Distributed Evolutionary Algorithms with Adversarial Corruption

Brahim Aboutaib
baboutai@d.umn.edu
University of Minnesota Duluth
USA

Andrew M. Sutton
amsutton@d.umn.edu
University of Minnesota Duluth
USA

Abstract

Evolutionary algorithms are inherently parallel optimization methods that have been successfully applied to many critical systems. While our understanding of their performance has steadily progressed, most existing work on distributed evolution assumes a perfect runtime environment. This assumption does not always hold in practice, and it is important to consider how robust such systems are to attack.

We consider distributed evolutionary algorithms in a setting where a malicious adversary can confound the search process by corrupting the communication of candidate solutions in the distributed architecture. We study this setting for parallel evolutionary algorithms optimizing pseudo-Boolean functions under different adversarial models. We prove asymptotic bounds in various adversarial models that relate the number and budget of the adversary to success probability, solution quality and runtime on the OneMax problem. For a (1+λ) EA running in a distributed master/worker architecture, we prove upper bounds on the slow-down incurred by adversarial attacks as a function of population size, number of compromised nodes and adversarial budget. For a (1,λ) EA, we prove a negative result that reveals a parameter regime of the adversary that prevents efficient optimization.

We also conduct experiments on OneMax and the combinatorial problem MaxCut. On the latter, we empirically find that (1,λ) EA is more robust to adversarial corruption and, surprisingly, may even be able to leverage weak levels of corruption to outperform the adversary-free setting.

CCS Concepts

• **Theory of computation → Theory of randomized search heuristics**.

Keywords

runtime analysis, parallel evolutionary algorithms, optimization under uncertainty

ACM Reference Format:
Brahim Aboutaib and Andrew M. Sutton. 2025. Distributed Evolutionary Algorithms with Adversarial Corruption. In *Foundations of Genetic Algorithms XVIII (FOGA '25), August 27–29, 2025, Leiden, Netherlands.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3729878.3746622

1 Introduction

Optimization algorithms can often take advantage of distributed systems in order to increase their efficiency. However, distributed systems come with a degree of fragility that arises from potential hardware failure or vulnerability to attack. Evolutionary algorithms have a large degree of intrinsic parallelism [11] and can easily leverage distributed computing paradigms. Moreover, they are a popular choice in many critical systems [20], for example optimizing patient assignment in online healthcare systems [39], scheduling multi-resource satellites [45], vertical fragmentation for data privacy [18], solving large systems of equations [25], and training neural networks in the context of extremely large data sets where traditional backpropagation techniques struggle to converge [23].

Because of their use in critical systems, a clear understanding of how these algorithms behave in the presence of faulty or adversarial behavior is important. In this paper, we take steps toward a better understanding of these situations by constructing and analyzing models of distributed EAs when their interprocess communication is corrupted by malicious actors or faulty hardware. We study the classical master-worker architecture [10], which offloads the computational work of genetic operators and/or fitness evaluation to a pool of distributed worker processes. This architecture is particularly convenient for creating offspring populations since individual offspring can be created and evaluated independently [41], and we will study it in the context of distributed $(1+\lambda)$ and $(1,\lambda)$ evolutionary algorithms. The master-worker architecture was recently studied in the context of load balancing [1]. In this paper, we revisit this architecture in order to develop and study models of adversarial attacks in which a set of worker nodes are compromised and allowed to corrupt the data communicated back to the master process to effect the behavior of the optimization process.

We assume a set of preexisting but unknown compute nodes can corrupt only the genotype sent back to the master process. This is motivated by the situation in which an adversary attempts to manipulate an optimizer running in the cloud into disbursing low-quality solutions to its users, who would then incorrectly presume them to be optimized solutions. Such attacks could have far-reaching effects in fragile applications such as scheduling, where a system would unknowingly execute an inferior schedule, potentially causing a cascade of problems throughout the system of users. The effect of corrupted fitness values on optimization behavior, which we do not consider in this paper, has also been studied in the context of distributed evolutionary algorithms in the form of additive noise in the single-receiver island model [2]. However, we note that the corruption of fitness in that paper was motivated by modeling general uncertainties in the fitness computation rather than an adversarial manipulating the fitness values directly.

Our contribution. This paper presents a new model for representing adversarial attacks in a simple distributed EA in which data communicated from infected processes can be undermined in order to influence the optimization process, but only subject to some budget. We prove results on parallel versions of both the elitist (1+λ) EA and the non-elitist (1,λ) EA optimizing OneMax under the various adversarial models. For the (1+λ) EA, we prove upper bounds on the slow-down incurred by adversarial attacks as a function of population size, number of compromised nodes and adversarial budget. For the (1,λ) EA, we prove a negative result that reveals a regime for the number of compromised nodes and budget that guarantees superpolynomial runtime. We also conduct a number of experiments to fill in gaps left by theoretical results, and explore the adversarial model on the NP-hard combinatorial optimization problem MaxCut.

1.1 Related Work

The susceptibility of artificial intelligence techniques to adversarial attack has been a concern for some time, especially in the context of machine learning in which small but carefully crafted changes to model data or parameters cause significant deviation from expected behavior [35]. Moreover, as artificial intelligence algorithms increasingly leverage distributed compute architectures, the inherent attack vulnerability in distributed systems threatens these algorithms' stability and security, and it is important to design techniques that are resilient to such attacks [46].

The study of adversarial attacks for distributed optimization algorithms recently started in the field of consensus-based distributed optimization algorithms [43]. In this setting, each node in a distributed network maintains its own local cost function. Each local function is typically single-variable, continuous and convex, and the overall goal is for each node to optimize its own cost function, possibly by exchanging information with neighboring nodes. A number of works have studied the robustness of such networks to situations in which a subset of nodes deviate from their procedures due to adversarial attacks. This includes conditions on communication topology for tolerating a certain number of compromised nodes [43] as well as developing resilient optimization algorithms [47].

Population-based algorithms are natural candidates for parallelization since many operations can be executed independently on distinct individuals, thus evolutionary processes can be coordinated on distributed systems. Parallel evolutionary algorithms have been around for decades, and a number of comprehensive surveys exist, to which we direct the interested reader [3–5, 9, 22, 41, 44]. Theoretical work on parallel EAs in the context of runtime analysis started in the early 2010s with the work of Lässig and Sudholt [27–29] who conducted rigorous studies on the effect of migration, topology and other factors on the speed-up achieved by parallelization.

Cantú-Paz [9] classified parallel evolutionary algorithms into three main architectures: global single-population algorithms, fine-grained single-population algorithms (in which each individual is allocated a single processor) and coarse-grained multiple-population algorithms such as island models. The first architecture, called master-worker, is often the simplest approach to parallelizing evolutionary algorithms [10], and the architecture that we focus on in this paper.

Badkobeh et al. [7] proposed distributed black-box complexity to characterize the runtime of evolutionary algorithms evolving distributed populations in parallel to optimize a given problem. This was later refined by Lehre and Sudholt [31] who proved lower bounds on the runtime of (1+λ) EA evaluating λ solutions in parallel. Doerr et al. [13] proposed a parallelism model for EAs where receiving nodes are not fixed as in typical parallel EAs, but are selected at random over a fixed neighborhood structure (e.g., in the 2d torus, one of the four neighbors is selected to receive the message). Aboutaib and Sutton [2] analyzed the performance of parallel EAs running on the single receiver and rumor spreading topologies when optimizing the OneMax function affected with an additive noise term.

Antipov et al. [6] analyzed the runtime of serial (1+λ) EA and (1,λ) EA in the presence of prior noise. They focused on finding the noise intensity and offspring size λ so that the comma and plus selection strategies can optimize OneMax in $O(n \log n)$ time.

This paper complements the ongoing stream of research on evolutionary optimization in stochastic environments [2, 6, 12, 14, 19, 30, 40, 42] initiated by Droste [15]. The papers that followed Droste's work, although they did improve our fundamental understanding of the optimization performance of different EAs, mainly focused on understanding for what noise intensity a given EA can solve (find the global optimum) OneMax or LeadingOnes in polynomial time.

In this work we model an adversarial attack as a worker node that lies about the fitness of the string it has computed, but this lie is plausible because the fraudulent fitness was produced by the same process used by the uncompromised nodes produce strings. The string returned by the process is corrupted, while the original fitness remains unaffected. This attack could not be detected by analyzing statistical discrepancies in the fitness values returned from the worker nodes alone without having to explicitly reevaluate the strings.

We point out that this model is quite distinct from the *prior noise* model in which the fitness of a solution is perturbed but the solution is not. In our case, the solution is perturbed, but the fitness is not. This creates distinct behavior, especially late in the optimization process. For example, consider the optimization of OneMax when there are only a small number of bits remaining to solve. Prior noise models would typically discount good solutions by giving them a smaller fitness than their true fitness, while the adversarial model would assert the fitness of a solution is much better than it actually is. Indeed, it would theoretically be possible to detect a subset of nodes that were corrupting the returned fitness values via prior noise (or posterior noise) as these fitness values would necessarily come from a different distribution. As mentioned above, this fitness corruption would be statistically invisible in the fitness values returned by the adversarial nodes.

