12log(15n)/q$. To improve the performance of EA under noise, one can resample the solution multiple times and average the fitness values, [QYT$^+$18], [FKQS17]. Qian et al. [QYT$^+$18] proved that resampling when used within a $(1+1)$-EA yield a polynomial optimization time $O(poly(n))$, given a sample size $m = \lceil \frac{n\sigma^2}{log(n)} \rceil$ and a noise level $1 \leq \sigma^2 \in poly(n)$.

Regarding the posterior noise case, Gieben and Kotzing [GK16] analyzed EAs under the additive guassian distribution noise. They proved that $(1+1)$-EA solves the noisy OneMax in $O(nlog(n))$, when the $\sigma^2 \leq 1/(4log(n))$. Friedrich et al. [FKKS15] analyzed the runtime of $(\mu+1)$-EA and an estimation of distribution algorithm, the compact GA (cGA) under the additive Gaussian noise. They proved a polynomial time of the cGA $O(K\sigma^2\sqrt{n}log(Kn))$ when the number of considered parents K in the recombination is $\omega(\sigma^2\sqrt{n}log(n))$, and an exponential time $O(2^{\omega(log(n))})$ of the mutation only $(\mu+1)$-EA.

B. Parallelism Models for EAs

The parallelism of an EA can be achieved in different ways [AT02]. We briefly present common approaches used to implement parallel EA. The first one relies on the Master-Worker Model [DGP06]. In this model, a central computing unit, called *Master*, executes all the algorithm steps, except computing the fitness of solutions. The latter became the charge of other computing units, called *Workers*. This model is used in scenarios where computing the solutions' quality is expensive. The second approach proceeds by dividing the population into sub-populations. This model is called the *Island Model* [DFF$^+$19]. Each computing unit runs an EA and exchanges selected solutions with its neighbors in this model. A replacement mechanism is then triggered to select solutions to consider for replacement on each node. The most used technique for solution selection for communication is elitist selection. This model is usually implemented using the message passing protocol. Another model, similar to the island model, is the single receiver [FKK$^+$16] considered in this work and detailed in section III-A.

C. Theoretical Analysis of Parallel EAs

While the empirical work on p-EAs started in the early nineties of the past century [ZK93], the first rigorous theoretical analysis goes back to [LS10]. Since then, p-EAs have started to attract growing interest from theoreticians. The considered research questions range from the impact of the considered parallel design (the topology of p-EAs) [FK18], [MS14], [MS15], to characterizing the optimization time of p-EAs under some specific setting [LS14b], [LW17], to providing general techniques for mathematical analysis of p-EAs [BLS14], [BLS15], [LS19].

For the impact of the considered topology and communication, in [FK18], authors studied the impact of distributed EAs topologies on the diversity of solutions. They found that a ring topology with probabilistic migration helps the $(1+1)$-EA running on λ island not to get stuck in local optima. They proved a $O(n^r log^2(n))$ bound on this algorithm on the $Fork_r$ problem, under some conditions on the number of islands λ. In contrast, a complete topology deteriorates the performance of the algorithm. In another study, Mambrini and Sudholt [MS14], [MS15] analyzed the $(1+1)$-EA running on λ island using adaptive strategies that control the communication intervals. Given the fitness of the local population, the island decreases or increases its communication interval. The rationale is to speed up the spread of fittest individuals. This was proven to improve the upper bounds on the communication complexity. In [LS14b], [LS10], Lässig and Sudholt adapted the fitness level technique [Weg03] to prove general upper bounds on the runtime of elitist p-EAs ($(1+1)$-EA) running on regular graphs (i.e., ring, 2D and 3D torus), using synchronous communications. They proved a bound of $O(n)$ on $(1+1)$-EA to solve the OneMax if the ring, torus, hypercube, or complete topologies are considered. The communication cost (number of messages exchanged) is deduced from these topologies and provided. The proposed method is intended to be general and applicable to other parallel optimization heuristics. This technique was extended by [LS19] to any parallel unbiased optimization algorithms that evaluate different solutions in parallel. In [DFF$^+$19], authors derived bounds on the runtime and the communication complexity of island-based p-EAs running random-like communication topologies.

