ISSN 1870-4069

A First Approach to Asynchronous-Synchronous Tradeoff in 1D Cellular Genetic Algorithms

José Alejandro Cornejo Acosta[1], Jesús García Díaz[1,2]

[1] Instituto Nacional de Astrofísica, Óptica y Electrónica, Coordinación de Ciencias Computacionales, Mexico

[2] Consejo Nacional de Ciencia y Tecnología, Mexico

alexcornejo@inaoep.mx,jesus.garcia@conacyt.mx

Abstract. This paper explores the effect of the asynchronous-synchronous tradeoff in Cellular Genetic Algorithms (cGA). We perform this exploration by introducing an update policy called k-Fixed Line Sweep (kFLS). We tested the kFLS in a cGA over different uni-modal and multi-modal continuous optimization functions. The results show that for some cases, the cGA can converge more quickly to an optimal solution if the value of k is appropriately adjusted.

Keywords: Cellular genetic algorithms, asynchronous update, fixed line sweep.

1 Introduction

Cellular Genetic Algorithms (cGA) are a class of Evolutionary Algorithms (EA) that evolve a population of individuals, with overlapped neighborhoods, by applying genetic operators over a *small* neighborhood of each individual. cGAs have been used for finding near-optimal solutions to different kinds of optimization problems. This includes combinatorial [4] and multi-objective optimization [5, 6] problems. One of the advantages of cGAs is that parallelism mechanisms can be directly used. Usually, the individuals of a cGA are distributed in a regular 1D, 2D, or 3D grid [1]. Nevertheless, this paper only considers 1D grids with closed boundary conditions (a.k.a. rings).

There are many parameters that can influence a cGA's performance. These include the neighborhood, topology, and update policy. This work focuses on the latter. An update policy indicates the order in which individuals are explored and how they affect each other. Depending on the used policy, an individual's codification can spread differently across the grid. An update policy can be asynchronous or synchronous. Synchronous (SYN) update consists in independently updating all the cells and then replacing the whole population. In an asynchronous update, the cells are sequentially updated, immediately replacing the individuals. The main asynchronous update policies reported in the literature are the following [2]:

- *Uniform Choice* (UC). The cell to be updated is selected uniformly at random, with replacement.
- *Fixed Line Sweep* (FLS). The cells are updated row by row. The cells in the sequence are adjacent in the ring.
- *Fixed Random Sweep* (FRS). It is similar to FLS. However, the sequence of cells is generated at random and, once defined, it never changes.
- *New Random Sweep* (NRS). It is similar to FRS. However, every time a sequence is fully explored, a new random sequence is sampled.

Independently of the update policy, a *time step* is usually defined as updating all the cells in the grid. However, in this paper, we define a *time step* as the action of synchronously updating a subset of cells, i.e., not all of the cells are necessarily explored. This way, we set the basis for empirically investigating the tradeoff between asynchrony and synchrony. In more detail, this work introduces a generalization of the FLS update policy. We refer to this proposal as k-Fixed Line Sweep (kFLS). In fact, kFLS with $k = 1$ is equivalent to FLS and is equivalent to SYN when $k = n$, where n is the length of the ring. The following section introduces kFLS in detail.

The remaining part of the paper is structured as follows. Section 2 introduces the kFLS update policy. Then, Section 3 presents the experimental setup and results. Finally, Section 4 presents the concluding remarks and possible future work.

