Deterministically-Adaptive Genetic Algorithm To Solve Binary Communication Problems: Application On The Error Correcting Code Problem

Zakaria Abd El Moiz Dahi
Modelling and implementation of complex systems laboratory
Dept. of new technologies of information and communication
Constantine 2 university
Constantine City, Algeria
zakaria.dahi@univ-constantine2.dz

Mezioud Chaker
Modelling and implementation of complex systems laboratory
Dept. of new technologies of information and communication
Constantine 2 university
Constantine City, Algeria
chaker.mezioud@univ-constantine2.dz

Amer Draa
Modelling and implementation of complex systems laboratory
Dept. of new technologies of information and communication
Constantine 2 university
Constantine City, Algeria
draa.amer@gmail.com

ABSTRACT

Global optimisation plays a critical role in today's scientific and industrial fields. Optimisation problems are either continuous or combinatorial depending on the nature of the parameters to optimise. In the class of combinatorial problems, we find a sub-category which is the binary optimisation problems. Due to the complex nature of optimisation problems, exhaustive search-based methods are no longer a good choice. So, metaheuristics are more and more being opted in order to solve such problems. On the other hand, most of the proposed metaheuristics were hand-tuned through a long and exhaustive process that requires advanced knowledge. This fact makes them sensitive to any change of the problem properties, that probably might decrease their efficiency. So, their further application in real-life scenarios will be restricted or impossible. One of the most active topic of research of nowdays is the adaptation strategies. These last ones appear as a promising alternative to the hand-tuned approach. Deterministic adaptation is one of the several adaptation schemes that exist. Based on the latter, in this paper we propose several variants of one of the most studied metaheuristics; the Genetic Algorithm (GA). The efficiency of the variants was assessed for solving a complex optimisation problem in cellular networks which is the Error Correcting Code Problem (ECCP). They were compared against a classical hand-tuned genetic algorithm. The experiments gave promising results and encourage further investigation.

Categories and Subject Descriptors

H.4 [**Artificial Intelligence**]: Metaheuristics, Adaptation Strategies; D.2.8 [**Models and Principles**]: Systems and Information Theory—*Information theory*

IPAC '15, November 23-25, 2015, Batna, Algeria
© 2015 ACM. ISBN 978-1-4503-3458-7/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2816839.2816843

Keywords

Cellular Phone Networks, Transmission, Error Correcting Code Problem, Evolutionary Algorithms, Genetic Algorithm, Adaptation Strategies.

1. INTRODUCTION

Combinatorial problems are problems whose parameters belong to a finit set of integer ($x_i \in \mathbb{N}$). The latter includes a more specific type called *binary optimisation problems*: whose parameters can take values from a bi-valued search space called *genotype space* ($x_i \in \{0, 1\}$).

Transmission and signal processing are key factors in communication networks in general and specially in cellular networks (2G, EDGE, GPRS, 3G, 3G$^+$, LTE, 4G). In fact, any dysfunction of this sensitive core may alter the service quality of the network itself. Regarding the actual high standards of the mobile phone industry, these kinds of transmission failures are unacceptable for the telephony operators and the clients as well. Thus, the *Error Correcting Code Problem* (*ECCP*) is one of the most challenging optimisation issues in cellular networks. The ECCP has been formulated as a binary optimisation problem and was proven to be complex to solve.

Metaheuristics are efficient tools to use when engaging such optimisation problems. Based on the source of their inspiration, metaheuristics can be divided into two types. *Evolutionary Algorithms* (EAs): are algorithms inspired by the natural laws of evolution where the fittest individuals survive through the time. *Swarm-Inspired Algorithms* (SIAs): are algorithms inspired by the natural phenomenon of swarm movements. The *Genetic Algorithm* (GA) is one of the first proposed evolutionary metaheuristics [4], and also one of the most studied ones.

The main motivation behind the use of evolutionary algorithms is solving real-world problems, which often requires a problem-specific algorithm design. The complex customisation of the algorithms to a specific problem either needs a specialist or limits the field of application to mainly scientific research. But, there is a growing demand for up-to-date optimisation software, usable by non-specialists within an industrial development environment. Majority of the scientific works propose powerful algorithms, but mostly without further considertion for an industrial application. Another possible way to avoid this is to supply the industry with intelligent algorithms containing a mechanism for modifying the parameters without external control. Many schemes of adaptation exist in the literature, but all can be regrouped in the following taxonomy: *deterministic*, *adaptive* and *self-adaptive* algorithms. In this paper we are intrested in the first class to demonstrate that simple and non-complex adaptation strategies can lead to more efficient algorithms than the hand-tuned ones.