We also make an effort in this work to understand how different attack models impact how well parallel algorithms can approximate the optimal solution after a reasonable amount of time. This is important for a better understanding of the performance of parallel EAs in regimes where they have superpolynomial runtime.

2 Preliminaries

We consider optimization of pseudo-Boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$. Without loss of generality, we assume that f is to be maximized,

Algorithm 1: Parallel master-worker EA.

```
$g_0 \leftarrow (x, f_x)$, $x$ chosen uniformly at random
$t \leftarrow 0$
repeat
    Let $(x, f_x) = g_t$
    broadcast $x$ to workers
    for $i \in \{1, \ldots, \lambda\}$ in parallel do
        receive $x$ from master process
        $y^{(i)} \leftarrow$ MUTATE($x$)
        $f_y^{(i)} \leftarrow f(y^{(i)})$
        send $\left(y^{(i)}, f_y^{(i)}\right)$ to master process
    receive $P \leftarrow \left\{\left(y^{(1)}, f_y^{(1)}\right), \ldots, \left(y^{(\lambda)}, f_y^{(\lambda)}\right)\right\}$ from workers
    $g_{t+1} \leftarrow$ SELECT($g_t$,$P$)
    $t \leftarrow t + 1$
until stopping condition
```

Algorithm 2: SELECT(g_t, P)

```
input: a genotype $g_t = (x, f_x)$
input: an offspring population $P = \left\{\left(y^{(i)}, f_y^{(i)}\right) : i \in [\lambda]\right\}$
Function PLUS-SELECT($g_t$,$P$):
    $k \leftarrow \arg\max_i \left\{f_y^{(i)}\right\}$ breaking ties uniformly at random
    if $f_y^{(k)} > f_x$ then
        $g \leftarrow \left(y^{(k)}, f_y^{(k)}\right)$
    else
        $g \leftarrow g_t$
    return $g$
Function COMMA-SELECT($g_t$,$P$):
    $k \leftarrow \arg\max_i \left\{f_y^{(i)}\right\}$ breaking ties uniformly at random
    return $\left(y^{(k)}, f_y^{(k)}\right)$
```

i.e., we are seeking a point x^* such that $f(x^*) = \max_x f(x)$. The OneMax function over binary strings of length n is defined as $\text{OneMax}(x) := \sum_{i=1}^{n} x_i$. For $n \in \mathbb{N}^+$, $[n]$ denotes the interval $[1, n]$.

2.1 The Master-Worker Architecture

The master-worker architecture for parallelization of evolutionary algorithms dates back to early work by Bethke [8] in the 1970s and later by Grefenstette [21]. In this setting, a centralized "master" process manages a single population, as well as population-level bookkeeping such as selection. The work of individual-level operations are offloaded to a number of worker processes. This model can be implemented efficiently on shared-memory multiprocessors as well as distributed-memory systems since it does not require assumptions about the underlying parallel architecture [9]. The framework has also been successful in machine learning for distributed versions of stochastic gradient descent [16, 32, 33].

In this paper, we study (1+λ) and (1,λ) EA variants of the master-worker model. These variants maintain a single parent individual (population of size one), and offload the task of generating offspring and computing their fitness to a pool of λ worker nodes. In each generation, the master process synchronously broadcasts the current parent individual to the worker nodes. The offspring are then generated and evaluated in parallel by each worker, and the resulting offspring and fitness value are subsequently communicated back to the master process. The master process then selects out of these results the individual to become the next parent, depending on the selection strategy used.

Since the fitness of a bit string is calculated by a worker process and must be stored along with the string it is natural for us to represent each individual genotype as a pair (x, f_x) where f_x is a record of the fitness evaluation calculated by a worker for x. Throughout the paper, we will refer to f_x as the *stored fitness* of the genotype, which may be distinct from the *true fitness* $f(x)$. The framework for a master-worker parallel EA is listed in Algorithm 1.

The specific EA type depends on the survival selection employed in line 12 of Algorithm 1. For the (1+λ) EA variant, SELECT is implemented as the function PLUS-SELECT listed in Algorithm 2. In this case, survival selection is imposed on both the offspring population and the parent, ensuring that the algorithm is *elitist.* Note also that in line 3 we use strict selection, i.e., the current parent is only replaced by an offspring with a strictly better fitness value. This significantly eases the analysis later, but there also may be good reason to employ strict selection in this scenario since it reduces the number of times the parent is replaced. This is relevant because an adversarial attack only can have an effect when the parent is replaced by a corrupt offspring.

Survival selection that always replaces the parent with the best offspring (disregarding the parent's fitness completely) leads to the non-elitist (1,λ) EA variant, and this is achieved by implementing the COMMA-SELECT function for SELECT in Algorithm 2.

2.2 Adversarial Attack Model

Our goal is to conceive a tractable but realistic model of adversarial attacks on the master-worker model. In this model, we will assume that a set of $\ell < \lambda$ worker nodes have been compromised. On these compromised compute nodes, the data communicated back to the master process can be altered in some way to deceive the optimization process. Moreover, we control the power of an adversary by imposing a budget on the magnitude of perturbations they can induce (i.e., a limit on the number or distribution of the bits they can modify). This is a realistic assumption, as an adversary would seek to make the minimum number of perturbations necessary to disrupt the optimization algorithm in order to save computational resources or evade detection. Furthermore, from the designer's perspective, one would aim to design a robust optimization algorithm under the hypothesis that the adversary uses its entire budget.

If a worker node is compromised, an adversary has the opportunity to poison the data it sends back to the master process by altering the offspring in some way, subject to a budget b in the number of bits it can alter. This is achieved by replacing line 10 in Algorithm 1 with the line

```
send $\left(\textsc{Corrupt}(y^{(i)},b), f_y^{(i)}\right)$ to master process
```

We consider three adversarial models for corrupting the offspring returned to the master process.

Directional Choice Model $\textsc{Corrupt}(y,b)$ chooses the b bits to create an individual y with the largest fitness difference $f(y) - f(y')$.

Uniform Choice Model $\textsc{Corrupt}(y,b)$ inverts exactly b bits of y, but they must be chosen uniformly at random.

Uniform Flip Model $\textsc{Corrupt}(y,b)$ flips each bit uniformly with probability b/n. In this case, b is no longer a hard budget, but the magnitude of the perturbation is limited in expectation.

The Directional Choice Model permits the adversary the most power in the sense that it knows the fitness landscape and is able to make the most destructive perturbation (in terms of f) possible within the budget. The other two models limit the power of the adversary by forcing it to resort to stochastic data corruption. Of course there are multiple other ways to design adversarial attacks, but we focus on these three in this work.

2.3 Corrupt and degraded genotypes

We say a genotype $g = (x, f_x)$ is *corrupt* when $f(x) \neq f_x$, and a corrupt genotype is *degraded* when $f(x) < f_x$.

One challenge to the analysis of algorithm dynamics in this model is that it is always possible for an algorithm to generate an optimal solution but never identify it as fitter than suboptimal solutions. For example, suppose x^* is a global maximum of f. It is possible that the parent population consists of a genotype (y, f_y) where $f(y) < f(x^*)$, but $f_y = f(x^*)$ because (y, f_y) is corrupt. Thus if an uncompromised worker produces x^* as an offspring and sends it back to the master process, it would be rejected by Plus-Select. Technically, there is no mechanism for the algorithm to identify that an optimal solution was even generated.

In the next section, we will analyze a number of scenarios for the OneMax benchmark function and consider the behavior of the algorithms in terms of apparent fitness value. It is important to keep in mind that an algorithm that has hit an apparent global optimum is not necessarily at a true global optimum. There are two ways to reconcile this. First, one could easily bound how far from the true optimum a potentially degraded genotype is. In other cases, it might be useful to consider an oracle that recognizes true optimal solutions and determine whether or not the algorithm would ever actually generate a true optimum.

3 Results on OneMax

We consider the setting where ℓ worker nodes are compromised and corrupt the bit string returned to master process according to one of the adversarial models defined in Section 2.2. The following lemma bounds the probability that an offspring was sent from a compromised node.

Lemma 1. *Consider a run of the (1+λ) EA or the (1,λ) EA with ℓ compromised nodes. In any generation, let $(y^{(k)}, f_y^{(k)})$ be chosen uniformly at random among all the offspring with the highest stored fitness. Then the probability that $y^{(k)}$ was sent from a compromised node is exactly ℓ/λ.*

Proof. Let $f_y^{(1)}, \ldots, f_y^{(\lambda)}$ be the sequence of stored fitness values returned by the workers in an arbitrary generation. Since the mutation process is identical on every worker process, and a compromised node only corrupts the bit string that is sent back to the master (and not the fitness value), we may characterize the sequence of stored values by λ i.i.d. samples from some distribution.