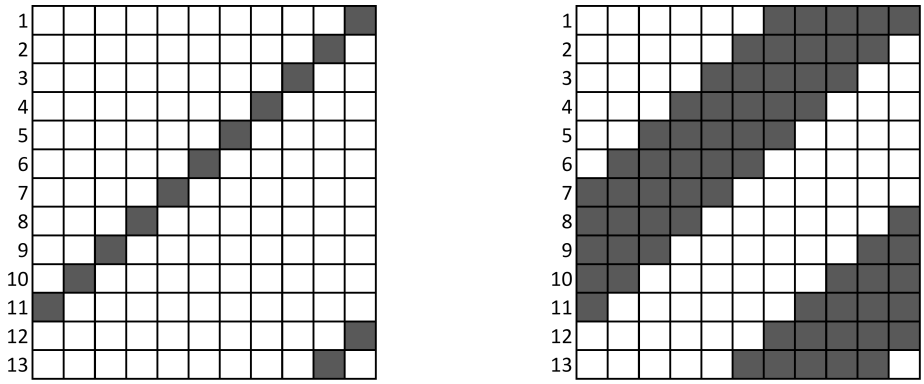
2 k-Fixed Line Sweep

This paper explores an update policy named k-Fixed Line Sweep (kFLS). This policy is straightforward. It is similar to FLS; however, it synchronously explores k adjacent cells instead of exploring cells one by one. Thus, kFLS with $k = 1$ (1FLS) is equivalent to FLS, and kFLS with $k = n$ is equivalent to SYN, where n is the number of cells. Since we are working with a 1D grid, the number of cells and the length of the ring are the same. Figure 1 shows an example of how FLS works. In these diagrams, each row represents a time step. Namely, a step where all the black cells are synchronously updated. In these figures, the length of the ring is 11, and the number of time steps is 13. Observe that the number of explored cells in Figure 1(a) is 13, and it is 65 in Figure 1(b).

As mentioned before, FLS and SYN are extreme cases of kFLS. In order to explore the effect of different values of k, we applied kFLS over a set of continuous optimization problems. The following section presents the experimental setup and results.

3 Experimental Results

This section presents the results of using the kFLS update policy over a cGA for solving a set of continuous single-objective optimization benchmark functions. These functions are separated into uni-modal and multi-modal (Tables 1 and



(a) $k = 1$ (b) $k = 5$

Fig. 1. kFLS.

Table 1. Benchmark uni-modal functions.

Name	Function	Domain
Sphere	$\sum_{i=1}^{d} x_i^2$	$[-5.12, 5.12]$
Sum Squares	$\sum_{i=1}^{d} i x_i^2$	$[-10, 10]$
Rotated Hyper-Ellipsoid	$\sum_{i=1}^{d} \sum_{j=1}^{i} x_j^2$	$[-65.536, 65.536]$
Zakharov	$\sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{d} 0.5 i x_i\right)^4$	$[-5, 10]$

2). Table 3 shows a simple configuration used for all experiments. The ring has closed boundary conditions, and its length is 100. So, we test kFLS with values $k \in \{1, 2, ..., 100\}$. The selected neighborhood is one of the simplest possible, EAST-WEST (left and right cells). Concerning recombination, the mother is the incumbent individual, and the father is randomly selected from its neighborhood. The crossover operator is the blending method, which generates the offspring by applying Eq. 1, where β is a random number in $[0, 1]$ and x_n (y_n) is the allele of the n^{th} gene in the chromosome of the mother (father) [3]. If the resulting offspring is better than the incumbent individual, it replaces it. All functions from Tables 1 and 2 have a fitness value of zero for every $d \in \mathbb{Z}^+$. All functions were solved with $d = 1$ and $d = 2$. For practical purposes, we established as "optimal" any solution of size less than 10^{-4} (10^{-2}) for $d = 1$ ($d = 2$).

$$z_n = \beta x_n + (1 - \beta) y_n, \tag{1}$$

Figures 2 and 3 show the results obtained by executing a cGA, using kFLS as update policy, over the set of uni-modal and multi-modal functions from Tables

Table 2. Benchmark multi-modal functions.