In this paper, we propose several deterministically adaptive variants of the genetic algorithm based on several adaptation strategies with different complexities and behaviour. The performances of the proposed variants is assessed by solving the ECC problem and compared to a classical hand-tuned genetic algorithm.

The paper is structured as follows. In Section 2, the basic concepts related to the canonical Genetic Algorithm (GA), adaptation, ECC problem are presented. In section 3, the proposed variants of the genetic algorithm are introduced. Section 4 is dedicated to the experimental results and their discussion. Finally, we conclude the paper in Section 5.

2. BASIC CONCEPTS

2.1 The Genetic Algorithm

The Genetic Algorithm (GA) represents a powerful class of search and optimisation techniques developed by Holland [4] in analogy to the genetic laws and natural selection.

A basic GA consists of three components; the fitness evaluation unit, the selection unit and the genetic operators for reproduction: crossover and mutation operations. The basic genetic algorithm is summarized in pseudo-code of Algorithm 1.

Algorithm 1 . The Genetic Algorithm

```
1: Initialize population
2: Repeat
3:    Selection
4:    Reproduction: Crossover, Mutation
5:    Evaluation
6: Until requirements are met
```

The initial population required at the start of the algorithm, is a set of number strings generated by a random generator. Each string is a representation of a solution to the optimisation problem being addressed. Binary strings are commonly employed. These strings are called *chromosomes*. A fitness value computed by the evaluation unit is associated with each chromosome. A fitness value is a measure of the goodness of the solution that it represents. The aim of the genetic operators is to transform this set of strings into ones with higher fitness values.

The reproduction operator performs a natural selection known as *seeded selection*. Chromosomes are copied from one set (i.e. representing a generation of solutions) to the next, according to their fitness values. The higher the fitness value, the greater the probability that the chromosome will be selected for the next generation.

The crossover operator chooses pairs of chromosomes parents and produces new pairs. The simplest crossover operation is to cut the original parent strings at a randomly-selected point and to exchange their tails. The number of crossover operations is ruled by a crossover probability P_c.

The mutation operator randomly mutates or reverses the values of *genes* in a chromosome. The number of mutation operations is determined by a mutation probability P_m.

A phase of the genetic algorithm consists of applying the *evaluation*, *reproduction* (i.e. *crossover* and *mutation* operations). A new generation of chromosomes is produced with each phase of the algorithm. In the genetic algorithm, the best solutions are allowed to evolve subject to some fitness criterion, while internally the mechanics are left largely as a black box, since several types of selection and reproduction exist.

2.2 Adaptation in Evolutionary Algorithms

Adaptation within EAs reflects the attempt to mimic process of natural evolution. State-of-the-art EA implementations often require a comprehensive algorithmic knowledge from the user in order to choose appropriate strategy parameters for solving a specific optimisation problem. Even for an expert, the parameters configuration for an optimal performance is hard to find. The idea of adaptation is to change these parameters according to a given scheme. A classification of these schemes can be made according on how they are performed. Many classifications exist in the litarture, but one can opt for the taxonomy proposed in [1].

Parameter Tuning: Is the scheme used whenever the parameters have constant values throughout the run of the EA (i.e. there is no adaptation). Consequently, an external agent or mechanism is needed to tune the desired strategy parameters and to choose the most appropriate values. The canonical GA is based on this scheme.

Dynamic: It appears whenever there is some mechanism that modifies a strategy search parameter without any external control. The class of EA that uses this scheme can be further sub-devided into other specific sub-schemes.

Deterministic: This scheme takes place if the value of parameter is tuned constantly by some deterministic rule. This rule modifies the strategy parameter deterministically without using any feedback from the EA. This scheme is the one we are intrested in our work.

Adaptive: This scheme takes place if there is some form of *feedback* got from the EA that is used to set the direction *or/and* magnitude of the change to the strategy parameters.

Self-adaptive: In this scheme the parameters are encoded with the variables and can evolve as well as the solution itself.