Let $k = \arg\max_i \left\{f_y^{(1)}, \ldots, f_y^{(\lambda)}\right\}$ with ties broken uniformly at random, and for all $i \in [\lambda]$, denote the event $\{k = i\}$ by $\mathcal{E}_i$. Since the samples are i.i.d., $\Pr(\mathcal{E}_i) = \Pr(\mathcal{E}_j) = 1/\lambda$. Moreover, the events $\mathcal{E}_i$ are all pairwise disjoint, so letting $S \subseteq [\lambda]$ denote the set of compromised nodes, the probability that $(y^{(k)}, f_y^{(k)})$ was sent from a compromised node is $\Pr\left(\bigcup_{i\in S} \mathcal{E}_i\right) = |S|/\lambda = \ell/\lambda$. □

3.1 (1+λ) EA on OneMax

We consider two different types of mutation for the (1+λ) EA. Standard mutation produces an offspring from x by flipping each bit independently with probability $1/n$, where random local search (RLS) mutation produces an offspring from x by choosing exactly one index $i \in [n]$ uniformly at random and flipping the bit at x_i.

Lemma 2 estimates the probability that the (1+λ) EA makes progress from a degraded genotype.

Lemma 2. *Consider the time-t parent genotype $g_t = (x, f_x)$ and let $r \coloneqq f_x - f(x)$. Suppose g_t is degraded, i.e., $0 < r < n$. Then the probability that at least one worker node sends an offspring (y, f_y) where $f_y > f_x$ is*

1. *zero for RLS mutation;*
2. *at least $1 - \exp(-\lambda/(en^{r+1}))$ for standard mutation.*

Proof. To produce an offspring y with $f(y) > f_x$, a worker node must flip at least $r+1$ bits of x to create y. Note that a compromised node might return a string less fit than $f(y)$, but the recorded fitness value f_y will remain the same.

In the case of RLS mutation, the probability of flipping at least $r + 1 \geq 2$ bits is zero.

For standard mutation, a worker node produces an offspring y with $f(y) > f_x$ by flipping at least $r+1$ zero bits. The probability of this is at least $\left(\frac{1}{n}\right)^{r+1}\left(1 - \frac{1}{n}\right)^{n-r-1} \geq \frac{1}{en^{r+1}}$. The probability that at least one of the worker nodes produces such an offspring is thus at least

$$1 - \left(1 - \frac{1}{en^{r+1}}\right)^{\lambda} \geq 1 - \exp(-\lambda/(en^{r+1})). \qquad \square$$

3.1.1 *RLS Mutation, uniform choice.*

The recorded fitness in a corrupt genotype will usually misrepresent the true fitness of the underlying string to some degree. This can easily lead to deceptive conditions in which optimization heuristics become trapped, even on simple functions such as OneMax. To demonstrate this, we start with the simple setting of the (1+λ) EA using RLS mutation together with the uniform choice adversarial model with budget $b = 1$. We point out that the results in this section depend critically on the strict selection setting of Plus-Select. By Lemma 2, under RLS mutation, the process will no longer make progress as soon as the master node accepts a degraded offspring. We say the process is *trapped* in generation t when $g_t = (x, f_x)$ with $f(x) < f_x$.

In this setting, an interesting question is what is the likelihood that the algorithm can successfully make a significant amount of progress before it becomes trapped, and how does this likelihood depend on the number of adversaries? The following lemma establishes bounds on the probability of making k fitness improvements on OneMax as a function of the ratio ℓ/λ.

Lemma 3. *Consider the parallel (1+λ) EA with RLS mutation optimizing OneMax in the uniform choice adversarial model using a budget $b = 1$ and $0 < \ell < \lambda$ compromised nodes.*

Starting from an initial solution $g_0 = (x_0, f_{x_0})$, let p_k, $k < n - |x_0|$ denote the probability that the algorithm makes at least k fitness improvements before becoming trapped or hitting the global optimum. Then for all constants $0 < \delta < 1/4$ and n sufficiently large,

$$\left(1 - \frac{\ell}{\lambda}\right)^k \leq p_k \leq \left(1 - \frac{\delta\ell}{\lambda}\right)^k.$$

Proof. Suppose the process is not yet trapped in generation t, i.e., $g_t = (x, f_x)$ and $f(x) = |x| \geq f_x$. Let $g_{t+1} = (y, f_y)$. Then either $f_x = f_y$ (in which case $x = y$, due to strict selection), or $f_y > f_x$.

Since the stored fitness distribution is identical on each node (cf. the proof of Lemma 1), we may appeal to the principle of deferred decisions [36] and assume the choice of whether or not an offspring comes from a corrupt node to be revealed after selection. This allows us to consider a (theoretical) intermediary individual $g'_t = (y', f_y)$ where $|y'| = f_y$, and y' is only subsequently changed if the offspring came from a corrupted node. In this case, in order to be accepted, we must have $f_y = |y'| = |x| + 1$. Now in order for the process to become trapped, the offspring must come from a corrupted node, and the corruption must degrade y', specifically $|y| = |y'| - 1$. The probability that the offspring comes from a corrupted node is ℓ/λ by Lemma 1, and the degradation probability for the uniform choice adversarial model with $b = 1$ is $|y'|/n$. Thus, conditioning on the event $\{f_y > f_x\}$, the probability the process becomes trapped is $\frac{\ell(|x|+1)}{\lambda n}$.

Since a fitness increase is a necessary (but not sufficient) condition for the process becoming trapped, we can compute the probability that a fitness increase does *not* cause the process to become trapped as

$$1 - \frac{(|x| + 1)\ell}{n\lambda}.$$

As $|x| \leq n - 1$, this probability is at least $1 - \ell/\lambda$. To estimate the upper bound, we condition on the event that the initial fitness is at least $2\delta n$. In that case, we have $|x| + 1 > 2\delta n$. Applying a Chernoff bound on the fitness of the initial solution, the initial fitness is at least $2\delta n$ with probability $1 - e^{-\Omega(n)}$. By the law of total probability, the fitness increase probability is at most $(1 - 2\delta\ell/\lambda)(1 - e^{-\Omega(n)}) + e^{-\Omega(n)} \leq (1 - \delta\ell/\lambda)$ for n sufficiently large. The claimed bounds follow since the probability of accepting a degraded offspring is independent in each generation. □

Lemma 3 allows us to determine the success probability of the process in terms of reaching fitness level n, that is, a genotype $g = (x, f_x)$ with $f_x = n$. Note that because of strict selection, the true fitness $f(x)$ of g is at least $n - 1$. The following theorem establishes asymptotic regimes (in n) for the quantity λ/ℓ and the probability of hitting fitness level n before becoming trapped.

Theorem 1. *Consider the parallel (1+λ) EA with RLS mutation optimizing OneMax in the uniform choice adversarial model using a budget $b = 1$ and $0 < \ell < \lambda$ compromised nodes.*
The algorithm reaches fitness level n on OneMax with

(1) *constant probability if $\lambda/\ell = \Theta(n)$,*
(2) *probability at least $1 - o(1)$ if $\lambda/\ell = \omega(n)$,*
(3) *probability at most $o(1)$ if $\lambda/\ell = o(n)$.*

Proof. In order to hit fitness level n, the algorithm must make at most n fitness improvements, so the hitting probability is at least $p_n \geq (1 - \ell/\lambda)^n \geq 1 - n\ell/\lambda$ by Lemma 3 and Bernoulli's inequality.

To estimate an upper bound, suppose x_0 is the initial bit string. To hit level n, the process must successfully increase the level at least $(n - |x_0|)/2$ times since the fitness level can increase by at most two levels. Conditioning on $|x_0| \geq \delta n$, and again applying Lemma 3, the hitting probability is at most $(1 - \delta\ell/\lambda)^{\delta n/2} \leq \frac{1}{1+\delta^2 n\ell/(2\lambda)}$.

When $\lambda/\ell = \Theta(n)$, the above estimates translate to $p_n = \Omega(1)$. When $\lambda/\ell = \omega(n)$, then $p_n \geq 1 - n\ell/\lambda = 1 - o(1)$. Similarly, when $\lambda/\ell = o(n)$, $p_n \leq \frac{1}{1+\omega(1)} = o(1)$. □

Using Lemma 3, it is straightforward to bound the expected fitness level the process reaches for sufficiently small ratios of adversarial workers.

Theorem 2. *Consider the parallel (1+λ) EA with RLS mutation optimizing OneMax in the uniform choice adversarial model using a budget $b = 1$ and $0 < \ell < \lambda$ compromised nodes. For any constant $0 < \epsilon < 1$, the expected fitness level the process reaches is at least*

(1) *$(1 - \epsilon)n$ if $\ell/\lambda < \epsilon/n$, and*
(2) *$(1 - o(1))n$ if $\ell/\lambda < 1/n^{1+\epsilon}$.*

Proof. Let $X \in \{0, \dots, n\}$ denote the random variable that corresponds to the fitness level that the process reaches before it becomes trapped. By Markov's inequality $\mathrm{E}[X] \geq n \Pr(X \geq n)$. From Lemma 3 we have $\Pr(X \geq n) \geq (1 - \ell/\lambda)^n$. The claimed bounds follow since $\ell/\lambda < \epsilon/n$ implies $\Pr(X \geq n) \geq 1 - \epsilon$, and $\ell/\lambda < 1/n^{1+\epsilon}$ implies $\Pr(X \geq n) \geq 1 - n^{-\epsilon} = 1 - o(1)$. □

3.1.2 *Standard uniform mutation.*

In the more interesting case of standard uniform mutation, the (1+λ) EA has a possibility of repairing a genotype that has been degraded by an adversarial. However, depending on the intensity of the degradation, the process may need to spend many generations until a satisfactory repair is made. It is interesting to determine how large λ must be in the different adversarial scenarios for the process to remain efficient.