Besides pseudo-boolean linear functions, in [LS14a] authors analyzed the runtime of a parallel $(1+\lambda)$-EA to find solutions for the shortest paths and Eulerian cycles. They illustrated scenarios where the parallel algorithms' runtime

varies drastically depending on how the communication is tuned. Specifically, the algorithm may have logarithmic or exponential speed depending on the communication.

III. Algorithms and Optimization Benchmark

In this section, we detail considered p-EA and the optimization problem.

A. A Parallel Evolutionary Algorithm

We are interested in analyzing the performance of p-EAs for noisy problems. We consider the following model for p-EA. Given a topology $G = (V, E)$ which abstracts computation resources (e.g., cluster of CPUs, multi-core CPU), each node $v \in V$ will run a $(1 + 1)$-EA detailed in the Algorithm 1. This is a simple EA using a mutation rate of $\frac{1}{n}$ over bit-strings. The best solution between the parent and the newly generated solution is kept for the next generation. Among the V nodes, we will consider a node $r \subset V$ that will execute a genetic algorithm or an EA (without crossover) as detailed in Algorithm 3. The r node will go through the following steps. After receiving a number $k \geq 1$ of solutions from senders, sampled uniformly at random, r will apply a crossover between received solutions. The crossover operators we consider are the majority vote and the uniform crossovers. The latter is the classical uniform crossover, in which a bit is selected from one of the two parents with a probability $\frac{1}{2}$. We abbreviate this genetic operator by *unif_x*. The other genetic operator we consider is detailed in Algorithm 2, the majority vote crossover. This operator builds an offspring from k parent by setting the bit at some position i to the 1 if it is the most recurrent among k parent at position i, else it is set to 0. This operator is abbreviated as *mv_x*. The obtained solution will then go through a selection phase. When the crossover is not used, that is a mutation only EA, after receiving k solution from parents, selected u.a.r, the best among the received and the receiver's solution is kept for the next generation. We name this operator *mut*.

Algorithm 1: $(1 + 1)$-EA for optimizing a function f as executed on each node, except the receiver.

```
Initialize x u.a.r.;
while not stop condition do
    y ← x;
    flip each bit of y independently with prob. 1/n;
    if f(y) ≥ f(x) then
        x ← y;
```

The communication step, sending and receiving of solutions between nodes and the receiver, happens periodically after a number of rounds, τ. The algorithm stops when the global optimum is reached or when a number of function evaluations are exhausted. We analyzed the robustness of this algorithm as a function of different parameters, namely the noise level, topology size, communication effort, problem size.

Algorithm 2: Majority Vote Crossover (*mv_x*)

```
X ← {x_1, x_2, ..., x_k};
y ← (0, 0, ..., 0);
for i ∈ [n] do
    O ← 0; Z ← 0;
    for j ∈ [k] do
        if x_j[i] == 1 then
            O ← O + 1;
        else
            Z ← Z + 1;
    if O > Z then
        y[i] ← 1;
return y;
```

Algorithm 3: EA with the *mv_x* optimizing a function f as executed on the receiver node.

```
Initialize x u.a.r.;
V_s, V ← {};
t ← 0;
while not stop condition do
    y ← x;
    flip each bit of y independently with prob. 1/n;
    if f(y) ≥ f(x) then
        x ← y;
    if t ≡ 0 mod τ then
        V ← sample k nodes u.a.r. for majority vote;
        V_s ← recieve solutions from V;
        z ← mv_x(V_s);
        if f(z) ≥ f(x) then
            x ← z;
    t ← t + 1;
```

B. Pseudo-Boolean Noisy Optimization Benchmark

We analyze the considered algorithms on the following artificial benchmark. This benchmark is constructed based on the OneMax function. We first detail this function, then detail the considered noise model.