In our work the main parameter that will undergo the adaptation is the mutation probability P_m. Our scientific methodology consists in using strategies with an increasing level of complexity. We firstly opted for the use of a canonical *monotone adaptation*: *linearly-decreasing* and *linearly-increasing*. Then, we thought of using a more complex adaptation strategy (i.e. not-monotone) called *oscillatory*. In this perspective, we opted for the *sinuosidal wave* with *decreasing* and *increasing amplitude*. As a final strategy, we used one of the state-of-the-art strategies whose the efficiency is well established. The choice was made for the one proposed in [2].

2.2.1 The Linear Adaptation Strategy

This strategy is based on adding (i.e when increasing) or subtracting (i.e. when decreasing) at each iteration "*t*" of the algorithm a constant "*step*" from the previous P_m value at iteration (*t - 1*). Generally, the step size used is computed using Formula 1.

$$Step = \frac{Upper\ Bound - Lower\ Bound}{Number\ of\ Iterations} \tag{1}$$

$$P_m(t) = P_m(t-1) \pm Step \tag{2}$$

Where the *Lower* and *Upper Bound* (*LB* and *UB*) are the bounds where the P_m value is supposed to evolve within. Figure 1 shows the resulting evolution of P_m value when using this strategy. Figure 1(a) represents a linearly-decreasing adaptation strategy, whereas Figure 1(b) represents a linearly increasing adaptation strategy.

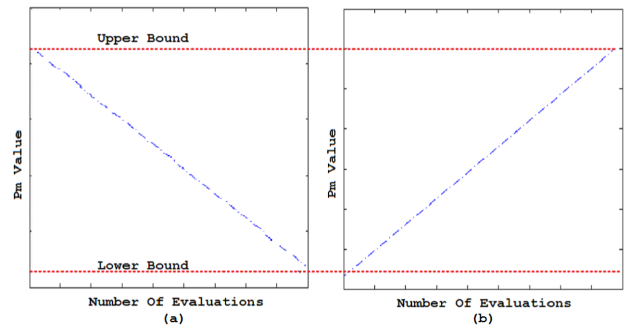

Figure 1: Linear Adaptation Strategy: (a) Decreasing and (b) Increasing

2.2.2 The Oscillatory Adaptation Strategy

This strategy is based on the formula defining a *sinuosidal wave* with a *decreasing* or an *increasing amplitude*. Formula 3 defines the sinuosidal wave with increasing amplitude, whereas Formula 4 defines a sinuosidal wave with decreasing amplitude.

$$P_m = \frac{1}{2} * \left(sin\left(2\pi * freq * it + \pi\right) * \frac{it}{Maxit} + 1 \right) \tag{3}$$

$$P_m = \frac{1}{2} * \left(sin\left(2\pi * freq * it + \pi\right) * \frac{Maxit - it}{Maxit} + 1 \right) \tag{4}$$

Where "*freq*" is a parameter controlling the frequency of the wave. "*it*" is the actual iteration and "*Max it*" is the maximum number of iterations allowed. Figure 2 shows the resulting evolution of the P_m value when using this strategy. Figure 2(a) represents an oscillatory-decreasing adaptation strategy, whereas Figure 2(b) represents an oscilatory-increasing one.

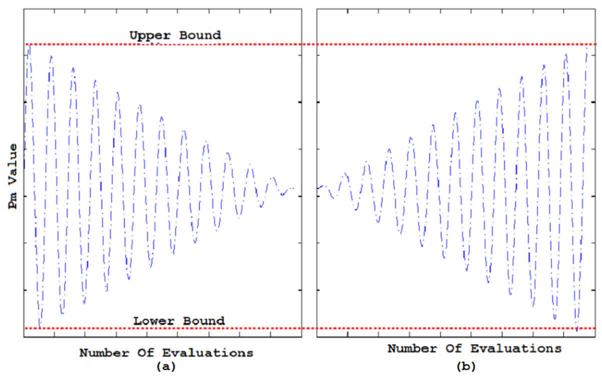

Figure 2: Oscillatory Adaptation Strategy: (a) Decreasing and (b) Increasing Amplitude

2.2.3 State-of-the-art Adaptation Strategy

This strategy was introduced by Bäck et. al [2]. It is defined by Formula 5.

$$P_m(t) = \left(2 + \frac{l-2}{T-1} * t\right) \tag{5}$$

Where "*l*" is the chromosome length, "*n*" is the population size and "*T*" is the number of generations. Figure 3 shows the resulted evolution of the P_m value when using this strategy.