The following fitness-level based theorem gives a technique for bounding the amount of slow-down incurred by the different adversaries.

Theorem 3. *Let $A_i = \{(x, i) : x \in \{0, 1\}^n\}$ for every $i \in [n]$. For the (1+λ) EA using standard uniform mutation, denote as $T_i = |\{t : g_t \in A_i\}|$ the random variable that tracks the number of generations spent in stored fitness level $f_x = i$. Let T_i^* be the number of generations that an adversary-free process ($\ell = 0$) spends in A_i. Then for* *both the directional choice adversarial model and the uniform choice adversarial model with budget $b > 0$, we have*

$$\mathrm{E}[T_i] \le \begin{cases} \left(1 - \frac{3\ell}{4\lambda}\right)\mathrm{E}[T_i^*] + \frac{\ell}{\lambda}\left(\frac{en^{b+1}}{\lambda} + 1.59\right) & \text{if } \lambda < n^{b+1}; \\ \left(1 - \frac{3\ell}{4\lambda}\right)\mathrm{E}[T_i^*] + \frac{3.25\ell}{\lambda} & \text{if } \lambda \ge n^{b+1}. \end{cases} \quad (1)$$

Proof. We partition each A_i for $i \in \{0, \dots, n\}$ into sub-partitions $A_{ij} = \{(x, i) : f(x) = j\}$. Since selection is strict, if a genotype $g \in A_i$ is generated, it would not be replaced until a genotype $g' \in A_j$ where $j > i$ is accepted. It is also possible that A_i is skipped during a run, in which case $T_i = 0$. Let g be the first genotype from A_i encountered by the (1+λ) EA. Then

$$\mathrm{E}[T_i] = \sum_{j=0}^{n} \mathrm{E}[T_i \mid g \in A_{ij}] \Pr(g \in A_{ij}).$$

In order for $g \in A_{ii}$, either the previous generation accepted an offspring from an uncorrupted node, or, in the uniform choice model, the corruption reverses the effect of mutation resulting in the correct fitness evaluation. By Lemma 1, the former happens with probability $(1 - \ell/\lambda)$. Otherwise, in the uniform choice model, in order to reverse the effect of mutation, the adversary would need to flip exactly $b/2$ zeros and $b/2$ ones which occurs with probability $(|x|/n)^{b/2}(1 - |x|/n)^{b/2} \le 4^{-b}$ if b is even. If b is odd, or in the directional choice model, this probability is zero. In any case, the probability that a corrupted node sends such a genotype is at most $4^{-b}\ell/\lambda \le \ell/(4\lambda)$. It follows that $\Pr(g \in A_{ii}) \le 1 - \ell/\lambda + \ell/(4\lambda)$, so

$$\mathrm{E}[T_i] \le \left(1 - \frac{3\ell}{4\lambda}\right) E[T_i \mid g \in A_{ii}] + \sum_{\substack{j=0 \\ j \ne i}}^{n} \mathrm{E}[T_i \mid g \in A_{ij}] \Pr(g \in A_{ij}).$$

Note that when $g = (x, i) \in A_{ii}$, we have $f(x) = i$ and under this event, T_i is the waiting time until an improving fitness value is produced via mutation from x. This is simply T_i^*. Thus we have

$$\mathrm{E}[T_i] \le \left(1 - \frac{3\ell}{4\lambda}\right) E[T_i^*] + \sum_{\substack{j=0 \\ j \ne i}}^{n} \mathrm{E}[T_i \mid g \in A_{ij}] \Pr(g \in A_{ij}).$$

It remains to bound the time until a corrupt offspring is successfully repaired in each adversarial model. For both the directional choice model and the uniform choice model, we argue that for $i \ne j$,

$$\mathrm{E}[T_i \mid g \in A_{ij}] \le \frac{1}{1 - e^{-\lambda/(en^{b+1})}}. \quad (2)$$

In the directional choice model, if g is corrupt, then $g \in A_{ij}$ almost surely where $j = \max\{0, i - b\}$. Thus to escape A_i, a worker must produce an offspring by flipping at most $b + 1$ bits, the probability of which is at most $1 - e^{-\lambda/(en^{b+1})}$ by Lemma 2. The waiting time to escape level i is geometrically distributed and the bound on its expectation is given by Equation (2).

For the uniform choice model, if $g = (x, i)$ is corrupt, then $g \in A_{ij}$ where $j \in [i - b, i + b] \setminus \{i\}$. If $j > i$, then it is sufficient to escape A_i just by copying x, which any worker can do with probability $(1 - 1/n)^n \ge 1/(2e)$ in which case the stored fitness returned is $j > i$. On the other hand, if $j < i$, then to escape level i, a worker must produce an offspring by flipping at least $i - j + 1$ zero bits. The waiting time for this event is maximized at $j = \max\{0, i - b\}$, and Equation (2) holds again by applying Lemma 2 and the expectation of a geometric distribution.

We thus have, for both the directional choice and the uniform choice model,

$$\begin{aligned} \sum_{\substack{j=0 \\ j \ne i}}^{n} \mathrm{E}[T_i \mid g \in A_{ij}] \Pr(g \in A_{ij}) &\le \frac{1}{1 - e^{-\lambda/(en^{b+1})}} \sum_{\substack{j=0 \\ j \ne i}}^{n} \Pr(g \in A_{ij}) \\ &= \Pr(g \notin A_{ii}) \frac{1}{1 - e^{-\lambda/(en^{b+1})}} \\ &\le \left(\frac{\ell}{\lambda}\right) \frac{1}{1 - e^{-\lambda/(en^{b+1})}}. \end{aligned}$$

To complete the proof, we calculate the remaining terms in Equation (1) by bounding the RHS of Equation (2) from above. These bounds depend directly on λ and n^{b+1}. When $\lambda < n^{b+1}$, we may apply the well-known exponential bound $e^z \le 1+z+z^2$ for $z < 1.79$, indeed $1 - e^{-\lambda/(en^{b+1})} \ge \frac{\lambda}{en^{b+1}} - \left(\frac{\lambda}{en^{b+1}}\right)^2$, and the RHS of Equation (2) is at most

$$\frac{en^{b+1}}{\lambda} + \frac{en^{b+1}}{en^{b+1} - \lambda} \le \frac{en^{b+1}}{\lambda} + \frac{e}{e - 1} \le \frac{en^{b+1}}{\lambda} + 1.59.$$

If $\lambda \ge n^{b+1}$, the above estimate fails because the lower bound for $1 - e^{-\lambda/(en^{b+1})}$ drops below zero. However, in this case we would have $e^{-\lambda/(en^{b+1})} \le e^{-1/e}$. Thus the RHS of Equation (2) is bounded above by $1/(1 - e^{-1/e}) < 3.25$. □

For the uniform flip adversary, $\Pr(g \in A_{ij})$ is nonzero for all j, as there is a positive probability that the adversary could degrade a genotype to any sub-level. This likely requires a different approach to derive good bounds on the escape probabilities and we do not treat that case in the current paper.

Note that the total waiting time over all fitness levels in the adversary-free setting is just the standard running time of the (1+λ) EA. This can be bounded [26] by $(en/\lambda)H_n$ where H_n is the n-the Harmonic number, and thus we may derive the following corollary that illustrates the potential slow down in terms of the number of compromised workers ℓ, the adversary's budget b and the total number of worker processes λ.

Corollary 4 (to Theorem 3). *Consider the parallel (1+λ) EA with standard mutation optimizing OneMax in the directional or uniform choice adversarial model using a budget b and $0 < \ell < \lambda$ compromised nodes.*

The expected number of generations until the algorithm generates a genotype (x, n) on the highest fitness level is at most

$$\left(1 - \frac{3\ell}{4\lambda}\right)\left(n + \frac{en(\ln n + 1)}{\lambda}\right) + \frac{\ell}{\lambda}\left(\frac{en^{b+1}}{\lambda} + 1.59\right)$$

if $\lambda < n^{b+1}$, and

$$\left(1 - \frac{3\ell}{4\lambda}\right)\left(n + \frac{en(\ln n + 1)}{\lambda}\right) + \frac{3.25\ell}{\lambda}$$

if $\lambda \ge n^{b+1}$.

3.2 (1,λ) EA on OneMax

The results of the previous section reveal how elitism can cause the algorithm to get trapped, and potentially require a long (possibly infinite) waiting time to escape a situation in which the process makes no progress due to receiving a corrupted genotype. In general, a non-elitist approach can be more robust to the corruption process. One clear advantage of a non-elitist algorithm is that even if the current parent is degraded with optimal fitness, it is still possible to later obtain a true optimum. However, if the adversary budget or the number of compromised nodes is high enough, then the algorithm can fail. In this section, we investigate this effect for the (1,λ) EA with respect to a critical parameter that quantifies the intensity of the perturbation as measured by the number of compromised nodes ℓ multiplied by the budget b. We begin with the following lemma that bounds the expected best fitness of the offspring population.

