ARTICLE IN PRESS

Computers and Mathematics with Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters

Jakub Muszyński [a], Sébastien Varrette [a,*], Pascal Bouvry [a], Franciszek Seredyński [b,c], Samee U. Khan [d]

[a] University of Luxembourg, Computer Science and Communications (CSC) Research Unit, 6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg

[b] Cardinal Stefan Wyszynski University, Department of Mathematics and Natural Sciences, Woycickiego 1/3, 01-938 Warsaw, Poland

[c] Polish-Japanese Institute of Information Technology, Department of Computer Science, Koszykowa 86, 02-008 Warsaw, Poland

[d] North Dakota State University (NDSU), Fargo, ND 58108–6050, USA

ARTICLE INFO

Keywords:
Evolutionary algorithms
Algorithmic based fault tolerance
Desktop grids and volunteer computing systems
Convergence proof

ABSTRACT

This paper analyzes the fault-tolerance nature of Evolutionary Algorithms (EAs) when executed in a distributed environment subjected to malicious acts. More precisely, the inherent resilience of EAs against two types of failures is considered: (1) *crash faults*, typically due to resource volatility which lead to data loss and part of the computation loss; (2) *cheating faults*, a far more complex kind of fault that can be modeled as the alteration of output values produced by some or all tasks of the program being executed. This last type of failure is due to the presence of *cheaters* on the computing platform. Most often in Global Computing (GC) systems such as BOINC, cheaters are attracted by the various incentives provided to stimulate the volunteers to share their computing resources: cheaters typically seek to obtain rewards with little or no contribution to the system. In this paper, the Algorithm-Based Fault Tolerance (ABFT) aspects of EAs against the above types of faults is characterized. Whereas the inherent resilience of EAs has been previously observed in the literature, for the first time, a formal analysis of the impact of the considered faults over the executed EA including a proof of convergence is proposed in this article.

By the variety of problems addressed by EAs, this study will hopefully promote their usage in the future developments around distributed computing platform such as Desktop Grids and Volunteer Computing Systems or Cloud systems where the resources cannot be fully trusted.

1. Introduction

Since their advent in the late 1990s, Desktop Grids and Volunteer Computing Systems (DGVCSs) also known as Global Computing (GC) platforms have been the largest and most powerful distributed computing systems in the world, offering an abundance of computing power at a fraction of the cost of dedicated supercomputers. Such systems, whose topology is illustrated in Fig. 1 are based on volunteer computing: idle cycles of desktop PCs and workstations voluntarily shared by the users worldwide are stolen through the Internet to compute parts of a huge problem. One of the best known projects that exploits this kind of platform is SETI@Home [1] which has now moved to BOINC [2]. The latest statistics of BOINC[1] are outstanding: 2.4 millions of users, 6.7 millions of machines and an average computing speed of 5.3 PetaFLOPS.

* Corresponding author. Tel.: +352 46 66 44 6600.
E-mail addresses: Jakub.Muszynski@uni.lu (J. Muszyński), Sebastien.Varrette@uni.lu (S. Varrette), Pascal.Bouvry@uni.lu (P. Bouvry), fseredynski@gmail.com (F. Seredyński), samee.khan@ndsu.edu (S.U. Khan).

[1] See BOINC stats, http://boincstats/.

0898-1221/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2012.03.004

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

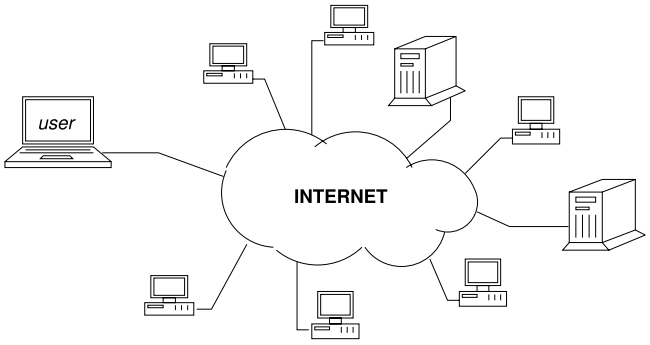

Fig. 1. A typical Global Computing platform.

Many applications from a wide range of scientific domains – including computational biology, climate prediction, particle physics and astronomy – have used the computing power offered by such systems. In general, the scheduling is handled in a master/slave model: a server distributes tasks to the participating clients or *workers*; they subsequently process their input data and send the computed results back to the server.

One important aspect of desktop grid is the heterogeneity and the extreme volatility of the resources as the owners of the computing resources may reclaim them without warning, leading therefore to what is commonly named a *crash fault*. In addition, the motivation for the users to contribute are manifold, including the altruistic desire to help or, more importantly, the assignment of credit points proportionally to a user's contribution. These points allow to reward hard-working clients in different ways [3]. Unfortunately, incentives also attract cheaters who seek to obtain these rewards with little or no contribution to the system. This is commonly referred as *cheating faults*. Such selfish behavior can be achieved by modifying the client software as experienced in SETI@Home [4]. The presence of cheaters in grid computing system is well-known and many countermeasures have been proposed in the literature [5–9], yet at the price of a relatively huge overhead as the only generic approach to detect wrong results created by lazy participants or malicious cheaters relies on duplication, either total or partial. Recently, efficient and generic result checking approaches based on partial task duplication and macro dataflow analysis have been proposed to tackle the issue of massive attacks in which the number of corrupted results exceed a given threshold [9]. Modifications (*i.e.* cheatings) up to the threshold are assumed to be handled by the application itself, typically by specific low-overhead system-level approaches, such as Algorithm-Based Fault Tolerance (ABFT) techniques where the fault tolerance scheme is tailored to the algorithm performed.

This paper studies the ABFT nature of Evolutionary Algorithms (EAs) by evaluating the impact of crash faults and cheating on such processes when executed on a distributed and potentially hostile environment such as a GC platform. Evolutionary Algorithms (EAs) are a class of stochastic search techniques which have been successfully applied to solve a large variety of hard optimization problems which typically require several days or months of computation. Decreasing the makespan of the execution is typically obtained by relying on a parallel version executed on a distributed platform. Yet, as mentioned above, such large scale systems are prone to errors. In this paper, we formally analyze in which conditions a population-based Evolutionary Algorithm (EA) is expected to converge (or not) towards valid solutions despite the presence of faults. This is a definitive contribution to the domain as many previous work [10–14] suggested through experiments the inherent resilience of EAs against the faults considered in this work. A tentative sketch of proof was initiated in [15], based on the convergence results found in the context of Interacting Particle System (IPS) [16] used to model Genetic Algorithm (GA). Yet the proof suffers from various flaws (see Section 2.3) that do not permit to conclude rigorously on the convergence. Here, we address the convergence problem from another perspective directly inspired by the results found in the seminal paper of Rudolph [17]. In particular, whereas the initial study concludes on the convergence in all circumstances, we demonstrate that there are some cases where the EA *will not* converge.

The convergence analysis conducted in this paper is done through a step-wise approach: we first concentrate on the most simple variant of an EA that is still of theoretical and practical interest: the $(1 + 1)$ EA where the size of the population is restricted to one individual and where the cross-over step is not used. We then extend our results to regular EAs to characterize their ABFT nature against both crash and cheating faults.

Note that the theoretical proofs presented in this paper eventually conclude on the convergence towards valid solutions despite the presence of faults, yet nothing permits to state that it will be done in a *reasonable* running time. More precisely, the overhead induced by the failure is not covered by our work. The contributions of this paper will help in promoting the usage of EA approaches in future developments around distributed computing platform such as DGVCS or Cloud systems where the resources cannot be fully trusted.

This paper is organized as follows: Section 2 details the background of this work and reviews related works. Section 3 offers an overview of the convergence analysis of EAs in a fault-free environment. Section 4 holds the main contribution of this paper as it details the convergence analysis of EAs in the presence of crash faults and, most importantly, cheating faults. The impact of these results on distributed computations is discussed in the Section 5 while Section 6 concludes the paper with a summary of our results and future directions.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

2. Background and related work

2.1. Evolutionary Algorithms (EAs)

Evolutionary Algorithm (EA) is a class of solving techniques based on the Darwinian theory of evolution [18] which involves the search of a *population* X_t of solutions. Members of the population are feasible solutions and called *individuals*. Each iteration of an EA involves a competitive selection that weeds out poor solutions through the *evaluation* of a fitness value that indicates the quality of the individual as a solution to the problem. The evolutionary process involves at each generation a set of stochastic operators that are applied on the individuals, typically recombination (or cross-over) and mutation. There exists many useful models of EAs yet a pseudo-code of a general execution scheme is provided in Algorithm 1.