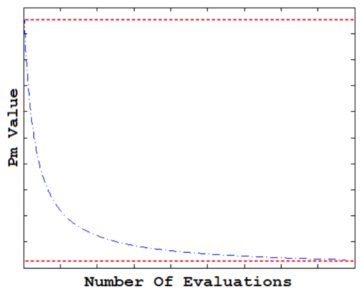

Figure 3: State-of-the-art Adaptation Strategy

2.3 Error Correcting Code Problem

The communication is one of nowadays unavoidable fields on the scientific and economic levels. Cellular networks are probably the most challenging type of communication networks of these last decades. The big popularity and availability of the services proposed by this last one made the mobile phone industry an industry with high standards where both clients and telephony operators are expecting a high level of service quality.

The cellular network optimisation is a key factor for the service quality. Any optimisation process that cannot cope with this task, will decrease the quality of services proposed. One of the critical components of communication network is the transmission core. This core is omni-present whatever the type of the network, or the technology used.

One of the optimisation issues that exist in this core, is the design of code words. This problem is known also as the *Error Correcting Code Problem* (*ECCP*) [5]. It aims at maximizing the reliability of message transmissions through a noisy channel by introducing some redunduncy in the message (i.e. using more bits than it is necessary) to increase the chances of recovering a message if some noise alter the bits contained during their transmission through a disrupted channel.

Coding a set of alphabet for transmission as binary code words using the least different code words structure (i.e. the code words must have the smallest Hamming distance between them). This, will allow to recover a disrupted code word by looking to the closet code word to this last one. We can see that if all code words are separated by at least d bits, any modification of at most $(d-1)/2$ bits in a valid code word can be easily reverted.

The ECCP can be considered as distributing M code words of N bits. Authors in [3] proposed a fitness function to evaluate the quality of the designed code words. The objective function of the ECC problem is defined by Formula 6. Where d_{ij} is the Hamming distance between code words i and j.

$$Fitness_{ECCP} = \frac{1}{\sum_{i=1}^{M} \sum_{j=1, i\neq j}^{M} \frac{1}{d_{ij}^2}} \tag{6}$$

3. THE PROPOSED APPROACH

Including one of the previously introduced strategies of adaptation of the P_m value after *line 5* of pseudo-code of Algorithm 1 gave birth to five new deterministically-adaptive variants of the genetic algorithm.

The first variant *LD-GA* (for Linearly Decreasing adaptation strategy based Genetic Algorithm) which uses a linearly-decreasing strategy. The second variant called *LI-GA* (for Linearly Increasing adaptation strategy based Genetic Algorithm) which is based on a linearly-increasing strategy. The *OD-GA* (for Oscillatory Decreasing adaptation strategy based Genetic Algorithm) which is based on a sinusoidal strategy with a decreasing amplitude. The *OI-GA* (for Oscillatory Increasing adaptation strategy based Genetic Algorithm) which is based on a sinusoidal strategy with an increasing amplitude. Finally, the *B-GA* (for Bäck adaptation strategy based Genetic Algorithm) which is based on the strategy proposed by Bäck et. al [2].

4. EXPERIMENTS AND DISCUSSION

The experiments are carried using an Intel I3 core with 2 GB Ram and a Windows 7 OS. The implementation is done using Matlab 7.12.0 (R2011a).

In this paper we conduct the experiments for designing blocks of 8 words (i.e. $M = 8$) with 4 bits length (i.e. $N = 4$).

Based on the literature that already studied the genetic algorithm (i.e. behaviour, architecture, and configuration), all the variants use an elitist binary tournament selection. They all also apply a $(\mu+\lambda)$ strategy. A two-point crossover with fixed crossover probability P_c equal to 1 is used also. For the canonical genetic algorithm (i.e. hand-tuned) who uses a static mutation probability P_m, we set the probability to $(1/L)$. Where L is the length of the chromosome. In our case it is the value $(M * N)$.

The parameter of the adaptation strategies of the proposed *LI-GA*, *LD-GA*, *OD-GA* and *OI-GA* are: Lower Bound $(LB) = (1/L)$. Where L is the length of the chromosome. In our case it is the value $(M * N)$, whereas the Upper Bound $(UB) = 0.5$.