Lemma 4. *Let $x \in \{0,1\}^n$ and suppose λ offspring $\{y^{(1)}, \dots, y^{(\lambda)}\}$ are generated from x via standard uniform mutation. Let $X^* \coloneqq \max_i\{|y^{(i)}| - |x|\}$. Then*

$$\mathrm{E}[X^*] \le \frac{\ln\lambda}{\ln 2} - \frac{3|x|/2 - n}{n\ln 2}$$

Proof. Let $k \coloneqq |x|$ and let $X_1, \dots, X_\lambda$ denote the sequence of λ random variables $X_i = |y^{(i)}| - |x|$. Then $\mathrm{E}[X^*] = \mathrm{E}[\max_i\{X_i\}]$. Note that each X_i is i.i.d. distributed with the difference of binomial distributions $X_i \sim B(n-k, 1/n) - B(k, 1/n)$. For a random variable X, let $M_X(t) = \mathrm{E}[e^{tX}]$ denote its moment generating function. By Jensen's inequality, we have

$$e^{t\,\mathrm{E}[X^*]} \le \mathrm{E}[e^{tX^*}] \le \mathrm{E}\left[\sum_{i=1}^{\lambda} e^{tX_i}\right] = \lambda M_{X_1}(t),$$

since the X_i's are i.i.d. Furthermore, the moment generating function for X_1 is given by

$$\begin{aligned} M_{X_1}(t) &= M_{B(n-k,1/n)}(t) \cdot M_{B(k,1/n)}(-t) \\ &= \left(1 - \frac{1}{n} + \frac{e^t}{n}\right)^{n-k} \cdot \left(1 - \frac{1}{n} + \frac{e^{-t}}{n}\right)^{k}. \end{aligned}$$

Substituting this yields the bound

$$\begin{aligned} \mathrm{E}[X^*] &\le \frac{\ln(\lambda M_{X_1}(t))}{t} \\ &= \frac{\ln\lambda}{t} + \frac{(n-k)\ln\left(\frac{n-1+e^t}{n}\right) - k\ln\left(\frac{n}{n-1+e^{-t}}\right)}{t}. \end{aligned}$$

Setting $t \coloneqq \ln 2$, we have

$$\mathrm{E}[X^*] \le \frac{\ln\lambda}{\ln 2} + \frac{(n-k)}{\ln 2}\ln\left(\frac{n+1}{n}\right) - \frac{k}{\ln 2}\ln\left(\frac{n}{n-1/2}\right).$$

The claimed bound holds due to $\ln((n+1)/n) < 1/n < 2\ln(n/(n-1/2))$ for all $n > 1$. □

We now prove a negative result for the (1,λ) EA that reveals a regime for the parameter ℓb that guarantees superpolynomial runtime.

Theorem 5. *Consider the parallel (1,λ) EA with standard uniform mutation optimizing OneMax with $0 < \ell < \lambda$ compromised nodes, each with a budget of $0 < b \le \log n$. Assume that $\lambda > 4\ln(4n/b)$. The process requires a superpolynomial number of generations to find an optimal solution when*

1. *$\ell b \ge \frac{\lambda\ln\lambda}{\ln 2}$ for the directional choice model, and*
2. *$\ell b \ge \frac{4\lambda\ln\lambda}{\ln 2}$ for the uniform flip model.*

Proof. We consider the drift of the true fitness of the process and bound the time until a true optimal solution is first accepted.

Let $g_t = (x, f_x)$ be the current genotype, and denote as $g_{t+1} = (y, f_y)$ the next genotype produced in generation $t+1$. Again we appeal to the principle of deferred decisions [36] and consider a two-step process in which an offspring y' is first generated and accepted, i.e., $f_y = |y'|$, and then the (possibly corrupt) string y is generated from y'.

Let $(X_t)_{t\ge 0}$ be the stochastic process that tracks the Hamming distance of the genotype to the true optimum, i.e., $X_t = n - |x|$ in $g_t = (x, f_x)$. We bound the drift $\mathrm{E}[X_t - X_{t+1} \mid X_t]$ by decomposing it into two components. Let $\Delta_t^a \coloneqq |y'| - |x|$, and let $\Delta_t^b \coloneqq |y| - |y'|$. Clearly we have $X_t - X_{t+1} = \Delta_t^a + \Delta_t^b$. For the (1,$\lambda$) EA, y' is an offspring of maximal fitness in the offspring population. It follows by Lemma 4 that

$$\mathrm{E}[\Delta_t^a] \le \frac{\ln\lambda}{\ln 2} - \frac{3|x|/2 - n}{n\ln 2} = \frac{\ln\lambda}{\ln 2} - \frac{n - 3X_t}{2n\ln 2}.$$

It remains to bound the contribution to the drift from the corruption coming from compromised nodes.

In the directional choice model, the adversary always reduces the number of one bits by the entire budget b. Thus, $\mathrm{E}[\Delta_t^b] = -\ell b/\lambda$ and the total drift is at most

$$\frac{\ln\lambda}{\ln 2} - \frac{n - 3X_t}{2n\ln 2} - \frac{\ell b}{\lambda}.$$

Hence, when $\ell b \ge \frac{\lambda\ln\lambda}{\ln 2}$ the drift between $0 < |X_t| < n/4$ is at most $-\frac{1}{8\ln 2}$.

For the uniform flip model, the expected number of bits corrupted depends on the intermediary string y'. In particular, we have

$$\mathrm{E}[\Delta_t^b] = \frac{\ell b}{\lambda}\left(1 - \frac{2|y'|}{n}\right).$$

Let $\mathcal{E}_t$ denote the event that $f_y = |y'| \ge |x|$. A sufficient condition for $\mathcal{E}_t$ is that x was cloned by at least one offspring, since the maximum fitness would be at least $|x|$. The probability that at least one offspring is a clone is at least

$$1 - \left(1 - \left(1 - \frac{1}{n}\right)^n\right)^\lambda \ge 1 - \left(\frac{3}{4}\right)^\lambda,$$

since $(1 - 1/n)^n \ge 1/4$ for all $n \ge 2$. Applying the law of total expectation we have, in the uniform flip model,

$$\begin{aligned} \mathrm{E}[\Delta_t^b] &= \mathrm{E}[\Delta_t^b | \mathcal{E}_t]\Pr(\mathcal{E}_t) + \mathrm{E}[\Delta_t^b | \overline{\mathcal{E}_t}](1 - \Pr(\mathcal{E}_t)) \\ &\le \mathrm{E}[\Delta_t^b | \mathcal{E}_t] + \mathrm{E}[\Delta_t^b | \overline{\mathcal{E}_t}](3/4)^\lambda \\ &\le \frac{\ell b}{\lambda}\left(1 - \frac{2|x|}{n}\right) + \frac{\ell n (3/4)^\lambda}{\lambda}. \end{aligned} \tag{3}$$

Using the bound $\ln(1+x) \ge x/(1+x)$ for $x > -1$, we have $4 \ge (\ln(4/3))^{-1}$. The restriction on λ implies $\lambda > 4\ln(4n/b) \ge \log_{4/3}(4n/b)$, and so $(3/4)^\lambda < \frac{b}{4n}$. This, together with the inequality in (3), along with the substitution $X_t = n - |x|$, yields an upper

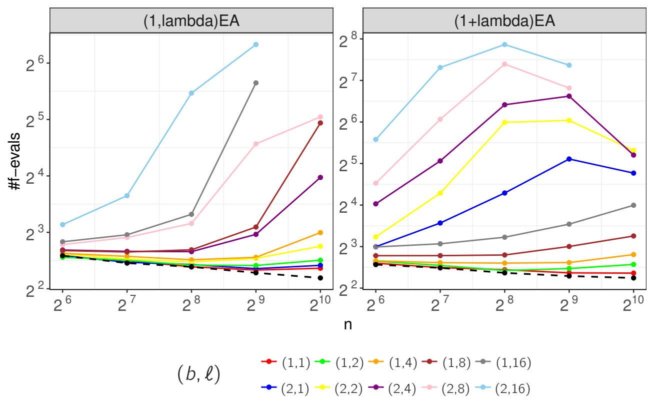


Figure 1: Mean number of fitness evaluations (divided by $n\log(n)$) for the (1,λ) EA and (1+λ) EA as a function of n for adversary budget b and count ℓ. Dashed black line reports the number of fitness evaluations for corruption free runs.

bound on the expected value of Δ_t^b.

$$E[\Delta_t^b] \le \frac{\ell b}{4\lambda} - \frac{\ell b}{\lambda}\left(1 - \frac{2X_t}{n}\right), \tag{4}$$

For $0 \le X_t \le n/4$, the total drift is thus at most

$$E[\Delta_t^a + \Delta_t^b] \le \frac{\ln\lambda}{\ln 2} - \frac{1}{8\ln 2} - \frac{\ell b}{4\lambda}.$$

When $\ell b > \frac{4\lambda\ln\lambda}{\ln 2}$, this is at most $-\frac{1}{8\ln 2}$.