The OneMax (OM) problem is a linear unimodal pseudo-boolean function defined as follow. For $x \in \{0, 1\}^n$ we define

$$f_{OM}(x) = \sum_{i=1}^{n} x_i$$

to be the fitness of x. Using this function, we will define a noisy OneMax function to understand the performances and capacities of Algorithm 3 under noise. We consider the noisy OneMax function under the following noise models.

The first model defined in equation 1, is the additive noise model, also known by posterior noise, where we add to the fitness value of a solution a term, d, sampled from the Gaussian

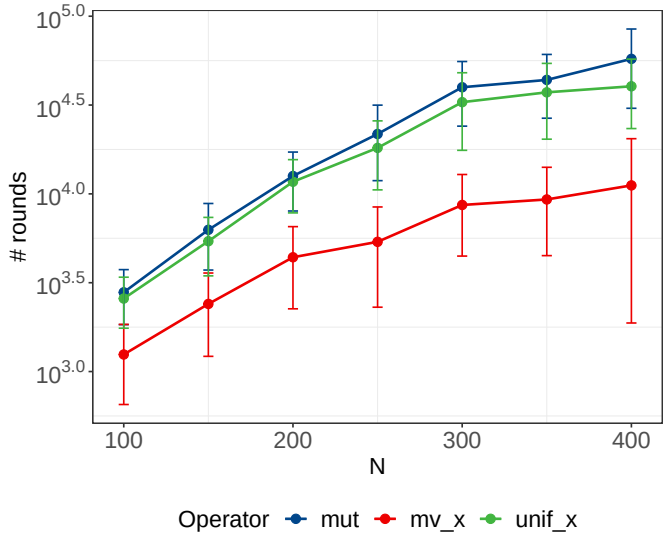

Fig. 1. Mean number of rounds as a function of problem size. Topology size is set to 32, $\tau = 8$. Number of parent for the *mut* and *mv_x* is 3, and 4 for *unif_x*. $\sigma^2 = 0.8$.

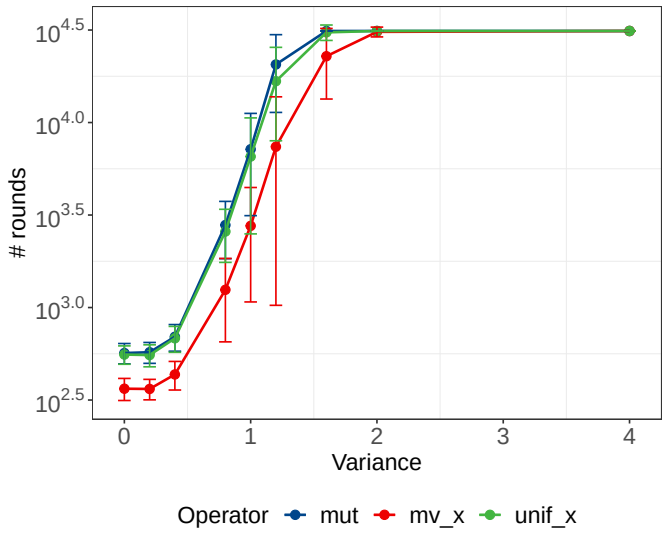

Fig. 2. Mean number of rounds of the single receiver model as a function of variance σ^2 (in log scale) to hit the optimum of OneMax for $n = 100$, for different the three operators. $\tau = 8$. Number of parents is $k = 3$ for *mut* and *mv_x*, and 4 for *unif_x*. Topology size is 32.

distribution with mean 0 and variance σ^2. Notice that d can be sampled from any distribution, not necessarily a Gaussian.

$$f_{OM}(x) = f_{OM}(x) + d, \; d \sim N(0, \sigma^2); \tag{1}$$

IV. Experimental Analysis

This section reports the experimental protocol followed and presents results and our analysis.