In all the experiments, variants of the GA and the canonical GA use a population of 175 chromosomes. The experiments are performed till reaching a stopping criterion of 175000 evaluations. The experiments are repeated for 20 executions.

Table 1 shows the results of the experiments when using the canonical genetic algorithm and its five variants.

Table 1: Experimental Results for the Instance (8,4) Using Canonical-GA, LD-GA, LI-GA, OD-GA, OI-GA, B-GA

Algorithm	Best	Worst	Mean	Std
Canonical-GA	16.263886	18.805553	17.564579	0.954
LD-GA	12.500001	19.749994	17.127079	1.505
LI-GA	12.500001	18.805552	**16.754164**	1.753
OD-GA	16.263887	18.805551	17.221523	0.860
OI-GA	16.263886	19.847218	18.056942	1.006
B-GA	16.263887	18.805553	18.047913	0.844

On the basis of the results of Table 1, the first finding that one can make is that the results obtained by 3/5 of the proposed adaptation strategies outperformed the results obtained by the hand-tuned genetic algorithm. These variants contains both variants based on a linear strategy (i.e. LD-GA and LI-GA) and one of the variants based on an oscillatory strategy (i.e. OD-GA). Also, we can conclude that for this size of problem a linear adaptation of the P_m value, is more efficient than an oscillatory one. In addition, these results confirm that complex strategies of adaptation are not always needed for designing more efficient algorithms.

Another observation could be made which is that in 3/5 cases the proposed variants of the genetic algorithm outperformed the state-of-the-art adaptive variant proposed in [2]. In this perspective, even among the linear strategies, one can notice that the linear increasing strategy gave better results than the decreasing one, since the variant LI-GA is the variant that gave the best results so far.

Table 2 regroups the final code words obtained by the canonical-GA, LD-GA, LI-GA, OD-GA, OI-GA and BGA. They correspond to the best individuals obtained at the end of the execution.

Table 2: Code Words Corresponding to the Best Chromosomes Obtained by the Canonical-GA, LD-GA, LI-GA, OD-GA, OI-GA, B-GA

$$\underset{\text{Canonical-GA}}{\begin{pmatrix} 0&1&1&0\\ 0&0&0&1\\ 1&1&0&0\\ 0&0&1&0\\ 1&0&0&0\\ 1&0&1&1\\ 0&0&0&0\\ 0&1&1&0 \end{pmatrix}} \quad \underset{\text{LD-GA}}{\begin{pmatrix} 1&1&1&0\\ 1&0&1&1\\ 1&0&1&1\\ 1&0&1&0\\ 0&0&0&0\\ 1&0&0&1\\ 0&1&1&0\\ 1&1&0&1 \end{pmatrix}} \quad \underset{\text{LI-GA}}{\begin{pmatrix} 0&1&1&0\\ 0&1&1&0\\ 0&1&0&0\\ 1&0&1&0\\ 1&0&0&1\\ 1&0&0&0\\ 0&1&1&0\\ 1&0&1&1 \end{pmatrix}}$$

$$\underset{\text{OD-GA}}{\begin{pmatrix} 0&1&0&1\\ 1&1&0&1\\ 0&1&1&0\\ 1&0&1&1\\ 1&0&0&0\\ 1&1&1&1\\ 0&0&1&0\\ 1&0&0&1 \end{pmatrix}} \quad \underset{\text{OI-GA}}{\begin{pmatrix} 1&1&0&0\\ 1&1&1&1\\ 1&1&0&0\\ 1&1&1&1\\ 0&0&1&0\\ 0&0&0&0\\ 1&0&1&1\\ 1&0&0&0 \end{pmatrix}} \quad \underset{\text{B-GA}}{\begin{pmatrix} 1&0&0&1\\ 0&0&1&0\\ 1&1&0&0\\ 1&0&0&0\\ 1&1&0&0\\ 0&0&1&1\\ 0&1&1&0 \end{pmatrix}}$$

Figure 4 shows the fitness value evolution using the Canonical-GA, LD-GA, LI-GA, OD-GA, OI-GA, B-GA of a randomly chosen execution. Based on these last ones, many intresting findings can be made concerning the behaviour of the genetic algorithm and its variants and also the properties of the problem itself. Since, in all cases a normal decreasing evolution of the algorithms can be noticed at the first steps of the algorithms, than a brutal stagnation is observed which tend to suppose that the problem contains many local optima. In addition, this finiding shows that different adaptation strategies may produce a GA with the same behaviour.

