Tackling Adversarial Faults in Panmictic Evolutionary Algorithms

Carlos Cotta*
ccottap@lcc.uma.es
ITIS Software, University of Málaga
Málaga, Spain

ABSTRACT

We analyze the performance of panmictic evolutionary algorithms in byzantine environments in which fitness can be computed by malicious agents. We measure the influence of the rate of unreliability of the environment, and the effect that a simple mechanism based on redundant computation can have on the results attained.

CCS CONCEPTS

• **Computing methodologies** → **Genetic algorithms**; **Randomized search**; • **Theory of computation** → **Design and analysis of algorithms**.

KEYWORDS

Evolutionary Algorithms, Byzantine faults, Panmixia, Resilience

ACM Reference Format:
Carlos Cotta. 2023. Tackling Adversarial Faults in Panmictic Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3583133.3596426

1 INTRODUCTION

When using emerging computational environments such as peer-to-peer networks and volunteer computing networks [5] for running computationally intensive tasks –such as evolutionary algorithms (EAs)– challenges are manifold due to the dynamic nature and irregularity of the resulting computational landscape [1], hence underpinning the need for algorithmic resilience [3]. In this work we focus on another hazard source, namely malicious activities [9]. More precisely, we consider *cheating faults*, whereby a contributor of computational resources alters the outcome of the computation by purposefully submitting wrong results. It has been shown that these faults can have an impact on EAs depending on which components of the algorithm are affected by such faults [6], and that cellular EAs can withstand certain types of byzantine failures [7]. We focus here on analyzing the effect on EAs of other types of cheating faults, and how to counter them.

*This work is supported by Spanish Ministry of Science and Innovation under project Bio4Res (PID2021-125184NB-I00 – http://bio4res.lcc.uma.es) and by Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3596426

2 ALGORITHMIC SETTING

Let us consider an EA with a panmictic population. This algorithm optimizes a certain objective function $f(\cdot)$, which we will denote as the *true fitness* function. In order to evaluate this function for the individuals in the population, let us further assume that a master-slave model is used, i.e., the task is distributed among a network of computational nodes that provide this fitness evaluation service, some of which are *cheaters*. As a initial step, we consider a very simple model in which a fitness evaluation request returns a wrong result with some probability ρ (similarly to [7]). Let this wrong result $\hat{f}^t(\cdot)$ be termed the *unreliable fitness*, where the superscript t is used to denote the fact that cheaters are not assumed to consistently return the same wrong result at any time evaluation of the same solution is attempted. Needless to say, we cannot determine a priori whether the value obtained when evaluating an individual is its true fitness or is a wrong value.

Following [2], we are going to consider a simple model of malicious behavior (termed inverter), whereby cheaters return a value which is negatively correlated with the true fitness in order to purposefully damage the optimization process. In our experiments, $\hat{f}^t(x) = f^t_{\max} - (f(x) - f^t_{\min})$, where $f^t_{\max}$ and $f^t_{\min}$ are the maximum and minimum fitness observed by cheaters so far. Optimization in the presence of this kind of failures has some resemblance with the case of noisy environments [4, 8] although it must be noted that the fact that the unreliable fitness does not gravitate around some underlying true fitness (as it is commonly assumed in many scenarios with uncertainty) makes the nature of the phenomenon be fundamentally different. As an initial step, we consider a simple handling mechanism which we term majority$k$. This mechanism is based on redundant computation: each individual is evaluated k times, keeping the most repeated value (or an average of the most repeated values if there was a tie).

3 RESULTS

We have experimented with four objective functions, namely 100-OneMax, (25,4)-Trap, 17-MMDP, and 100-Leading-Ones. As to the EA, it has a population of $\mu = 100$ individuals, uses binary tournament selection, single-point crossover ($p_X = .9$), bit-flip mutation (p_M equivalent to a mutation rate $1/\ell$ per bit, where ℓ is the number of bits), and elitist generational replacement. The total number of fitness evaluations is 10^6 (redundant computations are accounted for in this budget). As to the unreliability rate ρ, we focus on values between 0 and 0.5 in steps of 0.05, and use the results for $\rho = 0$ to define the base-performance of the algorithm. We will focus on the majority3 handler. We perform 50 runs for each parameter setting and problem. Table 1 shows the results obtained.