A. Experimental Protocol

To gain an idea about the performance of Algorithm 3 optimizing the noisy OneMax, we consider the following experiments and parameters settings. We consider OneMax instances of size $n \in \{100, 150, 200, \ldots, 400\}$ under the noise model in 1. We experiment with different values of $\tau \in \{1, 2, 4, \ldots, 32\}$, $\sigma^2 \in \{0.0, 0.2, \ldots, 4.0\}$ and topology sizes, $m \in \{4, 8, \ldots, 64\}$. We consider the case of $\sigma^2 = 0$ to compare with the scenario of p-EA optimizing the OneMax function without noise. The stop condition for the considered algorithms is either finding the (true) optimum solution or exhausting a budget of $\lceil \frac{n^3}{m} \rceil$ function evaluation, for the single receiver p-EA using m node[1]. Reported results are, mainly, the mean and the standard deviation of the number of rounds to hit the optimum, 1^n, over 100 independent runs.

B. Experimental Results

a) Average optimization time: To analyze the performance of the considered p-EA and genetic operators, we first plot the mean number of rounds as a function of the problem size in Fig. 1. Figure indicate an $nlog(n)$ growth of the optimization time as function of n when the *mv_x* is used, while a growth more fast for *mut* and *unif_x*. Notice also the factor improvement of the optimization performance gained by using *mv_x*, over *mut* and *unif_x*.

b) Performance under Noise: Fig. 2 plots the mean number of rounds to hit the optimum of Algorithm 3 (and the standard deviation) as a function of the variance for the three operators. The figure indicates that adding noise to the optimization problem increases the average time needed to find the optimal for all operators. Moreover, the algorithm stops when the $\sigma^2 \geq 2$ without finding the optimum. Also, the figure indicates exponential growth of the optimization time for all three algorithms as a function of σ^2. However, notice the slight advantage of using the majority vote over the other two operators for all noise levels.

To complete the picture of the previous figure, we report in figure 3 the proportion of successful runs as a function of noise (σ^2) for different topology sizes, when the majority crossover is used. This figure helps us gain an idea about the impact of considering more computation power to cope with a certain level of noise. We observe that at $\sigma^2 \leq 1.0$ the algorithm always finds the global optimum for the problem. At $\sigma^2 = 1.2$, we observe that a topology of size 16 always solves the problem. This is not the case for low topology sizes, 4 or 8. However, surprisingly, a topology of size 16 helps cope with noise better than a large topology, 32. This is true too, at noise level $\sigma^2 = 1.4$. The difference of resilience to noise is pronounced and remarkable when $\sigma^2 = 1.4$. At this noise level, we notice that using topologies of size, $16, 32$, makes the proportion of failed runs around 25%, while 50% and more than 75% at topologies of size 8, 4, respectively.

c) Impact of Number of Parents: Herein, we are interested in understanding the impact of the mutation and crossover operators on the optimization time. Fig. 4 plots the

[1]We divide by m so our results can be compared as a function of different topology sizes, given that we are analyzing a fixed budget scenario.

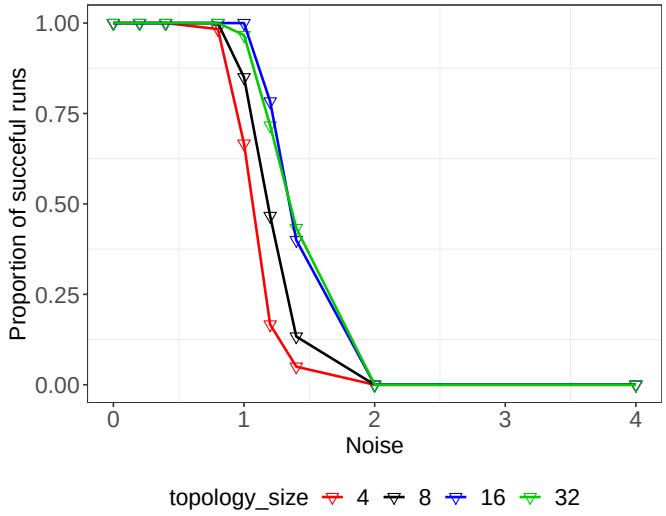

Fig. 3. Portion of successful runs in function of noise, for different topology sizes for the majority vote crossover, $k = 3$. $n = 100$