Finally, we supply tail bounds on the absolute change in X_t. Note that $|\Delta_t^a + \Delta_t^b| \le |\Delta_t^a| + b$. Setting $\delta \coloneqq \sqrt{e} - 1$, we claim

$$\Pr(|X_t - X_{t+1}| \ge j) \le \Pr(|\Delta_t^a| \ge j - b) \le \frac{\sqrt{n}}{(1+\delta)^j}. \tag{5}$$

When $j \le b$, $\Pr(|\Delta_t^a| \ge j - b) = 1$ and the RHS of Equation (5) holds since $b \le \log n$. When $j > b$, the probability of flipping at least $j - b$ bits is at most $1/(1+\delta)^{j-b} = (\sqrt{e})^b/(1+\delta)^j \le \sqrt{n}/(1+\delta)^j$. Note that $\sqrt{n} = o(n/\log(n))$, so this satisfies the exponential decay conditions required by the Negative Drift Theorem [37, 38]. It therefore follows that in both the directional choice model and the uniform flip model, the first hitting time of $X_t = 0$, corresponding to an optimal solution, is $2^{\Omega(\sqrt{n})}$ with high probability. □

4 Adversary Budget vs Fitness

In this section, we empirically analyze the impact of adversaries' budgets and the expected fitness of output solutions. We evaluate the (1+λ) EA and the (1,λ) EA on OneMax for a fixed number of fitness function evaluations under different adversarial models. Experiments are run on instances of length $n \in \{64, 128, 512, 1024\}$ and λ is set to 64. We focus on three evaluation scenarios. First, we want to see how the runtime scales as a function of n for different numbers of adversaries and budgets. Second, we seek to evaluate how much the algorithms deviate from the optimal solution as a function of the budget b and the number of compromised nodes ℓ. Third, we empirically characterize the distribution of the runtime and the probability of solving OneMax, given the fixed maximum budget, in the presence of different adversaries. The reported statistics are computed over 100 independent runs. The stop condition is either finding the global optimum, which could be corrupted, or exhausting the fixed budget of 10^4 generations.

Figure 1 reports the mean number of fitness evaluations divided by $n\log n$ as a function of n. The mean is computed over successful runs (returning the all-ones string). For reference, we also plot the number of fitness evaluations in the absence of adversaries. We can see that both algorithms optimize OneMax in $n\log n$ when facing a small number of adversaries and budgets. However, as expected, the required computation effort to deal with a larger corruption budget scales very fast.

To observe the effect of the ratio ℓ/λ and the budget b on the best fitness found, we run the (1+λ) EA and the (1,λ) EA on OneMax in the uniform choice model for a variety of compromised nodes $\ell \in \{1, 2, 4, 8, 16\}$ and budget $b \in \{1, 2, 4, 8\}$. For each configuration, we plot the deviation from the optimal solution. The deviation relative to the optimal fitness is computed as the difference between the mean of 100 independent runs and the optimal fitness $f(x) = n$.

In Figure 2a, we plot the deviation from the optimal solution ($f(x) = n$) for the parallel (1,λ) EA and (1+λ) EA as a function of the number ℓ of compromised nodes and adversary budget b. From this figure, we can see that (1,λ) EA seems to cope well with adversarial corruption relative to (1+λ) EA, especially when adversaries use a small corruption budget. This would be explained by the comma selection mechanism that enables the optimization algorithm to *forget* about a corrupted solution, and so to be able to repair it and continue improving which can't be achieved with an elitist selection. We also see that the magnitude of deviation in fitness is much less for the (1,λ) EA compared to the (1+λ) EA.

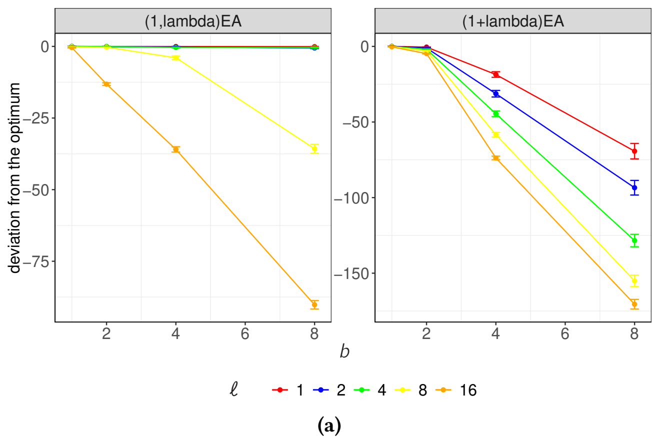
We also consider the empirical runtime of the (1+λ) EA and the (1,λ) EA under different budgets and number of compromised nodes. Notice that an algorithm can stop because it found a corrupted individual with fitness equal to the optimum, so it is important to plot the empirical cumulative distribution function (ECDF) for these algorithms to precisely characterize their runtime distribution. Figure 2b illustrates the ECDFs for both the (1,λ) EA and the (1+λ) EA for different combinations of the number of adversaries and budget. Overall, the ECDFs suggest that the (1,λ) EA has a better success probability compared to (1+λ) EA to solve OneMax and better runtime, up to some threshold of adversaries and corruption budget where it becomes unsuccessful (cf Theorem 5).

5 Results on Maximum Cuts in Graphs

In this section, we consider a compromised master-worker EA on the NP-hard combinatorial problem MaxCut. We first discuss bounds on how disruptive a corrupted offspring can be and provide empirical results on a number of benchmark graphs.

Let $G = (V, E)$ be an undirected graph. A *maximum cut* of G is a partition $(S, V \setminus S)$ such that the number of edges crossing from S to $V \setminus S$ is maximized. The MaxCut problem is the problem of finding a maximum cut.

In a graph with $|V| = n$ vertices, a partition can be represented as a binary string $x \in \{0, 1\}^n$ with $x_i = 1$ for $i \in [n]$ indicating if the i-th vertex belongs to S. The problem of solving MaxCut on G



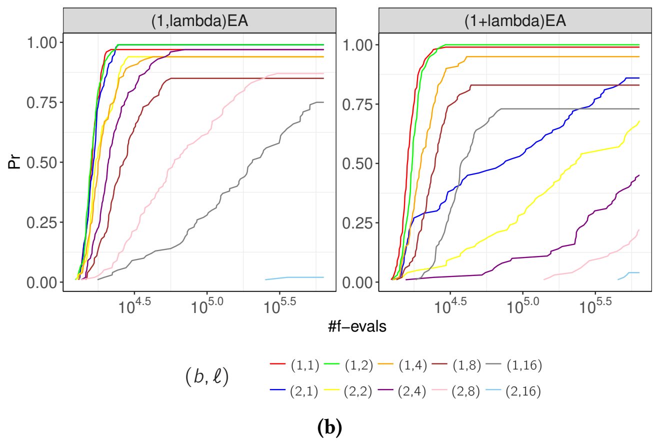


Figure 2: Impact of adversary budget b and number of adversaries ℓ on performance of the parallel (1,λ) EA and (1+λ) EA on OneMax for $n = 512, \lambda = 64$. (a) Deviation (mean and standard error) of the returned fitness from the optimal ($f(x) = n$) as a function of adversary budget and ℓ. (b) Empirical Cumulative Distribution Function for different adversary budget and ℓ.

reduces to the problem of maximizing the pseudo-Boolean function $f(x) = \sum_{(u,v)\in E}[x_u \neq x_v]$.

We consider the adversarial models described in Section 2.2. An adversary with budget b is able to change the membership of a vertex from S to $V \setminus S$ and vice versa.

Friedrich et al. [17] showed that it is always possible to improve the fitness of a candidate solution on MaxCut by a multiplicative factor of $(1+\epsilon/n^2)$ by making a single bit flip until a point $\hat{x}$ which is a $(1+\epsilon/n^2)$-approximation of a local maximum. One consequence of this is that the uniform choice or directional choice adversary with $b = 1$ can return an offspring (y, f_y) of such a point $\hat{x}$ where $f_y/f(y) \geq (1+\epsilon/n^2)$ with probability at least $1/n$.

We consider MaxCut instances generated by Helmberg and Rendl [24] from the instance library hosted at University of Bonn [34]. Instances G_2, G_3, G_4 are random graphs on $|V| = 800$ vertices and $|E| = 19176$ edges; instances G_{14}, G_{15}, G_{16} are the union of two almost maximal planar graphs with $|V| = 800$ and $4661 \leq |E| \leq 4694$.

To evaluate the impact of the uniform adversarial model on solutions fitness, we sampled solutions using a randomized local search using a single-bit flip search move. We ran RLS for 10^4 steps and retain the final solution returned on G_2 and G_{16}. For each returned solution, for different budgets $b \in \{1, \dots, 4\}$ we apply the uniform adversarial model on the returned solutions and record the fitness after perturbation. We repeat this 100 times for each budget.