Notice firstly that while OneMax is barely affected due to its simplicity, other problems result in noticeably different performance. This is more clearly seen for the Leading-Ones problem in which

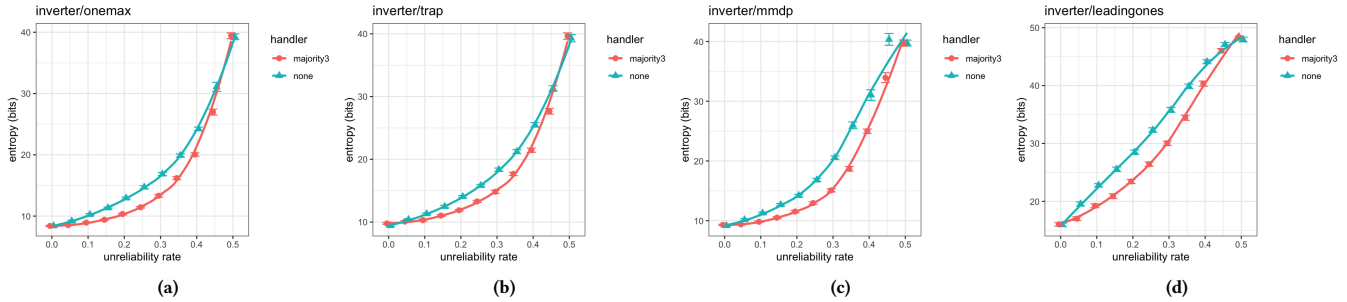

Figure 1: Minimum population entropy as a function of the unreliability rate. The data points are the average of 50 runs and the error bars span the standard error of the mean. (a) OneMax (b) Trap (c) MMDP (d) Leading-Ones

Table 1: Results for the inverter model using no handler (four leftmost columns) and using the majority3 handler (four rightmost columns). Each entry in the table indicates the mean and standard error of the mean of the best true fitness generated for the corresponding problem and unreliability rate, and two symbols indicating the the statistical significance (according to a Wilcoxon test) of the difference with respect to the base case (no handler, $\rho = 0$) and to the corresponding case (same ρ) with no handler. •, ◦ and no symbol indicate significance at $\alpha = .05$, $\alpha = 0.1$ and no statistical significance respectively.

	none				majority3			
ρ	OneMax	Trap	MMDP	Leading-Ones	OneMax	Trap	MMDP	Leading-Ones
0.00	100.00 ± 0.00	23.75 ± 0.13	17.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	22.64 ± 0.09••	17.00 ± 0.00	100.00 ± 0.00
0.05	100.00 ± 0.00	23.69 ± 0.11	17.00 ± 0.00	97.20 ± 0.69•	100.00 ± 0.00	22.59 ± 0.12••	17.00 ± 0.00	99.68 ± 0.21◦•
0.10	100.00 ± 0.00	23.74 ± 0.10	17.00 ± 0.00	89.06 ± 1.00•	100.00 ± 0.00	22.59 ± 0.12••	17.00 ± 0.00	97.74 ± 0.62••
0.15	100.00 ± 0.00	24.06 ± 0.09◦	17.00 ± 0.00	81.12 ± 0.80•	100.00 ± 0.00	22.63 ± 0.10••	17.00 ± 0.00	93.78 ± 0.91••
0.20	100.00 ± 0.00	23.84 ± 0.10	17.00 ± 0.00	73.68 ± 0.90•	100.00 ± 0.00	22.48 ± 0.11••	17.00 ± 0.00	87.02 ± 1.04••
0.25	100.00 ± 0.00	23.50 ± 0.09•	17.00 ± 0.00	63.06 ± 0.58•	100.00 ± 0.00	22.52 ± 0.10••	16.99 ± 0.01	76.88 ± 0.92••
0.30	100.00 ± 0.00	23.25 ± 0.08•	16.88 ± 0.04•	54.88 ± 0.61•	100.00 ± 0.00	22.51 ± 0.10••	16.97 ± 0.01••	66.72 ± 0.86••
0.35	100.00 ± 0.00	22.88 ± 0.07•	15.91 ± 0.12•	46.98 ± 0.71•	100.00 ± 0.00	22.40 ± 0.09••	16.85 ± 0.03••	55.80 ± 0.61••
0.40	100.00 ± 0.00	22.39 ± 0.07•	14.64 ± 0.17•	35.26 ± 0.42•	100.00 ± 0.00	21.89 ± 0.08••	16.09 ± 0.10••	42.56 ± 0.63••
0.45	98.78 ± 0.85	22.28 ± 0.06•	14.11 ± 0.18•	24.36 ± 0.40•	100.00 ± 0.00	21.76 ± 0.09••	14.49 ± 0.12•	27.22 ± 0.48••
0.50	83.20 ± 2.41•	20.10 ± 0.49•	13.49 ± 0.22•	16.92 ± 0.39•	83.48 ± 2.37•	19.42 ± 0.45••	13.15 ± 0.18•◦	15.24 ± 0.35••