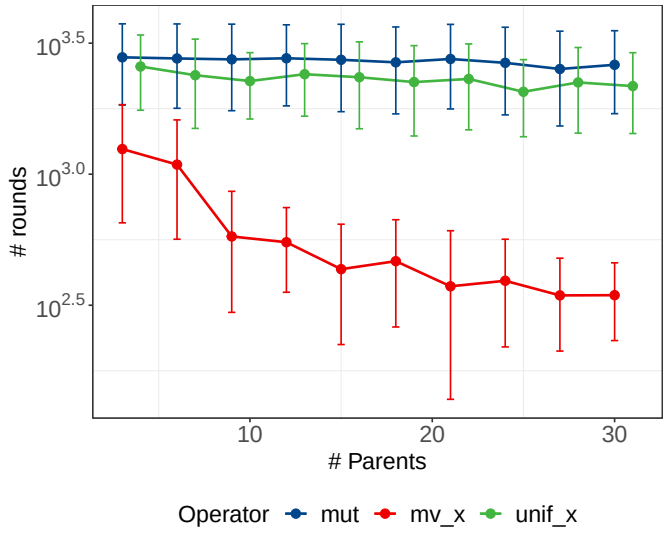

Fig. 4. Mean number of rounds (and standard deviation) as a function of the number parents used in crossover, or for selection in the mutation only algorithm. $n = 100$, $\tau = 8$, $\sigma^2 = 0.8$, and topology of size 32.

mean number of rounds to find the optimum. Two interesting points can be retained here. First, *mut* and *unif_x* perform the same independently on the number of parents, while considering more parents for *mv_x* decreases the mean optimization time. Second, considering more parents in the *mv_x* leads to an optimization time similar to $\sigma^2 = 0.0$, (the no noise scenario). Thus, this indicates that the majority vote crossover would, up to some noise level, cancel the noise, if a sufficient number of parents are provided for the crossover.

d) Parallelism parameters: Topology size and Communication intervals: Figure 5 reports the mean number of rounds to hit the optimum as a function of topology size for the three operators. We can see that more computation helps improve

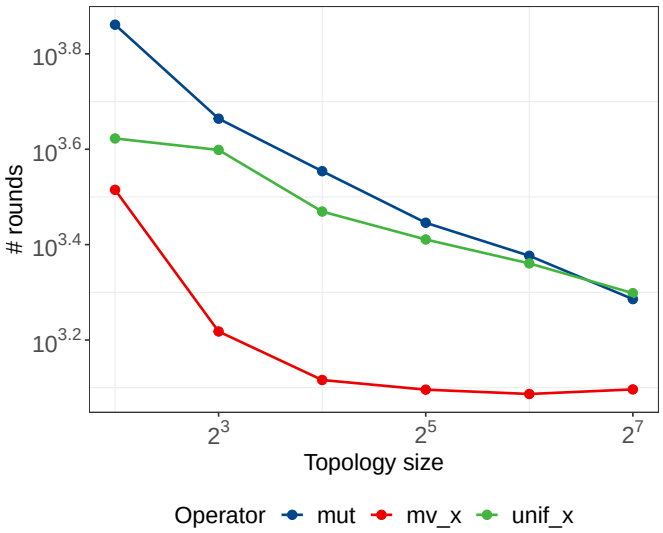

Fig. 5. Mean number of rounds as a function of topology size. Number of parent for the *mut* and *mv_x* is 3, and 4 for *unif_x*. $n = 100$. $\tau = 8$. $\sigma^2 = 0.8$.

the mean optimization time, up to some topology size in the case of *mv_x*, after which adding more computation resources does not help reduce the optimization time. This is not the case when *mut* and *unif_x* are used. The figure indicates that using more computation resources lowers, by a factor, the optimization time.