In Figure 3 we plot the fitness of solutions found by the RLS *vs.* the fitness of solutions after corruption. The fitness after corruption for $b = 1$ is at best equal to the fitness of the corruption-free setting. While for a higher budget, all sampled solutions suffered a loss in fitness. For instance, for $b = 4$ the loss in fitness is up to 30 on G_{16} and goes up to around 40 on G_2. This figure suggests that for fitter solutions, adversarial corruption deteriorates the fitness of solutions. We also conducted a similar analysis for solutions sampled uniformly at random (not in the plot). For these solutions, their fitness is low and the effect of corruption is mixed. That is, for some solutions corruption was beneficial, and detrimental for others. It is important to note that we can not assume that adversarial corruption will always deteriorate the fitness of local optima solutions as the structure of their basin is not known a priori, and it is possible that some local optima may be on neutral patches in the landscape which may result in no effect for corruption.

Figure 4 reports the performance of the parallel $(1+\lambda)$ EA and (1,λ) EA on the considered instances. We plot the average final fitness and its standard deviation as a function of the number of adversaries (#adv) for different budgets. The mean is computed over 100 independent runs for each configuration. We also plot for reference the mean fitness and standard deviation of the corruption-free runs (gray horizontal lines). For the parallel (1+λ) EA, we can see that for budget $b \geq 2$, the (1+λ) EA suffers a loss in quality under any number of adversaries. This loss is around 5% when there are few adversaries, and increases to around 10% when there are 16 adversaries out of 64 workers. When adversaries use a small budget, $b = 1$, the deviation from the corruption-free performance is relatively small even for higher l. For the parallel $(1, \lambda)$ EA, we can observe the magnitude of deviation from the adversary-free scenario is about half as small compared to the (1+λ) EA: around 5% for 16 adversaries with $b = 4$, and around 10% under the same configuration for the (1+λ) EA. This suggests that an EA using a non-elitist selection can be more resilient to adversaries compared to elitist selection. The surprising result is that the non-elitist (1,λ) EA seems to leverage a small amount of adversarial corruption to even increase its performance above the corruption-free process. For instance, for $b = 1$ and a number of adversaries less than 8, the mean final fitness of (1,λ) EA is higher than the fitness of the non-adversarial setting. To statistically compare this improvement in performance of comma selection over a corruption-free scenario, we performed a one-sided Wilcoxon–Mann–Whitney U-test with a 95% confidence level. The results of these tests are reported in Table 1. For every MaxCut instance, when the adversary budget and count is comparatively small, we can observe a statistically significant effect. We posit that the (1,λ) EA leverages the extra variation arising from the corrupted strings. Understanding this effect better is an avenue of future work.

G_2		ℓ				
		1	2	4	8	16
	1	★	★	★	★	
b	2	★	★	★		
	4	★	★	★		

G_3		ℓ				
		1	2	4	8	16
	1	★	★	★	★	
b	2	★	★	★		
	4	★	★	★		

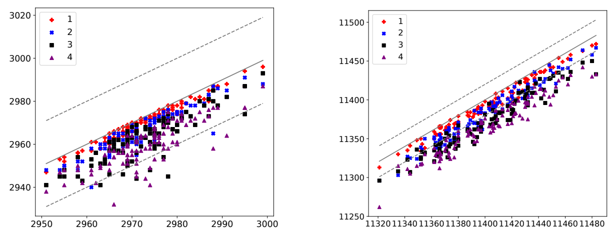
G_4		ℓ				
		1	2	4	8	16
	1	★	★	★	★	
b	2	★	★	★		
	4	★	★	★		

G_{14}		ℓ				
		1	2	4	8	16
	1	★	★	★	★	
b	2	★	★	★		
	4	★	★			

G_{15}		ℓ				
		1	2	4	8	16
	1	★	★	★	★	
b	2	★	★	★		
	4	★	★			

G_{16}		ℓ				
		1	2	4	8	16
	1	★	★	★	★	
b	2	★	★	★		
	4	★	★			

Table 1: Statistical significance results of one-sided Wilcoxon–Mann–Whitney U-tests that the (1,λ) EA obtains a higher final fitness than the corruption-free setting on each MaxCut instance. A star ($\star$) in row i and column j denotes $p < 0.05$ for budget $b = i$ and $\ell = j$ on the corresponding instance.



(a) **(b)**

Figure 3: Fitness of solutions on MaxCut instances G_{16} (a) and G_2 (b) returned by RLS (x-axis) *vs.* their fitness after corruption (y-axis) varying corruption strengths. Dashed lines are $y = x \pm 20$.

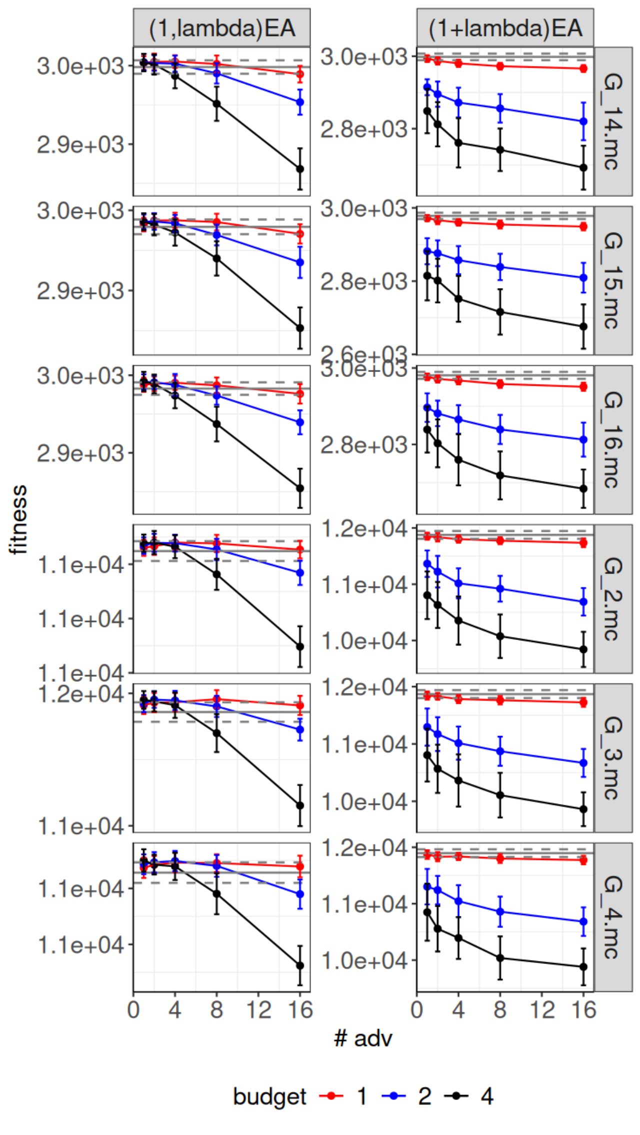


Figure 4: Final fitness vs adversary count ℓ (#adv) and budget b for (1+λ) EA and (1,λ) EA on MaxCut instances $G_2, G_3, G_4, G_{14}, G_{15}, G_{16}$. $\lambda = 64$. Horizontal line corresponds to average and standard deviation of performance in adversary-free setting.

6 Conclusion

In this paper, we analyzed distributed evolutionary algorithms for pseudo-Boolean optimization in the presence of adversarial corruption. We analyzed elitist and non-elitist distributed evolutionary algorithms when communicated solutions between computing nodes and a central machine are corrupt. Under different corruption models, we showed how elitist plus and comma selection strategies affect the optimization performance and solution quality on the OneMax and MaxCut problems.

While our analysis showed that comma selection can be beneficial to some extent to limit the impact of adversaries, it is important to conceive mechanisms that will enable distributed EAs to better cope with adversarial corruption, especially when deployed for critical optimization problems. Moreover, understanding the influence of algorithmic parameters such as mutation strength and population size on the robustness of distributed EAs in the presence of adversaries is crucial to better determine the extent to which we can expect them to be reliable. Also, while our analysis focused on the master/worker parallelization technique, applying adversarial models to other distributed architectures such as island models would further improve our understanding of the strengths and limitations of distributed evolutionary optimization algorithms.

Acknowledgments

This work was supported by NSF grant 2144080.

References

[1] Brahim Aboutaib and Andrew M Sutton. 2022. Runtime analysis of unbalanced block-parallel evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature*. Springer, 555–568.

[2] Brahim Aboutaib and Andrew M. Sutton. 2024. The Influence of Noise on Multi-parent Crossover for an Island Model Genetic Algorithm. *ACM Trans. Evol. Learn. Optim.* 4, 2 (2024), 11. https://doi.org/10.1145/3630638

[3] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow. 2013. Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research* 20, 1 (2013), 1–48. https://doi.org/10.1111/j.1475-3995.2012.00862.x

[4] Enrique Alba and Marco Tomassini. 2002. Parallelism and Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 6, 5 (2002), 443–462.

[5] Enrique Alba and José M. Troya. 1999. A survey of parallel distributed genetic algorithms. *Complex.* 4, 4 (1999), 31–52. https://doi.org/10.1002/(SICI)1099-0526(199903/04)4:4%3C31::AID-CPLX5%3E3.0.CO;2-4

[6] Denis Antipov, Benjamin Doerr, and Alexandra Ivanova. 2024. Already moderate population sizes provably yield strong robustness to noise. In *Proceedings of the Genetic and Evolutionary Computation Conference.* 1524–1532.