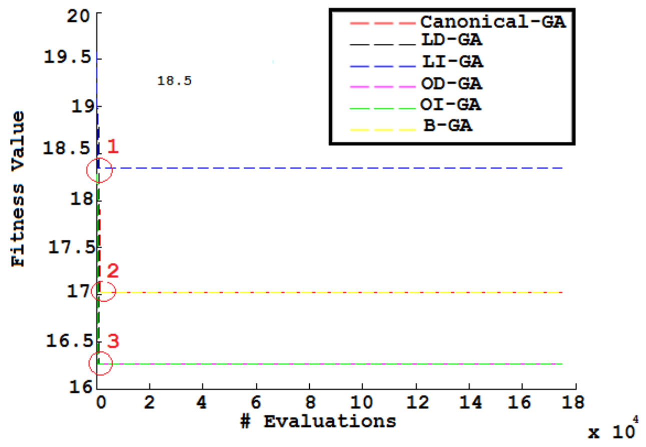

Figure 4: Fitness Value Evolution When Using the Canonical-GA, LD-GA, LI-GA, OD-GA, OI-GA and the B-GA

Figure 5 shows a more accurate display of the fitness value evolution when using the Canonical-GA, LI-GA, LD-GA, OI-GA, OD-GA and the B-GA at different specific periods labelled as 1, 2 and 3 in Figure 4.

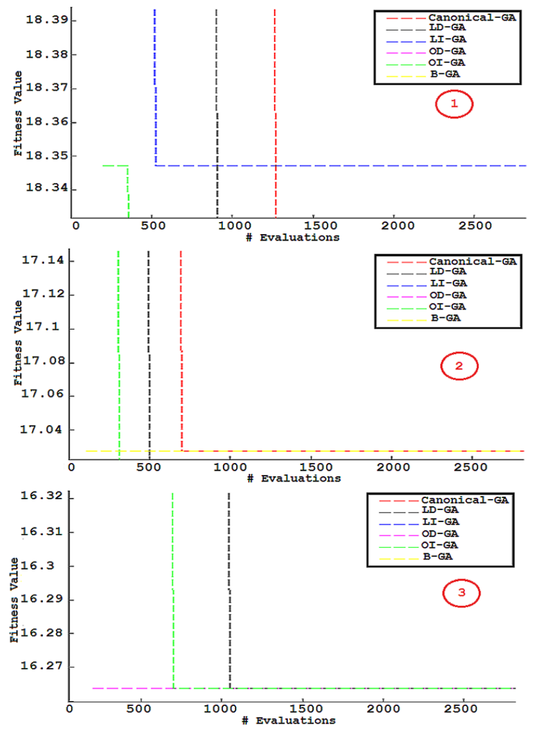

Figure 5: Fitness Value Evolution When Using the Canonical-GA, LD-GA, LI-GA, OD-GA, OI-GA and the B-GA at Periods 1, 2 and 3

5. CONCLUSION

In this paper the genetic algorithm has been modified to include adapation strategies to demonstrate the need for having more flexible and more workable algorithms in industrial fields and not only within abstract scientific research. The proposed variants where tested on the ECC problem and showed promising results.

As perspecitve, one seek to use more complex schemes of adaptation to solve more complex binary problems.

6. REFERENCES

[1] Alba, E., and Dorronsoro, B. The exploration exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation 9*, 2 (2005), 126–142.

[2] Bäck, T., and Schütz, M. Intelligent mutation rate control in canonical genetic algorithms. In *Foundations of Intelligent Systems*, vol. 1079. Springer, 1996, pp. 158–167.

[3] Dontas, K., and De Jong, K. Discovery of maximal distance codes using genetic algorithms. In *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence* (Nov 1990), pp. 805–811.

[4] Holland, J. Genetic algorithms and adaptation. In *Adaptive Control of Ill-Defined Systems*, vol. 16. Springer, 1984, pp. 317–333.

[5] León, C., Martin, S., Miranda, G., Rodriguez, C., and Rodriguez, J. Parallelizations of the error correcting code problem. In *Proceedings of the 6th International Conference on Large-Scale Scientific Computing*, vol. 4818. Springer, 2008, pp. 696–704.