Algorithm 1: General scheme of an EA in pseudo-code.

```
t := 0;
Generation(X_t); // generate the initial population
Evaluation(X_t); // evaluate population
while Stopping criteria not satisfied do
    X̂_t := ParentsSelection(X_t); // select parents
    X'_t := Modification(X̂_t); // cross-over + mutation
    Evaluation(X'_t); // evaluate offspring
    X_{t+1} := Selection(X_t, X'_t); // select survivors for the next generation
    t := t + 1;
```

Execution of simple EA requires high computational resources in case of non-trivial problems. It might be encountered when dealing with large individuals (*e.g.* in case of GA– long sequences of genes, in case of Genetic Programming (GP)– large parse trees) and/or large populations. This influences time required to evaluate the population, which usually is the costliest operation in EAs. In such cases, time-to-solution on a single computer is prohibitively long for practitioners (especially with usage of GP). Such an example of highly expensive EA for a computer vision problem is described in [19], where more than 24 h is required to execute the algorithm. Another instance of even bigger requirements was reported by Melab et al. in [20], where the predictive mathematical model for the concentration of sugar in beets was constructed using parallel GA, where the cumulative CPU time exceeded 27 days.

2.2. Fault tolerance and robustness in distributed systems

Fault tolerance can be defined as the ability of a system to behave in a well-defined manner once a failure occurs. A failure is due to an error of the system which is a consequence of a fault. Different kinds of faults are usually distinguished in function of their origin and their temporal duration [21]. They could be intentional or not, software or hardware—modify the processing time of an operation, provide a wrong result or return no result at all.

Failures and errors can be classified into the following categories based on their semantics:

- *crash-stop failures*, also called fail-stop: the system stops working and does not execute any operation, nor does it send any signals. This behavior corresponds to what the literature often refers to as *crash faults* which are considered in this article;
- *omission* is a communication failure: typically, a message is not transmitted by a communication channel, or not sent by the sending process (send-omission), or not received by the receiving process (receive-omission);
- *duplication* is the opposite from omission: a message is sent or received twice;
- *timing* if the system's behavior deviation concerns only a time criterion (reaction time to a given event for instance);
- *byzantine errors* are arbitrary errors: the system arbitrarily does not have the expected behavior or has an erroneous one.

The detection of failures is clearly outside the scope of this article (the interested reader may refer to [22]). In all cases, Byzantine errors are the hardest to detect, as the behavior of the system is often similar to the expected one. A typical example of byzantine failure covered in this work comes from volunteer computing where it is called *cheating faults*. Indeed, the motivation for the users to contribute to desktop grids are manifold, including the altruistic desire to help or, more importantly, the assignment of credit points proportionally to their contribution. These points reward hard-working clients in different ways [3]. Unfortunately, incentives also attract cheaters who seek to obtain these rewards with little or no contribution to the system. Such selfish behavior can be achieved by modifying the client software as experienced in Seti@Home[2] [4]. Whether these modifications were the consequences of malicious acts or not, this example illustrates a

[2] Some volunteers modified the client executed on local machines to "better" compute the fast Fourier transform (FFT) in order to send results faster. Yet the modification made to the software had so many glitches that incorrect results were sent back from client using the altered version of the software. This leads to the loss of several months of computations.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

scenario where the results of a computation conducted by a remote client are corrupted by cheaters (hence the reference to cheating faults). What makes this kind of fault byzantine and difficult to catch is that the data sent to the server, although erroneous, generally respect the expected format and do not raise alarm regarding their integrity.

2.2.1. System robustness

When a given system is resilient to a given type of fault, one generally claims that this system is *robust*. Yet defining rigorously robustness is not an easy task and many contributions come with their own interpretation of what robustness is. Actually, there exists a systematic framework that permits to define a robust system unambiguously. In fact, this should be probably applied to any system or approach claiming to propose fault-tolerance mechanism. This framework, formalized in [23], answers the following three questions:

1. What behavior of the system makes it robust?
2. What uncertainties is the system robust against?
3. Quantitatively, exactly how robust is the system?

The first question is generally linked to the technique or the algorithm applied. The second question explicitly lists the type of faults or disturbing elements targeted by the system. Answering this question is critical to delimit the application range of the designed system and to avoid counter examples selected in a context not addressed by the robust mechanism. The third and the last question is probably the most difficult to answer, and at the same time the most vital to characterize the limits of the system. Indeed, there is nearly always a threshold on the error/fault rate above which the proposed infrastructure fails to remain robust and breaks (in some sense). This framework will be applied for the two type of faults considered here, namely crash and cheating faults.

2.2.2. Algorithm-Based Fault Tolerance (ABFT)

Algorithm-Based Fault Tolerance (ABFT) [24] is an error detection technique currently applied to various matrix operations like matrix multiplication, LU decomposition or matrix inversion. The fault tolerance scheme is tailored to the algorithm performed to make it resilient to a limited number of byzantine failures producing falsified results. ABFT can be tuned to provide the desired fault tolerance (e.g. single error detection, single error correction, etc.) and this class of approaches usually achieve a very low overhead. For instance, in matrix computations, ABFT consists in encoding a "checksum matrix" by adding checksum rows or columns from the original matrix as discussed in [24]. The ABFT concept was initially designed to achieve high reliability on a high performance computing platforms made of parallel processors with shared memory. This approach has been since extended to large scale computational grids with dynamic resources. For instance in [25,26], a checkpoint-free fault tolerance mechanism is provided for parallel matrix computation over volatile resources assuming a fail-stop failure (or crash-fault) model. In this paper, we perform the analysis of the ABFT nature of EAs, not only against crash faults but also against the more complex failure class presented above: cheating faults.

2.3. Fault-tolerant aspects of EAs

A set of recent studies [10–12,14] illustrate what seems to be a natural resilience of evolutionary algorithms against a model of destructive faults (often referred to as *crash faults*). In these studies, experiments were conducted on sequential GA and GP heuristics (for instance through the Ant problem in [10]). It has also been experimented by the NASA on hardware components in [27]. Furthermore, these empirical studies mainly focused on sequential GA have been extended in [11] on coarse-grain distributed Evolutionary Algorithms (dEAs) with island model, yet on GA benchmarks such as the analysis of the Fmodal or the Schwefel's function.

This tendency has been confirmed on a more complex kind of faults that appear on Global Computing platforms, namely *cheating faults* [13]. This places EAs as a potential ABFT technique against such faults. To the best of our knowledge, there exists a single study that attempts to provide a theoretical analysis of the impact of these faults over EAs. More precisely, a tentative sketch of proof of the above property, *i.e.* the ABFT nature of EAs, was initiated in [15], based on the convergence results found in the context of Interacting Particle System (IPS) [16] used to model GA. Yet this study suffers from various flaws that do not permit to conclude rigorously on the convergence. First of all, Markovian kernels defined in the paper do not fulfill properties of this construct (described in the next section). Additionally, the selection schema used in the model and incorporated in the paper does not guarantee survival of the best individual at each generation. There is also no other mechanism preserving this property. Thus the sequence of a "population-best-fitness" is *not* monotonic – it will wander through every possible value (however, some values may be much more likely to occur than others) – yet the monotonicity assumption is required to apply the convergence results found in the literature [16,17].

In this article, we address the convergence problem in the described context from another perspective, directly inspired by the results found in the seminal article of Rudolph [17].

2.4. Convergence analysis of EA

To analyze the ability of EA to converge in some sense to "*good*" solutions, several notions need to be introduced. This is the purpose of the following sections.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

2.4.1. Markov chain and Markovian kernels

Let E be a set of a measurable space $(E, \mathcal{A})$ and T be an index set identical with $\mathbb{N}_0$. Note that this choice over $\mathbb{N}_0$ is done for convenience. Then a family of random variables $(X_t : t \in T)$ on a joint probability space $(\Omega, \mathcal{F}, \mathrm{P})$ is called a *stochastic process with discrete time*. Additionally the set E is called the *state space* of the process and set T is interpreted as points in time.

If for $0 < t_1 < t_2 < \cdots < t_k < t$ with some $k \in \mathbb{N}$ and $A \in \mathcal{A}$

$$\mathrm{P}\left\{X_t \in A \mid X_{t_1}, X_{t_2}, \ldots, X_{t_k}\right\} = \mathrm{P}\left\{X_t \in A \mid X_{t_k}\right\} \tag{1}$$

then $(X_t : t \geq 0)$ is called a *Markov chain*. If additionally

$$\forall_{s,t,k \in \mathbb{N}_0 \wedge s \leq t} \mathrm{P}\{X_{t+k} \in A \mid X_{s+k}\} = \mathrm{P}\{X_t \in A \mid X_s\} \tag{2}$$

then the Markov chain is termed *homogeneous*, otherwise *inhomogeneous*.