In figure 6, we report the mean number of rounds as a function of communication intervals τ. Note that communication intervals are related to the number of exchanges messages. The shortest the interval, more messages are exchanged, and hence the communication effort is considerable. From this figure, we can note that communication intervals have a little impact on the *mut* and *unif_x* performances based on the single receiver p-EA. This is not the case when the *mv_x* operator is used. We can see that *spacing* the communication intervals increase the mean number of rounds to find the optimum, while short communication intervals decrease the optimization time.

V. Conclusion and Future Work

We reported our preliminary results on the robustness of a p-EA, the single receiver p-EA when optimizing a simple noisy pseudo-boolean black-box function. We considered the posterior noise model based on the normal distribution and studied the performances of three p-EAs, considering mutation and recombination operators. We also studied the scalability of the considered p-EA and the influence of communication intervals on its performance. Overall, the experiments indicate an advantage of the majority vote crossover-based p-EA over uniform and mutation-only p-EA to deal with noise, up to some noise threshold.

As future work, we are interested in proving these results mathematically and thus generalizing our understanding to settings unreachable by experiments. Also, analyzing others models of p-EAs, and other optimization problems. For

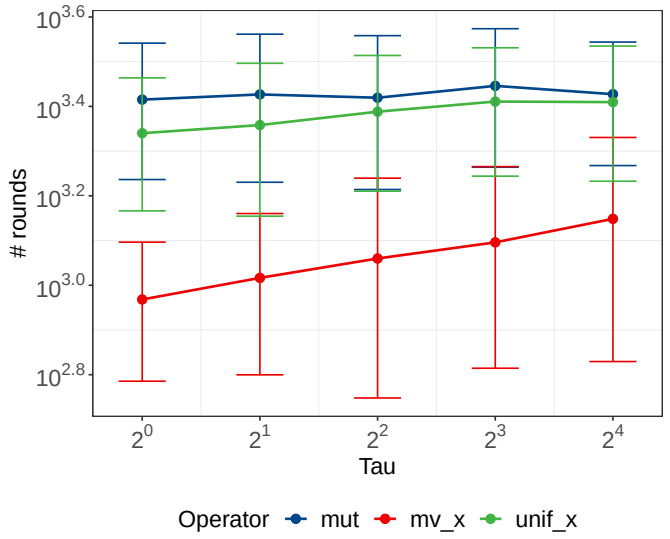

Fig. 6. Mean number of rounds(and standard deviation) as a function of the communication interval τ. $n = 100$. $\sigma^2 = 0.8$, topology of size 32. Number of parent for the *mut* and *mv_x* is 3, and 4 for *unif_x*.

instance, considering other decentralized parallelism models with different communication topologies. Indeed, from our experimental analysis, we would expect to prove an expected runtime of $O(2^{poly(\sigma^2)}nlog(n))$ for the single receiver model on the OneMax problem under the additive noise using the majority vote crossover.

ACKNOWLEDGEMENT

This research is supported by the IEEE CIS Graduate Student Research Grants.

REFERENCES

[ALN13] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow. Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research*, 20(1):1–48, 2013.

[AT02] Enrique Alba and Marco Tomassini. Parallelism and evolutionary algorithms. *IEEE transactions on evolutionary computation*, 6(5):443–462, 2002.

[BLS14] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. Unbiased black-box complexity of parallel search. In *International Conference on Parallel Problem Solving from Nature*, pages 892–901. Springer, 2014.

[BLS15] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. Black-box complexity of parallel search with distributed populations. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pages 3–15, 2015.

[DFF+19] Benjamin Doerr, Philipp Fischbeck, Clemens Frahnow, Tobias Friedrich, Timo Kötzing, and Martin Schirneck. Island models meet rumor spreading. *Algorithmica*, 81(2):886–915, 2019.