there is a very marked degradation. For this problem and MMDP majority3 provides a statistically significant improvement for a broad range of values of ρ. Quite interestingly, the performance of the raw EA improves for Trap when ρ is moderately low. This is due to a boost in diversity that allows the algorithm escaping from local optima. Indeed, as seen in Figure 1, the use of the majority3 handler keeps the search more focused (which is reflected in a lower population diversity), and this seems counterproductive in this particular problem.

4 CONCLUSIONS

We have here studied the impact that the presence of adversarial cheaters has in the performance of panmictic EAs. It has been shown that these can pose a problem even for simple objective functions. A simple model based on redundant computation can provide improved results in terms of solutions attained, albeit the computational tradeoffs involved deserve further study. We are currently working towards more sophisticated handling mechanisms and algorithmic models. More realistic problems will be considered as well.

REFERENCES

[1] David Camacho et al. 2018. From ephemeral computing to deep bioinspired algorithms: New trends and applications. *Future Generation Computer Systems* 88 (2018), 735–746.
[2] Carlos Cotta. 2023. On the Performance of Evolutionary Algorithms with Unreliable Fitness Information. In *EvoStar 2023 Late Breaking Abstracts*, Antonio M. Mora (Ed.). Brno, Czech Republic, 4 pages.
[3] Carlos Cotta and Gustavo Olague. 2022. Resilient Bioinspired Algorithms: A Computer System Design Perspective. In *Applications of Evolutionary Computation*, Juan Luis Jiménez Laredo et al. (Eds.). Lecture Notes in Computer Science, Vol. 13224. Springer, Cham, 619–631.
[4] Yaochu Jin and Jürgen Branke. 2005. Evolutionary Optimization in Uncertain Environments – A Survey. *IEEE Transactions on Evolutionary Computation* 9, 3 (2005), 303–317.
[5] Tessema M. Mengistu and Dunren Che. 2019. Survey and Taxonomy of Volunteer Computing. *Comput. Surveys* 52, 3, Article 59 (2019), 35 pages.
[6] Jakub Muszyński et al. 2012. Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters. *Computers & Mathematics with Applications* 64, 12 (2012), 3805–3819.
[7] Jakub Muszyński et al. 2015. Distributed Cellular Evolutionary Algorithms in a Byzantine Environment. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. IEEE Press, Hyderabad, India, 307–313.
[8] Pratyusha Rakshit, Amit Konar, and Swagatam Das. 2017. Noisy evolutionary optimization algorithms – A comprehensive survey. *Swarm and Evolutionary Computation* 33 (2017), 18–45.
[9] Luis F.G. Sarmenta. 2002. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems* 18, 4 (2002), 561–572.