For a homogeneous Markov chain $(X_t : t \geq 0)$ on a probability space $(\Omega, \mathcal{F}, \mathrm{P})$ with image space $(E, \mathcal{A})$, map $K : E \times \mathcal{A} \to [0, 1]$ is termed a *Markovian kernel* or a *transition probability function* if

- $\forall_{A \in \mathcal{A}} K(., A)$ is measurable,
- $\forall_{x \in E} K(x, .)$ is a probability measure on $(E, \mathcal{A})$.

In particular $\forall_{x \in E} \forall_{A \in \mathcal{A}} \forall_{t \in \mathbb{N}_0} K(x, A) = \mathrm{P}\{X_{t+1} \in A \mid X_t = x\}$.

Let $x \in E$ be a starting state and $A \subseteq E$ be a target set, then probability to transition from state x to set A within t steps is given by t-th iteration of the Markovian kernel:

$$K^{(t)}(x, A) = \begin{cases} K(x, A), & t = 1 \\ \displaystyle\int_E K^{(t-1)}(y, A) K(x, \mathrm{d}y), & t > 1 \end{cases} \tag{3}$$

where integration is operated with respect to an appropriate measure on $(E, \mathcal{A})$.

Finally, among the various properties of Markov kernel, the following ones will be used throughout the article: for all $x \in E$ and $A \subseteq E$:

$$\int_E K(x, \mathrm{d}y) \cdot \mathbb{1}_A(y) = \int_A K(x, \mathrm{d}y) = K(x, A) \tag{4}$$

$$K(x, A) + K(x, A^c) = 1. \tag{5}$$

2.4.2. Markov model of EA and convergence conditions

Markov chain have been used quite earlier in the literature [28] as a basis to establish the convergence in a broad sense of EAs. In this context, an EA is modeled as follows: the algorithm is executed using a population of N individuals represented by the N-tuple $(x_1, \ldots, x_N)$ with $x_i \in \mathcal{M}$ for $i = 1, \ldots, N$. $\mathcal{M}$ is a search space of a real-valued objective/fitness function $f : \mathcal{M} \to \mathbb{R}$ bounded from below, *i.e.* $\forall_{x \in \mathcal{M}} f(x) > -\infty$, and being minimized by the algorithm. This gives a state space $E = \mathcal{M}^N$. The population at step $t = 0$ (called *initial population*) is created using some initial distribution $p(.)$ yielding the random population $X_0 = (X_{0,1}, \ldots, X_{0,N})$. A population $X_t = (X_{t,1}, \ldots, X_{t,N})$ at time $t > 0$ is generated by so-called genetic operators (described by the associated stochastic kernels $K(., .)$). These operators permit to evolve the population and they depend only on a population from a previous step $(t - 1)$.

The above model leads to a conclusion that the stochastic sequence $(X_t : t \geq 0)$ is a Markov chain.

To analyze the ability of EA to converge in some sense to a specific set associated with globally optimal solutions of the optimization problem, the term *convergence* has to be precisely defined.

Definition 2.1 (*Complete Convergence [17]*)**.** Let (D_t) be a sequence of random variables defined on a probability space $(\Omega, \mathcal{F}, P)$. Then (D_t) is said to *converge completely* to 0, denoted as $D_t \xrightarrow{0} 0$, if for any $\epsilon > 0$

$$\lim_{t \to \infty} \sum_{i=1}^{t} P\{|D_i| > \epsilon\} < \infty. \tag{6}$$

Definition 2.2 (*Convergence in Probability [17]*)**.** Let (D_t) be a sequence of random variables defined on a probability space $(\Omega, \mathcal{F}, P)$. Then (D_t) is said to *converge in probability* to 0, denoted as $D_t \xrightarrow{\mathrm{P}} 0$, if for any $\epsilon > 0$

$$\lim_{t \to \infty} P\{|D_t| > \epsilon\} = 0. \tag{7}$$

It is worth noting here that complete convergence implies convergence in probability. Now we define additional notations that will be used in the sequel:

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

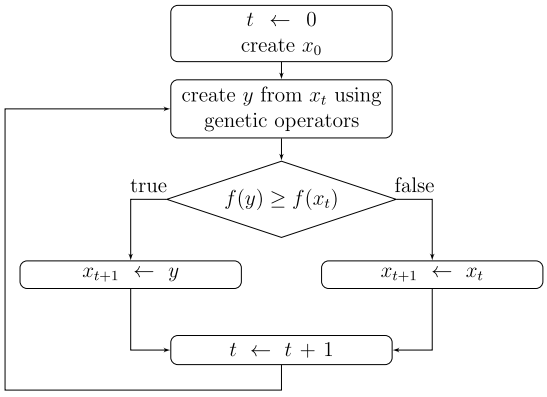

Fig. 2. Execution scheme of a $(1+1)$ EA with elitist selection.

- $b(X_t) = \min\{f(X_{t,i}) : i = 1, \ldots, N\}$—the best objective/fitness function value of population X_t at step $t \geq 0$;
- f^*—global minimum of the objective/fitness function $f : \mathcal{M} \to \mathbb{R}$;
- $d(X_t) = b(X_t) - f^*$—distance of the best objective/fitness function value of population X_t at step $t \geq 0$ to the global optimum f^*;
- $A_\epsilon = \{x \in E : d(x) < \epsilon\}$—set of ϵ-optimal states with $\epsilon > 0$;
- $B(x) = \{y \in E : b(y) \leq b(x)\}$—set of states which are better or equal than state x according to the objective function.

At this point we are able to introduce the main theorem on top of which we base the analysis conducted in this paper.

Theorem 2.1 (*Condition of EA Convergence [17]*). *Let $A_\epsilon = \{x \in E : d(x) < \epsilon\}$ with some $\epsilon > 0$ be the set of ϵ-optimal states. An evolutionary algorithm, whose stochastic kernel satisfies*

- $\forall_{x \in A_\epsilon^c = E \setminus A_\epsilon} K(x, A_\epsilon) \geq \delta > 0$
- $\forall_{x \in A_\epsilon} K(x, A_\epsilon) = 1$

will converge to the global minimum of a real-valued objective function $f : \mathcal{M} \to \mathbb{R}$ with $f > -\infty$ defined on an arbitrary space $\mathcal{M}$, regardless of the initial distribution.

Proof. See [17]. □

Corollary 1. *The random sequence $(d(X_t) : t \geq 0)$ converges in probability to 0 (denoted as $d(X_t) \xrightarrow{P} 0$) and converges completely to 0 (denoted as $d(X_t) \xrightarrow{0} 0$).*

3. Convergence analysis in a fault-free environment

In the EAs analyzed in this article one iteration of the algorithm is divided into two phases: modification and selection. Associated Markovian kernel can be described as a product kernel defined below [17]:

$$K(x, A) = (K_m K_s)(x, A) = \int_E K_m(x, \mathrm{d}y) \cdot K_s(y, A) \tag{8}$$

where K_m is a modification (mutation/mutation-crossover) kernel and K_s is a selection kernel. In all algorithms presented here, the selection phase is performed by an *elitist selection*. It can be realized in many ways, but all of them have in common one property: best individual in current population is not worse than the best individual from a previous one.

3.1. Convergence of $(1+1)$ EA

In this section we will recall (from [17]) the convergence analysis operated on the most simple variant of an EA that is still of theoretical and practical interest: the $(1+1)$ EA. In this case, the population consists only of a single individual, which in every iteration of the algorithm, is used to generate exactly one offspring. Therefore the state space is $E = \mathcal{M}$. For an illustration of the algorithm execution, see Fig. 2.

It is assumed that the phase of generating the offspring is performed by modification (in this case mutation only) kernel $K_m(x, A)$, which allows transition to any set $A_\epsilon \subseteq \mathcal{A}$ from any state $x \in E$, in short,

$$\forall_{\epsilon > 0} \forall_{x \in E} \forall_{A_\epsilon \subseteq \mathcal{A}} K_m(x, A_\epsilon) > 0. \tag{9}$$

Additionally, each individual is modified at random independently. The selection phase is described by the elitist selection kernel:

$$K_s(y, A; x) = \mathbb{1}_{A\cap B(x)}(y) + \mathbb{1}_A(x) \cdot \mathbb{1}_{B^c(x)}(y) \tag{10}$$

and may be interpreted as follows (quoting [17]):

> "If state $y \in E$ is better or equal than state x (*i.e.* $y \in B(x)$) and also in set A, then y transitions to set A, and more precisely to set $A \cap B(x)$, with probability one. If y is worse than x (*i.e.* $y \in B^c(x)$) then y is not accepted. Rather, y will transition to the old state x with probability one. But if x was in set A then y will transition to $x \in A$ with probability one. All other cases have probability zero".