[DGP06] Marc Dubreuil, Christian Gagné, and Marc Parizeau. Analysis of a master-slave architecture for distributed evolutionary computations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(1):229–235, 2006.

[DL16] Duc-Cuong Dang and Per Kristian Lehre. Runtime analysis of non-elitist populations: From classical optimisation to partial information. *Algorithmica*, 75(3):428–461, 2016.

[Dro04] Stefan Droste. Analysis of the (1+ 1) ea for a noisy onemax. In *Genetic and Evolutionary Computation Conference*, pages 1088–1099. Springer, 2004.

[FG88] J Michael Fitzpatrick and John J Grefenstette. Genetic algorithms in noisy environments. *Machine learning*, 3(2):101–120, 1988.

[FK18] Clemens Frahnow and Timo Kötzing. Ring migration topology helps bypassing local optima. In *International Conference on Parallel Problem Solving from Nature*, pages 129–140. Springer, 2018.

[FKK+16] Tobias Friedrich, Timo Kötzing, Martin S Krejca, Samadhi Nallaperuma, Frank Neumann, and Martin Schirneck. Fast building block assembly by majority vote crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 661–668, 2016.

[FKKS15] Tobias Friedrich, Timo Kötzing, Martin Krejca, and Andrew M Sutton. The benefit of sex in noisy evolutionary search. *arXiv preprint arXiv:1502.02793*, 2015.

[FKQS17] Tobias Friedrich, Timo Kötzing, Francesco Quinzan, and Andrew M Sutton. Resampling vs recombination: A statistical run time estimation. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, pages 25–35, 2017.

[GK16] Christian Gießen and Timo Kötzing. Robustness of populations in stochastic environments. *Algorithmica*, 75(3):462–489, 2016.

[JB05] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on evolutionary computation*, 9(3):303–317, 2005.

[LS10] Jörg Lässig and Dirk Sudholt. The benefit of migration in parallel evolutionary algorithms. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, page 1105–1112, New York, NY, USA, 2010. Association for Computing Machinery.

[LS14a] Jörg Lässig and Dirk Sudholt. Analysis of speedups in parallel evolutionary algorithms and (1+ λ) eas for combinatorial optimization. *Theoretical Computer Science*, 551:66–83, 2014.

[LS14b] Jörg Lässig and Dirk Sudholt. General upper bounds on the runtime of parallel evolutionary algorithms. *Evolutionary Computation*, 22(3):405–437, 2014.

[LS19] Per Kristian Lehre and Dirk Sudholt. Parallel black-box complexity with tail bounds. *arXiv preprint arXiv:1902.00107*, 2019.

[LW17] Andrei Lissovoi and Carsten Witt. A runtime analysis of parallel evolutionary algorithms in dynamic optimization. *Algorithmica*, 78(2):641–659, 2017.

[MS14] Andrea Mambrini and Dirk Sudholt. Design and analysis of adaptive migration intervals in parallel evolutionary algorithms. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, pages 1047–1054, 2014.

[MS15] Andrea Mambrini and Dirk Sudholt. Design and analysis of schemes for adapting migration intervals in parallel evolutionary algorithms. *Evolutionary computation*, 23(4):559–582, 2015.

[QYT+18] Chao Qian, Yang Yu, Ke Tang, Yaochu Jin, Xin Yao, and Zhi-Hua Zhou. On the effectiveness of sampling for evolutionary optimization in noisy environments. *Evolutionary computation*, 26(2):237–267, 2018.

[Sud21] Dirk Sudholt. Analysing the robustness of evolutionary algorithms to noise: refined runtime bounds and an example where noise is beneficial. *Algorithmica*, 83(4):976–1011, 2021.

[Weg03] Ingo Wegener. Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In *Evolutionary optimization*, pages 349–369. Springer, 2003.

[ZK93] Bernard P Zeigler and Jinwoo Kim. Asynchronous genetic algorithms on parallel computers. In *Proceedings of the 5th International Conference on Genetic Algorithms*, page 660, 1993.