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

[8] Albert D. Bethke. 1976. *Comparison of Genetic Algorithms and Gradient-Based Optimizers on Parallel Processors: Efficiency of Use of Processing Capacity.* Technical Report 197. University of Michigan.

[9] Erick Cantú-Paz. 1998. A survey of parallel genetic algorithms. *Calculateurs Parallèles Reseaux et Systems Repartis* 10, 2 (1998), 141–171.

[10] Erick Cantú-Paz. 2001. Master-Slave Parallel Genetic Algorithms. In *Efficient and Accurate Parallel Genetic Algorithms.* Springer, New York, 33–48.

[11] Wei-Neng Chen, Feng-Feng Wei, Tian-Fang Zhao, Kay Chen Tan, and Jun Zhang. 2023. A Survey on Distributed Evolutionary Computation.

[12] Matthieu Dinot, Benjamin Doerr, Ulysse Hennebelle, and Sebastian Will. 2023. Runtime Analyses of Multi-Objective Evolutionary Algorithms in the Presence of Noise. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China.* ijcai.org, 5549–5557. https://doi.org/10.24963/IJCAI.2023/616

[13] Benjamin Doerr, Philipp Fischbeck, Clemens Frahnow, Tobias Friedrich, Timo Kötzing, and Martin Schirneck. 2019. Island Models Meet Rumor Spreading. *Algorithmica* 81, 2 (2019), 886–915. https://doi.org/10.1007/S00453-018-0445-2

[14] Benjamin Doerr and Andrew M. Sutton. 2019. When resampling to cope with noise, use median, not mean. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, Anne Auger and Thomas Stützle (Eds.). ACM, 242–248. https://doi.org/10.1145/3321707.3321837

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

[16] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 3518–3527. http://proceedings.mlr.press/v80/mhamdi18a.html

[17] Tobias Friedrich, Francesco Quinzan, and Markus Wagner. 2018. Escaping large deceptive basins of attraction with heavy-tailed mutation operators. In *Proceedings of the Genetic and Evolutionary Computation Conference.* 293–300.

[18] Yong-Feng Ge, Jinli Cao, Hua Wang, Yanchun Zhang, and Zhenxiang Chen. 2020. Distributed Differential Evolution for Anonymity-Driven Vertical Fragmentation in Outsourced Data Storage. In *Web Information Systems Engineering – WISE 2020*, Zhisheng Huang, Wouter Beek, Hua Wang, Rui Zhou, and Yanchun Zhang (Eds.). Springer International Publishing, Cham, 213–226.

[19] Christian Gießen and Timo Kötzing. 2016. Robustness of Populations in Stochastic Environments. *Algorithmica* 75, 3 (2016), 462–489.

[20] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. 2015. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing* 34 (2015), 286–300. https://doi.org/10.1016/j.asoc.2015.04.061

[21] John J. Grefenstette. 1981. *Parallel adaptive algorithms for function optimization.* Technical Report CS-81-19. Vanderbilt University, Nashville, TN.

[22] Tomohiro Harada and Enrique Alba. 2020. Parallel Genetic Algorithms: A Useful Survey. *ACM Comput. Surv.* 53, 4 (2020), 86:1–86:39. https://doi.org/10.1145/3400031

[23] K. Haritha, S. Shailesh, M. V. Judy, K. S. Ravichandran, Raghunathan Krishankumar, and Amir H. Gandomi. 2023. A novel neural network model with distributed evolutionary approach for big data classification. *Scientific Reports* 13, 1 (Jul 2023), 11052. https://doi.org/10.1038/s41598-023-37540-z

[24] Christoph Helmberg, Franz Rendl, and A. Oh. 2000. A Spectral Bundle Method for Semidefinite Programming. *SIAM Journal on Optimization* 10, 3 (2000), 673–696. https://doi.org/10.1137/S1052623497328987 arXiv:https://doi.org/10.1137/S1052623497328987

[25] Moslema Jahan, M. M. A Hashem, and Gazi Abdullah Shahriar. 2011. Distributed Evolutionary Computation: A New Technique for Solving Large Number of Equations. *International Journal of Distributed and Parallel System* 2, 6 (2011), 31–49. https://doi.org/10.5121/ijdps.2011.2604

[26] Thomas Jansen, Kenneth A De Jong, and Ingo Wegener. 2005. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation* 13, 4 (2005), 413–440.

[27] Jörg Lässig and Dirk Sudholt. 2013. Design and analysis of migration in parallel evolutionary algorithms. *Soft Comput.* 17, 7 (2013), 1121–1144. https://doi.org/10.1007/S00500-013-0991-0

[28] Jörg Lässig and Dirk Sudholt. 2014. Analysis of speedups in parallel evolutionary algorithms and (1+λ) EAs for combinatorial optimization. *Theoretical Computer Science* 551 (2014), 66–83. https://doi.org/10.1016/j.tcs.2014.06.037

[29] Jörg Lässig and Dirk Sudholt. 2014. General Upper Bounds on the Runtime of Parallel Evolutionary Algorithms. *Evol. Comput.* 22, 3 (2014), 405–437. https://doi.org/10.1162/EVCO_A_00114

[30] Per Kristian Lehre and Xiaoyu Qin. 2023. Self-adaptation can improve the noise-tolerance of evolutionary algorithms. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms.* 105–116.

[31] Per Kristian Lehre and Dirk Sudholt. 2020. Parallel Black-Box Complexity With Tail Bounds. *IEEE Trans. Evol. Comput.* 24, 6 (2020), 1010–1024. https://doi.org/10.1109/TEVC.2019.2954234

[32] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. 2014. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on operating systems design and implementation (OSDI 14).* 583–598.

[33] Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. 2014. Communication efficient distributed machine learning with the parameter server. *Advances in Neural Information Processing Systems* 27 (2014).

[34] Sven Mallach. 2025. MaxCut and BQP Instance Library. http://bqp.cs.uni-bonn.de/library/html/instances.html. Online; accessed January 2025.

[35] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On Detecting Adversarial Perturbations. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.* OpenReview.net. https://openreview.net/forum?id=SJzCSf9xg

[36] Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms.* Cambridge University Press.

[37] Pietro S. Oliveto and Carsten Witt. 2011. Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation. *Algorithmica* 59, 3 (2011), 369–386. https://doi.org/10.1007/S00453-010-9387-Z

[38] Pietro S. Oliveto and Carsten Witt. 2012. Erratum: Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation. *CoRR* abs/1211.7184 (2012). arXiv:1211.7184 http://arxiv.org/abs/1211.7184

[39] Xinyu Pang, Yong-Feng Ge, Kate Wang, Agma J M Traina, and Hua Wang. 2023. Patient assignment optimization in cloud healthcare systems: a distributed genetic algorithm. *Health Inf Sci Syst* 11, 1 (June 2023), 30.

[40] Chao Qian, Yang Yu, and Zhi-Hua Zhou. 2018. Analyzing Evolutionary Optimization in Noisy Environments. *Evolutionary Computation* 26, 1 (03 2018), 1–41. https://doi.org/10.1162/evco_a_00170 arXiv:https://direct.mit.edu/evco/article-pdf/26/1/1/1548288/evco_a_00170.pdf

[41] Dirk Sudholt. 2015. *Parallel Evolutionary Algorithms.* Springer Berlin Heidelberg, Berlin, Heidelberg, 929–959. https://doi.org/10.1007/978-3-662-43505-2_46

[42] Dirk Sudholt. 2021. Analysing the Robustness of Evolutionary Algorithms to Noise: Refined Runtime Bounds and an Example Where Noise is Beneficial. *Algorithmica* 83, 4 (2021), 976–1011. https://doi.org/10.1007/S00453-020-00671-0

[43] Shreyas Sundaram and Bahman Gharesifard. 2018. Distributed optimization under adversarial nodes. *IEEE Trans. Automat. Control* 64, 3 (2018), 1063–1076.

[44] El-Ghazali Talbi. 2015. *Parallel Evolutionary Combinatorial Optimization.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1107–1125. https://doi.org/10.1007/978-3-662-43505-2_55

[45] Darrell Whitley, Ozeas Quevedo De Carvalho, Mark Roberts, Vivint Shetty, and Piyabutra Jampathom. 2023. Scheduling multi-resource satellites using genetic algorithms and permutation based representations. In *Proceedings of the Genetic and Evolutionary Computation Conference.* 1473–1481.

[46] Yusen Wu, Hao Chen, Xin Wang, Chao Liu, Phuong Nguyen, and Yelena Yesha. 2021. Tolerating Adversarial Attacks and Byzantine Faults in Distributed Machine Learning. In *2021 IEEE International Conference on Big Data (Big Data).* 3380–3389. https://doi.org/10.1109/BigData52589.2021.9671583

[47] Chengcheng Zhao, Jianping He, and Qing-Guo Wang. 2020. Resilient Distributed Optimization Algorithm Against Adversarial Attacks. *IEEE Trans. Automat. Control* 65, 10 (2020), 4308–4315. https://doi.org/10.1109/TAC.2019.2954363