Combining these two definitions together gives the following product kernel[3]:

$$\begin{aligned} K(x, A) &= \int_E K_m(x, \mathrm{d}y) \cdot K_s(y, A) \\ &= K_m(x, A \cap B(x)) + \mathbb{1}_A(x) \cdot K_m(x, B^c(x)). \end{aligned}$$

Applying restriction to the set A_ϵ of ϵ-optimal solutions, then either $A_\epsilon \subset B(x)$, in which case $x \notin A_\epsilon$ ($\mathbb{1}_{A_\epsilon}(x) = 0$), $A_\epsilon \cap B(x) = A_\epsilon$ and

$$K(x, A_\epsilon) = K_m(x, A_\epsilon).$$

Or in case of $B(x) \subseteq A_\epsilon$, $x \in A_\epsilon$ (*i.e.* $\mathbb{1}_{A_\epsilon}(x) = 1$), $A_\epsilon \cap B(x) = B(x)$ and

$$K(x, A_\epsilon) = K_m(x, B(x)) + K_m(x, B^c(x)) = 1 \qquad \text{from (5)}$$

Therefore, the Markovian kernel restricted to the set A_ϵ is

$$K(x, A_\epsilon) = \mathbb{1}_{A_\epsilon}(x) + \mathbb{1}_{A_\epsilon{}^c}(x) \cdot K_m(x, A_\epsilon)$$

which fulfills the preconditions of Theorem 2.1 and consequently the algorithm will converge to the global minimum of a real-valued objective function.

3.2. Convergence of population-based EA

In our model, population-based EA consists of N individuals (from arbitrary set $\mathcal{M}$), which in every iteration of the algorithm are used to generate N offspring (also from the same set $\mathcal{M}$). Therefore, the state space is $E = \mathcal{M}^N$.

Now it is needed to replace previously defined mutation kernel K_m by the corresponding modification kernel, fulfilling the same properties (summarized in Eq. (9)). With special version of elitist selection, selection kernel could be defined as follows:

- if $y \in E$ is in $B(x) \cap A$ then population transitions to A,
- if $y \in E$ is not in $B(x)$ (*i.e.* the best individual of population y is worse than the best individual of population x) then entire population is rejected.

So the population-based selection kernel is identical to the individual-based selection kernel defined in Eq. (10) and structure of kernel K defined in case of $(1 + 1)$ EA remains valid.

But under usual elitist selection (which is analyzed further in this paper), in second case described above, the best individual is re-inserted – somehow – into population y yielding $y' = e_{\text{best}}(x, y) \in B(x)$. Where the map $e_{\text{best}} : E \times E \to E$ encapsulates the method to re-insert the best individual of $x \in E$ into y. Consequently, the elitist selection kernel becomes:

$$K_s(y, A; x) = \mathbb{1}_{A\cap B(x)}(y) + \mathbb{1}_A(x) \cdot \mathbb{1}_{B^c(x)}(y) \cdot \mathbb{1}_A(e_{\text{best}}(x, y)). \tag{11}$$

For an illustration of the algorithm execution, see Fig. 3.

Combining the elitist selection kernel with the modification kernel (both defined above) gives the following product kernel[4]:

$$\begin{aligned} K(x, A) &= \int_E K_m(x, \mathrm{d}y) \cdot K_s(y, A) \\ &= K_m(x, A \cap B(x)) + \mathbb{1}_A(x) \cdot \int_{B^c(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_A(e_{\text{best}}(x, y)). \end{aligned}$$

[3] For the full development of this equation, see [17].
[4] For the full development of this equation, see [17].

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

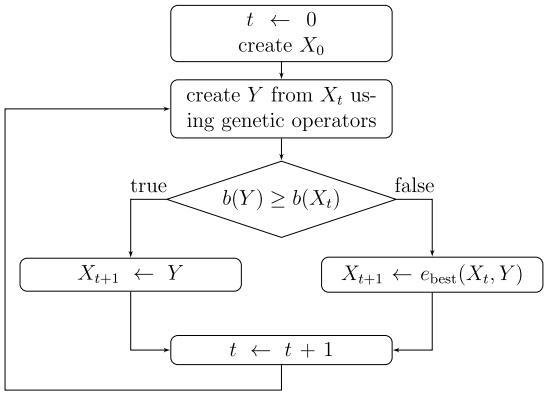

Fig. 3. Execution scheme of a population-based EA with elitist selection.

Applying restriction to the set A_ϵ of ϵ-optimal solutions, then either $A_\epsilon \subset B(x)$, in which case $x \notin A_\epsilon$ ($\mathbb{1}_{A_\epsilon}(x) = 0$), $A_\epsilon \cap B(x) = A_\epsilon$ and

$$K(x, A_\epsilon) = K_m(x, A_\epsilon).$$

Or in case of $B(x) \subseteq A_\epsilon, x \in A_\epsilon$ (*i.e.* $\mathbb{1}_{A_\epsilon}(x) = 1$), $A_\epsilon \cap B(x) = B(x)$ and since $e_{\text{best}}(x, y) \in B(x) \subseteq A_\epsilon$ (which leads to $\mathbb{1}_{A_\epsilon}(e_{\text{best}}(x, y)) = 1$):

$$K(x, A_\epsilon) = K_m(x, B(x)) + K_m(x, B^c(x)) = 1 \qquad \text{from (5)}$$

Therefore, the Markovian kernel restricted to the set A_ϵ is

$$K(x, A_\epsilon) = \mathbb{1}_{A_\epsilon}(x) + \mathbb{1}_{A_\epsilon{}^c}(x) \cdot K_m(x, A_\epsilon)$$

which fulfills preconditions of Theorem 2.1 and consequently the algorithm will converge to the global minimum of a real-valued objective function.

4. Convergence analysis in a hostile environment

This section details the main contribution of this paper, *i.e.* a theoretical analysis of the convergence (or non-convergence) of EAs in the presence of faults. In this context, the formal definition of the considered fault model is now proposed.

4.1. Fault model

4.1.1. Crash-faults

As mentioned before, heterogeneity and extreme volatility of the resources in GC platforms can lead to crash-faults—at any moment, owners of the resources can reclaim them without warning. Additionally with such large scale, there is a higher risk that nodes will experience hardware failures during the execution of an application. There are four typical approaches to tackle these problems [29]:

1. Re-execution of a task on a new resource after a failure.
2. Local checkpoints—a task is restarted on the same node from the last checkpoint after a failure.
3. Checkpointing server—checkpoints are stored on a centralized resource. After a failure, computation is restarted from the last checkpoint on a new resource.
4. Task replication—same task is sent to two or more nodes. First answer without any failure is accepted and stored. This technique has to be combined with at least one from above to deal with the failure of all nodes executing replicas of the task.

First strategy is commonly used because of its simplicity, reasonable effectiveness and low overhead [29]. In case of EA executed in a master–slave model (considered in this paper) it has been showed experimentally in [29] that it is possible to use natural fault-tolerance of this metaheuristic. Instead of implementing any kind of protection techniques, a solution based on the idea of *dynamic populations*, (*i.e.* population size is reduced and/or increased in order to minimize fitness stagnation) is proposed. In this approach individuals being computed by crashed resource are considered lost. This leads to population size being changed (decreased over time) in uncontrolled/random fashion—according to the nodes' failures.

Additionally it is assumed that the master is always in a safe state. It is justified because this node is usually under control of a single organization or a person and could be easily adjusted to tolerate failures.

Mathematical models of population-based EA proposed in this paper do not consider any particular population size for convergence proofs. Additionally convergence speed is not studied in this paper. Therefore, any of the described techniques can be used to tackle crash-faults during execution of the algorithm.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

4.1.2. Cheating fault

As explained in Section 2.2, cheaters on GC platforms are seeking to obtain credit rewards with little or no contribution to the system. The impact of cheating on a task T involved in the evolutionary process is now formalized. First, let $W_1(T)$ be the total work *i.e.* the number of unit operations of T and $W_\infty(T)$ be its depth work, *i.e.* the maximal number of unit operations on a critical path when executed on an unbounded number of processors. For the sake of simplicity, we assume that for each task T that takes part of the execution, there always exists a cheated version T' that secretly replace T when a cheating is conducted. In the sequel, T' is defined with the following constraints:

1. The prototype of T', *i.e.* its signature (as defined in the C/C++ programming language) is unaffected by the cheating. In particular, T' has the same function name, arity, argument types, and (more importantly) return type than T. This is legitimate by the fact that the cheaters want to stay in the system as long as possible. Falsifying the return type of a task will probably lead to a crash fault that can help to immediately identify the cheater. Whereas, if the attacker is satisfied with corrupting only the return value (and not its type), he can expect this modification to be hidden during a sufficient long time for him to stay in the system and collect credits.
2. The result of T' is altered such that it has the worst impact on the evolutionary process without consuming more computing power than T, *i.e.*

$$W_1(T') \leq W_1(T) \quad \text{and} \quad W_\infty(T') \leq W_\infty(T). \tag{12}$$

This last constraint models "lazy" cheaters that want little or no contribution to the system. The notion of "worst impact" is more difficult to catch and is specific to the considered task and granularity of the execution. It provides for our analysis an upper bound on the cheating impact.

The model of a cheater behavior assumed in our study is the Bernoulli process. At any given point in time a task is either cheated (with probability δ_c) or it is executed without any modifications. This decision is made without considering tasks' execution history. Additionally we assume that there is no cooperation between the cheaters.

4.2. Proof of convergence of (1 + 1) EA with cheating at mutation level

Based on mutation kernel definition in Eq. (9), cheated mutation kernel can be defined as:

$$K_m^{\text{cheat}}(x, A) = \delta_c \cdot \mathbb{1}_A(x) + (1 - \delta_c) \cdot K_m(x, A) \tag{13}$$

where δ_c corresponds to the probability of cheating. The above equation means that if cheating occurred (with probability δ_c) and state x was in set A, then it will stay in this set (lazy cheater model). If cheating did not occur then x will transition to set A with probability of uncheated version of mutation. Selection kernel is defined like previously (see Eq. (10)).

Theorem 4.1. *(1 + 1) EA with transition kernels defined above will converge to the global minimum of a real-valued objective function $f : \mathcal{M} \to \mathbb{R}$ with $f > -\infty$ defined on an arbitrary space $\mathcal{M}$, regardless of the initial distribution, despite cheating at the level of mutation.*

Proof.

$$\begin{aligned}
K(x, A) &= \int_E K_m^{\text{cheat}}(x, \mathrm{d}y) \cdot K_s(y, A; x) \\
&= \int_E K_m^{\text{cheat}}(x, \mathrm{d}y) \cdot \mathbb{1}_{A \cap B(x)}(y) + \mathbb{1}_A(x) \cdot \int_E K_m^{\text{cheat}}(x, \mathrm{d}y) \cdot \mathbb{1}_{B^c(x)}(y) && \text{from (10)} \\
&= K_m^{\text{cheat}}(x, A \cap B(x)) + \mathbb{1}_A(x) \cdot K_m^{\text{cheat}}(x, B^c(x)) && \text{from (4)} \\
&= \delta_c \cdot \mathbb{1}_{A \cap B(x)}(x) + (1 - \delta_c) \cdot K_m(x, A \cap B(x)) + \mathbb{1}_A(x) \cdot [\delta_c \cdot \mathbb{1}_{B^c(x)}(x) + (1 - \delta_c) \cdot K_m(x, B^c(x))] && \text{from (13)} \\
&= \delta_c \cdot \mathbb{1}_{A \cap B(x)}(x) + (1 - \delta_c) \cdot K_m(x, A \cap B(x)) + \delta_c \cdot \mathbb{1}_{A \cap B^c(x)}(x) + (1 - \delta_c) \cdot \mathbb{1}_A(x) \cdot K_m(x, B^c(x)) \\
&= \delta_c \cdot \mathbb{1}_A(x) + (1 - \delta_c) \cdot K_m(x, A \cap B(x)) + (1 - \delta_c) \cdot \mathbb{1}_A(x) \cdot K_m(x, B^c(x)).
\end{aligned}$$

Applying restriction to the set A_ϵ of ϵ-optimal solutions, then either $A_\epsilon \subset B(x)$ in which case $x \notin A_\epsilon$ ($\mathbb{1}_{A_\epsilon}(x) = 0$), $A_\epsilon \cap B(x) = A_\epsilon$ and

$$K(x, A_\epsilon) = (1 - \delta_c) \cdot K_m(x, A_\epsilon).$$

Or in case of $B(x) \subseteq A_\epsilon$, $x \in A_\epsilon$ (*i.e.* $\mathbb{1}_{A_\epsilon}(x) = 1$), $A_\epsilon \cap B(x) = B(x)$ and

$$\begin{aligned}
K(x, A_\epsilon) &= \delta_c + (1 - \delta_c) \cdot K_m(x, B(x)) + (1 - \delta_c) \cdot K_m(x, B^c(x)) \\
&= \delta_c + (1 - \delta_c) \cdot [K_m(x, B(x)) + K_m(x, B^c(x))] \\
&= \delta_c + 1 - \delta_c = 1 && \text{from (5)}
\end{aligned}$$

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

Therefore, the Markovian kernel restricted to the set A_ϵ is

$$K(x, A_\epsilon) = \mathbb{1}_{A_\epsilon}(x) + \mathbb{1}_{A_\epsilon{}^c}(x) \cdot (1 - \delta_c) \cdot K_m(x, A_\epsilon).$$

Consequently, $(1 + 1)$ EA with cheating at the level of mutation will converge to the global minimum of a real-valued objective function (according to Theorem 2.1, see Section 2.4.2). □

4.3. Proof of non-convergence of (1 + 1) EA with cheating at selection level

Mutation kernel is defined like in Section 3.1. The elitist selection kernel with cheating becomes:

$$K_s^{\text{cheat}}(y, A; x) = \delta_c \cdot [\mathbb{1}_{A \cap B^c(x)}(y) + \mathbb{1}_A(x) \cdot \mathbb{1}_{B(x)}(y)] + (1 - \delta_c) \cdot K_s(y, A; x) \tag{14}$$

where $K_s(y, A; x)$ is defined like previously (see Eq. (10)). The above equation can be interpreted as follows: if cheating occurred (with probability δ_c) and state $y \in E$ is not only worse than state x (*i.e.* $y \in B^c(x)$) but also in set A, then y transitions to set A, and more precisely to set $A \cap B^c(x)$. If y is better than x (*i.e.* $y \in B(x)$) then y is not accepted and will transition to old state x given that $x \in A$. If cheating did not occur then y will transition to set A with probability of uncheated version of selection.

Theorem 4.2. *$(1 + 1)$ EA with transition kernels defined above may not converge to the global minimum of a real-valued objective function $f : \mathcal{M} \to \mathbb{R}$ with $f > -\infty$ defined on an arbitrary space $\mathcal{M}$, regardless of the initial distribution, due to cheating at the level of selection.*

Proof.

$$\begin{aligned}
K(x, A) &= \int_E K_m(x, \mathrm{d}y) \cdot K_s^{\text{cheat}}(y, A; x) \\
&= \delta_c \cdot \left[\int_E K_m(x, \mathrm{d}y) \cdot \mathbb{1}_{A \cap B^c(x)}(y) + \mathbb{1}_A(x) \cdot \int_E K_m(x, \mathrm{d}y) \cdot \mathbb{1}_{B(x)}(y) \right] && \text{from (14)} \\
&\quad + (1 - \delta_c) \cdot \int_E K_m(x, \mathrm{d}y) \cdot K_s(y, A; x) \\
&= \delta_c \cdot \left[K_m(x, A \cap B^c(x)) + \mathbb{1}_A(x) \cdot K_m(x, B(x)) \right] && \text{from (4)} \\
&\quad + (1 - \delta_c) \cdot \left[\int_E K_m(x, \mathrm{d}y) \cdot \mathbb{1}_{A \cap B(x)}(y) + \mathbb{1}_A(x) \cdot \int_E K_m(x, \mathrm{d}y) \cdot \mathbb{1}_{B^c(x)}(y) \right] && \text{from (10)} \\
&= \delta_c \cdot \left[K_m(x, A \cap B^c(x)) + \mathbb{1}_A(x) \cdot K_m(x, B(x)) \right] + (1 - \delta_c) \cdot \left[K_m(x, A \cap B(x)) + \mathbb{1}_A(x) \cdot K_m(x, B^c(x)) \right].
\end{aligned}$$

Applying restriction to the set A_ϵ of ϵ-optimal solutions, then either $A_\epsilon \subset B(x)$ ($\mathbb{1}_{A_\epsilon}(x) = 0$), in which case $x \notin A_\epsilon$, $A_\epsilon \cap B(x) = A_\epsilon$, $A_\epsilon \cap B^c(x) = \emptyset$ and

$$K(x, A_\epsilon) = (1 - \delta_c) \cdot K_m(x, A_\epsilon).$$

Or in the case of $B(x) \subseteq A_\epsilon$, $x \in A_\epsilon$ (*i.e.* $\mathbb{1}_{A_\epsilon}(x) = 1$), $A_\epsilon \cap B(x) = B(x)$, $A_\epsilon \cap B^c(x) = A_\epsilon \setminus B(x)$ and

$$\begin{aligned}
K(x, A_\epsilon) &= \delta_c \cdot [K_m(x, A_\epsilon \setminus B(x)) + K_m(x, B(x))] + (1 - \delta_c) \cdot \left[K_m(x, B(x)) + K_m(x, B^c(x)) \right] \\
&= \delta_c \cdot [K_m(x, A_\epsilon) - K_m(x, B(x)) + K_m(x, B(x))] + 1 - \delta_c \\
&= \delta_c \cdot K_m(x, A_\epsilon) + 1 - \delta_c.
\end{aligned}$$

Therefore, the Markovian kernel restricted to the set A_ϵ is

$$K(x, A_\epsilon) = \mathbb{1}_{A_\epsilon}(x) \cdot \left(1 - \delta_c + \delta_c \cdot K_m(x, A_\epsilon)\right) + \mathbb{1}_{A_\epsilon{}^c}(x) \cdot (1 - \delta_c) \cdot K_m(x, A_\epsilon).$$

There is a probability of losing optimal solution, such that $(1 + 1)$ EA with cheating at the level of selection may not converge to the global minimum of a real-valued objective function (according to Theorem 2.1, the above kernel does not fulfill preconditions for convergence). □

We now extend our results to regular population-based EAs.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

4.4. Proof of convergence of population-based EAs with cheating at modification level

The Markovian kernels used in this section are defined like previously:

- uncheated modification kernel $K_m(x, A)$—see Eq. (9),
- cheated modification kernel $K_m^{\text{cheat}}(x, A)$—see Eq. (13),
- elitist selection kernel $K_s(x, A)$—see Eq. (10).

Theorem 4.3. *Population-based EA with transition kernels defined above will converge to the global minimum of a real-valued objective function $f : \mathcal{M} \to \mathbb{R}$ with $f > -\infty$ defined on an arbitrary space $\mathcal{M}$, regardless of the initial distribution, despite cheating at the level of modification.*

Proof.

$$\begin{aligned}
K(x, A) &= \int_E K_m^{\text{cheat}}(x, \mathrm{d}y) \cdot K_s(y, A; x) \\
&= \int_E K_m^{\text{cheat}}(x, \mathrm{d}y) \cdot \mathbb{1}_{A\cap B(x)}(y) && \text{from (11)} \\
&\quad + \mathbb{1}_A(x) \cdot \int_E K_m^{\text{cheat}}(x, \mathrm{d}y) \cdot \mathbb{1}_{B^c(x)}(y) \cdot \mathbb{1}_A(e_{\text{best}}(x, y)) \\
&= K_m^{\text{cheat}}(x, A \cap B(x)) && \text{from (4)} \\
&\quad + \mathbb{1}_A(x) \cdot \int_{B^c(x)} K_m^{\text{cheat}}(x, \mathrm{d}y) \cdot \mathbb{1}_A(e_{\text{best}}(x, y)) && \text{from (4)} \\
&= \delta_c \cdot \mathbb{1}_{A\cap B(x)}(x) + (1 - \delta_c) \cdot K_m(x, A \cap B(x)) && \text{from (13)} \\
&\quad + \mathbb{1}_A(x) \cdot \int_{B^c(x)} [\delta_c \cdot \mathbb{1}_{\mathrm{d}y}(x) + (1 - \delta_c) \cdot K_m(x, \mathrm{d}y)] \cdot \mathbb{1}_A(e_{\text{best}}(x, y)) && \text{from (13)} \\
&= \delta_c \cdot \mathbb{1}_{A\cap B(x)}(x) + (1 - \delta_c) \cdot K_m(x, A \cap B(x)) \\
&\quad + \mathbb{1}_A(x) \cdot \delta_c \cdot \int_{B^c(x)} \mathbb{1}_{\mathrm{d}y}(x) \cdot \mathbb{1}_A(e_{\text{best}}(x, y)) + \mathbb{1}_A(x) \cdot (1 - \delta_c) \cdot \int_{B^c(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_A(e_{\text{best}}(x, y)).
\end{aligned}$$

Applying restriction to the set A_ϵ of ϵ-optimal solutions, then either $A_\epsilon \subset B(x)$, in which case $x \notin A_\epsilon$ ($\mathbb{1}_{A_\epsilon}(x) = 0$), $A_\epsilon \cap B(x) = A_\epsilon$ and

$$K(x, A_\epsilon) = (1 - \delta_c) \cdot K_m(x, A_\epsilon).$$

Or in the case of $B(x) \subseteq A_\epsilon$, $x \in A_\epsilon$ (*i.e.* $\mathbb{1}_{A_\epsilon}(x) = 1$), $A_\epsilon \cap B(x) = B(x)$ and since $e_{\text{best}}(x, y) \in B(x) \subseteq A_\epsilon$ (leading to $\mathbb{1}_{A_\epsilon}(e_{\text{best}}(x, y)) = 1$):

$$\begin{aligned}
K(x, A_\epsilon) &= \delta_c \cdot \mathbb{1}_{B(x)}(x) + (1 - \delta_c) \cdot K_m(x, B(x)) + \delta_c \cdot \int_{B^c(x)} \mathbb{1}_{\mathrm{d}y}(x) + (1 - \delta_c) \cdot \int_{B^c(x)} K_m(x, \mathrm{d}y) \\
&= \delta_c + (1 - \delta_c) \cdot K_m(x, B(x)) + (1 - \delta_c) \cdot K_m(x, B^c(x)) && \text{from (4)} \\
&= \delta_c + (1 - \delta_c) \cdot \left[K_m(x, B(x)) + K_m(x, B^c(x))\right] \\
&= \delta_c + 1 - \delta_c = 1 && \text{from (5)}
\end{aligned}$$

Therefore, the Markovian kernel restricted to the set A_ϵ is

$$K(x, A_\epsilon) = \mathbb{1}_{A_\epsilon}(x) + \mathbb{1}_{A_\epsilon{}^c}(x) \cdot (1 - \delta_c) \cdot K_m(x, A_\epsilon).$$

Consequently, population-based EA with cheating at the level of modification will converge to the global minimum of a real-valued objective function (according to Theorem 2.1, see Section 2.4.2). □

4.5. Proof of non-convergence of population-based EA with cheating at selection level

The modification kernel is defined like previously, see Eq. (9). The elitist selection kernel with cheating becomes:

$$K_s^{\text{cheat}}(y, A; x) = \delta_c \cdot \left[\mathbb{1}_{A\cap B^c(x)}(y) + \mathbb{1}_{B(x)}(y) \cdot \mathbb{1}_A(x) \cdot \mathbb{1}_A(e_{\text{worst}}(x, y))\right] + (1 - \delta_c) \cdot K_s(y, A; x) \tag{15}$$

where $K_s(y, A; x)$ is defined in Eq. (11). For this definition additional map $e_{\text{worst}} : E \times E \to E$ is needed, which encapsulates the method to insert the worst individual of $y \in E$ into x.

Eq. (15) can be interpreted as follows: if cheating occurred (with probability δ_c) and state $y \in E$ is worse than state x (*i.e.* $y \in B^c(x)$) and also in set A, then y transitions to set A, and more precisely to set $A \cap B^c(x)$. If y is better than x (*i.e.* $y \in B(x)$),

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

then y is not accepted and will transition to the state composed from the worst individuals of both populations (x and y), given that this state is in set A. If cheating did not occur then y will transition to set A with probability of uncheated version of selection.

Theorem 4.4. *Population-based EA with transition kernels defined above may not converge to the global minimum of a real-valued objective function $f : \mathcal{M} \to \mathbb{R}$ with $f > -\infty$ defined on an arbitrary space $\mathcal{M}$, regardless of the initial distribution, due to cheating at the level of selection.*

Proof.

$$\begin{aligned}
K(x, A) &= \int_E K_m(x, \mathrm{d}y) \cdot K_s^{\text{cheat}}(y, A; x) \\
&= \delta_c \cdot \left[K_m(x, A \cap B^c(x)) + \mathbb{1}_A(x) \cdot \int_{B(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_A(e_{\text{worst}}(x, y)) \right] + (1 - \delta_c) \cdot K_s(y, A; x) && \text{from (15)} \\
&= \delta_c \cdot \left[K_m(x, A \cap B^c(x)) + \mathbb{1}_A(x) \cdot \int_{B(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_A(e_{\text{worst}}(x, y)) \right] \\
&\quad + (1 - \delta_c) \cdot \left[K_m(x, A \cap B(x)) + \mathbb{1}_A(x) \cdot \int_{B^c(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_A(e_{\text{best}}(x, y)) \right]. && \text{from (11)}
\end{aligned}$$

Applying restriction to the set A_ϵ of ϵ-optimal solutions, then either $A_\epsilon \subset B(x)$ in which case $x \notin A_\epsilon$ ($\mathbb{1}_{A_\epsilon}(x) = 0$), $A_\epsilon \cap B(x) = A_\epsilon$, $A_\epsilon \cap B^c(x) = \emptyset$ and since $e_{\text{best}}(x, y) \in B(x) \subseteq A_\epsilon$ (leading to $\mathbb{1}_{A_\epsilon}(e_{\text{best}}(x, y)) = 1$):

$$K(x, A_\epsilon) = (1 - \delta_c) \cdot K_m(x, A_\epsilon).$$

Or in the case of $B(x) \subseteq A_\epsilon$, $x \in A_\epsilon$ (*i.e.* $\mathbb{1}_{A_\epsilon}(x) = 1$), $A_\epsilon \cap B(x) = B(x)$, $A_\epsilon \cap B^c(x) = A_\epsilon \setminus B(x)$ and

$$\begin{aligned}
K(x, A_\epsilon) &= \delta_c \cdot \left[K_m(x, A_\epsilon \setminus B(x)) + \int_{B(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_{A_\epsilon}(e_{\text{worst}}(x, y)) \right] \\
&\quad + (1 - \delta_c) \cdot \left[K_m(x, B(x)) + \int_{B^c(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_{A_\epsilon}(e_{\text{best}}(x, y)) \right] \\
&= \delta_c \cdot \left[K_m(x, A_\epsilon \setminus B(x)) + \int_{B(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_{A_\epsilon}(e_{\text{worst}}(x, y)) \right] + (1 - \delta_c) \cdot \left[K_m(x, B(x)) + K_m(x, B^c(x)) \right] \\
&= \delta_c \cdot \left[K_m(x, A_\epsilon) - K_m(x, B(x)) + \int_{B(x)} K_m(x, \mathrm{d}y) \cdot \mathbb{1}_{A_\epsilon}(e_{\text{worst}}(x, y)) \right] + 1 - \delta_c && \text{from (5)} \\
&\leq 1 - \delta_c \cdot [1 - K_m(x, A_\epsilon)].
\end{aligned}$$

The Markovian kernel restricted to the set A_ϵ (both cases combined together) has a probability of losing optimal solution, such that population-based EA with cheating at the level of selection may not converge to the global minimum of a real-valued objective function (according to Theorem 2.1, the above kernel does not fulfill preconditions for convergence). □

5. Cheating impact on distributed computations

The above analysis shows that the selection mechanism is crucial to achieve convergence of EAs. The selection process also includes evaluation of the individuals forming the population, which is the costliest part of the algorithm execution. The main interest in distributed computing remains the execution of this evaluation step on the workers that compose the platform.

Typically, the execution of the algorithm in such an environment is organized in the master–slave model. In this centralized approach, selection can be done by the master node (which can be a trusted resource—therefore non-cheatable) and the generation of new individuals together with their evaluation can be executed by slaves. Even though slaves (among which cheaters can be located) are not directly responsible for selection, but they can influence the process. It can be achieved by altering a fitness value assigned to a particular individual (*e.g.* in case of minimization of the objective function $f : \mathcal{M} \to \mathbb{R}$, fitness value can be reversed $f^{\text{cheat}}(x) = 1/f(x)$). This way, worse genetic material will be chosen in favor of a better one during the selection process—which is the case with elitist selection considered in this article.

The execution scheme in uncheated version presented in the above sections guarantees preservation of the "best so far" individual during transitions from one generation to the next. Therefore, this member of the population has to be treated in a special manner. It has to be assured that, in a next generation, the new best individual is indeed not worse than the previous one (even despite of possible cheating). The aforementioned can be achieved by re-evaluating the candidates for new "best so far" before updating the population.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

Table 1
Summary of the robustness (or non-robustness) of EAs in the presence of crash and cheating faults.

		Robustness Questions [23]				
		Q1	What behavior of the system makes it robust?			
		Q2	What uncertainties is the system robust against?			
		Q3	Quantitatively, exactly how robust is the system?			
	Algorithm:	Generic EA	$(1+1)$ EA		Population-based EA	
Q1	FT mechanism	Increase initial population size by $\delta\%$ [29]	None		None	
Q2	Fault model	*Crash-fault*	*Cheating fault*		*Cheating fault*	
Q2	Attack model	Up to $\delta\%$ individual lost per generation	Bernoulli lazy cheater with cheating probability δ_c			
Q2	EA step affected	All	Mutation	Selection	Modification	Selection
Q3	EA Convergence	✓ (graceful degradation [29])	✓ ($\forall\delta_c$, see Theorem 4.1)	X ($\forall\delta_c$, see Theorem 4.2)	✓ ($\forall\delta_c$, see Theorem 4.3)	X ($\forall\delta_c$, see Theorem 4.4)
Q3	EA Robustness	Robust (ABFT)	Robust (ABFT)	Not-robust	Robust (ABFT)	Not-robust

6. Conclusion

In this paper, we studied the fault-tolerance nature of Evolutionary Algorithms (EAs) when executed in a distributed environment subjected to malicious acts. Whereas the impact of crash fault is covered, our main contribution resides in the deep analysis of a more complex kind of fault called *cheating faults* that can be modeled as the surreptitious alteration of the output values produced by some or all tasks of the program being executed. Such a selfish behavior is unfortunately common on Global Computing (GC) platforms where cheaters attempt to benefit from the system with little or no contribution.

Whereas some preliminary studies illustrate the remarkable resilience of EAs against crash faults, the enclosed convergence results by means of Markov chain modeling, offer new theoretical insights on the convergence of EAs despite the presence of cheaters. More precisely, the proposed analysis permits to conclude formally on the robustness (or non-robustness) of EA, as defined in Section 2.2.1. Table 1 summarizes our contributions from this perspective.

The fact that there exists some cases where the EA *always* converge despite the presence of cheating faults is quite encouraging. This will promote the usage of EAs in the future developments around distributed computing platform such as Desktop Grids and Volunteer Computing Systems or Cloud systems where the resources cannot be fully trusted. In particular, our work shows that the modification step of an EA can be "safely" executed on untrusted workers: whereas cheating cannot be avoided at this level, they will not impact the convergence of the EA process towards valid and (hopefully) optimal solutions. In this sense, an EA can be considered as an Algorithm-Based Fault Tolerance (ABFT) technique without any specific fault-tolerant mechanism. Alternatively, our study also highlights that as soon as a cheating happens at the level of the selection, there is a chance that the algorithm will *not* converge. It means that in this case, additional mechanisms should be considered to cover the cheating. A potential approach at this level consists in the partial or total duplication of the selection task, typically on trusted resources.

As a future work, we are currently working on experimental validation of the presented theorems. Additionally, our theoretical analysis could be extended to other models of EAs (with different selection and modification schemas). We are interested in particular in a coarse-grain distributed EA commonly referred to as the island model for EA. It is also interesting to analyze in which steps of the algorithm cheating has the biggest influence on the evolutionary process.

Also cheater models could be further extended to be more sophisticated. Instead of choosing cheating points at random, they could be selected according to their harmfulness. Cooperation of cheaters is also not considered in this paper and can be of interest for further research.

Moreover, we are now interested in the formal analysis of the overhead induced by the cheating fault when we prove the convergence of the EA. Indeed, the fact that the EA still converges towards valid and optimal solutions despite the presence of cheating faults does not mean that this will happen in a reasonable execution time (when comparing to an execution in a fault-free environment).

Acknowledgment

The authors are grateful to Xavier Besseron for his comments on this work.

Appendix A. Table of notations

In this article, we use the classical notations found in the literature relative to the convergence of EA. In particular, most of the conventions taken in this work are directly inspired by the one found in the seminal article of Rudolph [17]. They are now briefly reminded.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

	Description
$\mathcal{M}$	Search space
$f : \mathcal{M} \to \mathbb{R}$	Fitness/objective function minimized by EA, bounded from below $\forall_{x\in\mathcal{M}} f(x) > -\infty$
N	Number of individuals
$X_0 = (X_{0,1}, \ldots, X_{0,N})$	Initial population (at step $t = 0$) chosen according to some initial distribution $p(.)$; $\forall_{t\geq 0}\forall_{i=1,\ldots,N} X_{t,i} \in \mathcal{M}$
$E = \mathcal{M}^N$	State space
$K(.,.)$	Stochastic kernel describing transition of population from step t to $t+1$ for $t > 0$ using so-called genetic operators
$(X_t : t \geq 0)$	Markov chain with values in a set E of a measurable space $(E, \mathcal{A})$
$f^* = \min\{f(x) : x \in \mathcal{M}\}$	Global optimum

We also use the following definitions:

- $K^{(t)}(x, A) = \begin{cases} K(x, A), & t = 1 \\ \int_E K^{(t-1)}(y, A)K(x, \mathrm{d}y), & t > 1 \end{cases}$—$t$-th iteration of the Markovian kernel for any set A;
- $b(X_t) = \min\{f(X_{t,i}) : i = 1, \ldots, N\}$—the best objective/fitness function value of population X_t at step $t \geq 0$;
- f^*—global minimum of the objective/fitness function $f : \mathcal{M} \to \mathbb{R}$;
- $d(X_t) = b(X_t) - f^*$—distance of the best objective/fitness function value of population X_t at step $t \geq 0$ to the global optimum f^*;
- $A_\epsilon = \{x \in E : d(x) < \epsilon\}$—set of ϵ-optimal states with $\epsilon > 0$;
- $B(x) = \{y \in E : b(y) \leq b(x)\}$—set of states which are better or equal than state x according to the objective function.

Appendix B. Acronyms used

ABFT	Algorithm-Based Fault Tolerance
dEAs	distributed Evolutionary Algorithms
DGVCSs	Desktop Grids and Volunteer Computing Systems
EA	Evolutionary Algorithm
EAs	Evolutionary Algorithms
GA	Genetic Algorithm
GC	Global Computing
GP	Genetic Programming
IPS	Interacting Particle System

References

[1] W.T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye, D. Anderson, A new major SETI project based on project serendip data and 100,000 personal computers, in: Proceedings of the Fifth Intl. Conf. on Bioastronomy.
[2] D.P. Anderson, BOINC: a system for public-resource computing and storage, in: 5th IEEE/ACM Int. Workshop on Grid Computing, Pittsburgh, USA.
[3] Y.-K. Kwok, The Handbook of Computer Networks, vol. 3, John Wiley & Sons, 2007, pp. 168–188.
[4] D. Molnar, The SETI@Home problem, 2000. http://www.acm.org/crossroads/columns/onpatrol/september2000.html.
[5] H. Wasserman, M. Blum, Software reliability via run-time result-checking, Journal of the ACM 44 (1997) 826–849.
[6] L.F.G. Sarmenta, Sabotage-tolerance mechanisms for volunteer computing systems, Future Generation Computer Systems 18 (2002) 561–572.
[7] W. Du, J. Jia, M. Mangal, M. Murugesan, Uncheatable grid computing, in: Proceedings of 24th Intl. Conf. on Distributed Computing Systems, pp. 4–11.
[8] C. Germain, D. Monnier-Ragaigne, Grid result checking, in: Proceedings of the 2nd Conference on Computing Frontiers, ACM Press, Ischia, Italy, 2005, pp. 87–96.
[9] S. Varrette, Sécurité des architectures de Calcul distribué: authentification et certification de résultats, Ph.D. Thesis, INP Grenoble et Université du Luxembourg, 2007.
[10] F.F. De Vega, A fault tolerant optimization algorithm based on evolutionary computation, in: Proceedings of the International Conference on Dependability of Computer Systems, DEPCOS-RELCOMEX'06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 335–342.
[11] J.I. Hidalgo, J. Lanchares, F. Fernández de Vega, D. Lombrana, Is the island model fault tolerant? in: GECCO'07: Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation, ACM, London, United Kingdom, 2007, pp. 2737–2744.
[12] A. Morales-Reyes, E.F. Stefatos, A.T. Erdogan, T. Arslan, Towards fault-tolerant systems based on adaptive cellular genetic algorithms, in: Proceedings of the 2008 NASA/ESA Conference on Adaptive Hardware and Systems, AHS'08, IEEE Computer Society, Noordwijk, The Netherlands, 2008, pp. 398–405.
[13] S. Varrette, M. Ostaszewski, P. Bouvry, Nature inspired algorithm-based fault tolerance on global computing platforms. Application to symbolic regression, in: Proc. of the Int. Conf. on Metaheuristics and Nature Inspired Computing, META'08, Hammamet, Tunisia.
[14] D.L. González, F.F. de Vega, H. Casanova, Characterizing fault tolerance in genetic programming, in: Proc. of the 2009 Workshop on Bio-Inspired Algorithms for Distributed Systems, BADS'09, ACM, New York, NY, USA, 2009, pp. 1–10.
[15] S. Varrette, E. Tantar, P. Bouvry, On the resilience of [distributed] evolutionary algorithms against cheaters in global computing platforms, in: Proc. of the 14th Intl. Workshop on Nature Inspired Distributed Computing, NIDISC 2011, IEEE Computer Society, Anchorage (Alaska), USA, 2011, Part of the 25th IEEE/ACM Intl. Parallel and Distributed Processing Symposium (IPDPS 2011).
[16] P. Del Moral, L. Kallel, J.E. Rowe, Modeling genetic algorithms with interacting particle systems, Revista de Matem Gtica: Teor Ra y Aplicacione 8 (2001) 19–77.
[17] G. Rudolph, Convergence of evolutionary algorithms in general search spaces, in: International Conference on Evolutionary Computation, IEEE, 1996, pp. 50–54.
[18] C. Darwin, The Origin of Species, John Murray, 1859.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004

[19] L. Trujillo, G. Olague, Automated design of image operators that detect interest points, Evolutionary Computation 16 (2008) 483–507.
[20] N. Melab, S. Cahon, E. Talbi, Grid computing for parallel bioinspired algorithms, Journal of Parallel and Distributed Computing 66 (2006) 1052–1061.
[21] A. Avizienis, J.-C. Laprie, B. Randell, C.E. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transactions on Dependable and Secure Computing 1 (2004) 11–33.
[22] B. Bertholon, C. Cérin, C. Coti, J.-C. Dubacq, S. Varrette, Distributed Systems: Design and Algorithms, vol. 1, Whiley and Son, 2011, pp. 243–306.
[23] A. Vosoughi, K. Bilal, S.U. Khan, N. Min-Allah, J. Li, N. Ghani, P. Bouvry, S. Madani, A multidimensional robust greedy algorithm for resource path finding in large-scale distributed networks, in: Proceedings of the 8th International Conference on Frontiers of Information Technology, FIT'10, ACM, New York, NY, USA, 2010, pp. 16:1–16:6.
[24] K. Huang, J. Abraham, Algorithm-based fault tolerance for matrix operations, IEEE Transactions on Computers C (1984) 518–528. (Sp. issue Reliable & FT Comp.).
[25] J.S. Plank, Y. Kim, J.J. Dongarra, Algorithm-based diskless checkpointing for fault-tolerant matrix operations, in: FTCS'95: Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing, IEEE Computer Society, Washington, DC, USA, 1995, p. 351.
[26] Z. Chen, J.J. Dongarra, Algorithm-based checkpoint-free fault tolerance for parallel matrix computations on volatile resources, in: International Parallel and Distributed Processing Symposium, IPDPS 2006.
[27] A. Morales-Reyes, E.F. Stefatos, A.T. Erdogan, T. Arslan, Towards fault-tolerant systems based on adaptive cellular genetic algorithms, in: AHS'08: Proceedings of the 2008 NASA/ESA Conference on Adaptive Hardware and Systems, IEEE Computer Society, Washington, DC, USA, 2008, pp. 398–405.
[28] E.H.L. Aarts, A.E. Eiben, K.M. van Hee, A general theory of genetic algorithms, in: Computing Science Notes, Univ. of Technology, Dep. of Mathematics and Computing Science, Computing Science Section, 1989.
[29] D.L.N. González, F.F. de Vega, H. Casanova, Characterizing fault tolerance in genetic programming, Future Generation Computer Systems 26 (2010) 847–856.

Please cite this article in press as: J. Muszyński, et al., Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters, Computers and Mathematics with Applications (2012), doi:10.1016/j.camwa.2012.03.004