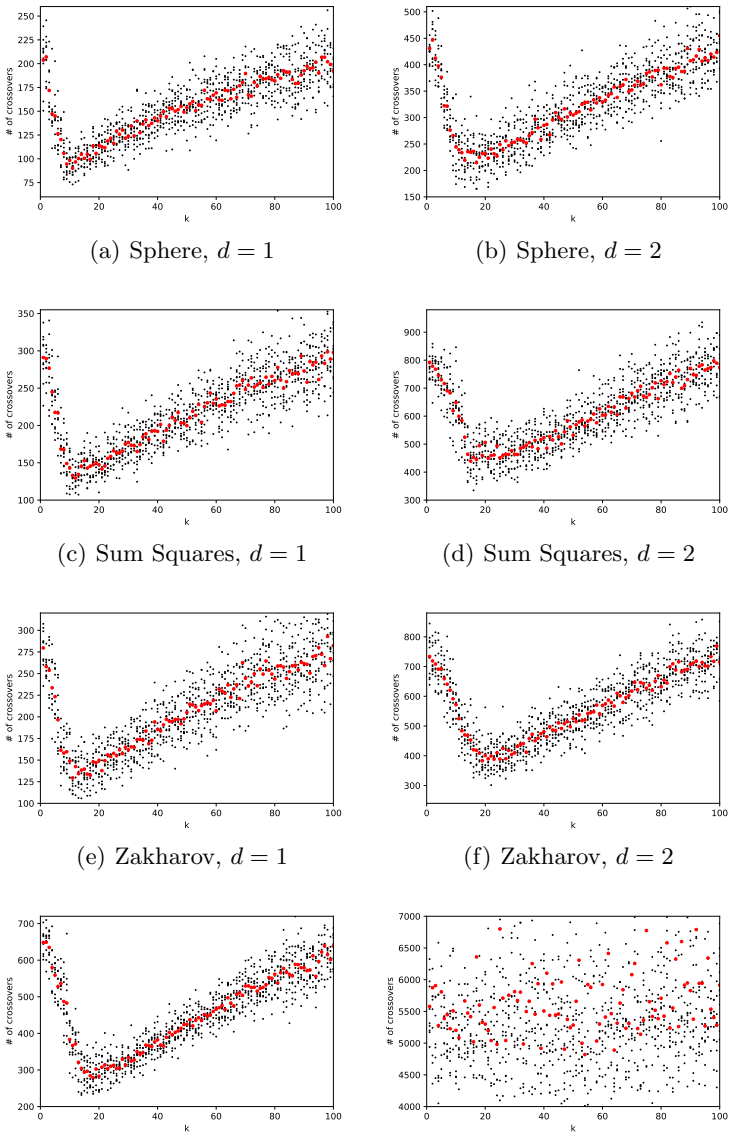
Name	Function	Domain
Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + \exp(1)$	$[-5.12, 5.12]$
Rastrigin	$10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]$
Schwefel	$418.9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_i\rvert})$	$[-500, 500]$
Griewank	$\sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$

Table 3. cGA's configuration.

Ring length	100
Boundary conditions	Closed
Neighborhood	East-West
Crossover operator	Blending method
Mutation operator	Uniform random
Mutation probability	0.1
Replacement criterion	Improvement

1 and 2. The reported values are the average number of crossovers performed by the cGA. For each value of k, the cGA was executed 300 times; the stop condition being to find an optimal solution. The black points represent the average of 30 different executions. So, for each value of k, 10 black points are reported. The red point represents the average of these 10 black points. From Figure 2, observe that all uni-modal functions with $d = 1$ tend to be solved faster for values of k around 10 and 20. Intuitively, this makes sense because uni-modal functions are relatively easy to solve. Thus, having more exploitation than exploration elements should improve the convergence time to optimal solutions. So, kFLS with small values of k has more exploitation elements because it takes advantage of the previously explored cells more quickly. Now, for $d = 2$, the value of k that improves the convergence time to optimal solutions is slightly greater. This makes sense too, because for $d = 2$ the functions are more difficult to solve. Thus, they should require less exploitation. Nevertheless, notice that the results from Figure 2(h) suggest that there might be functions such that there is no difference regarding the value of k.

Regarding multi-modal functions in Figure 3, the observations are similar to those obtained for the uni-modal functions. Namely, for $d = 1$, a greater value of k, around 20 and 40, helps improve the cGA's running time. Now, for $d = 2$, it is not easy to infer any improvement using intermediate values of k. From



(a) Sphere, $d = 1$ (b) Sphere, $d = 2$

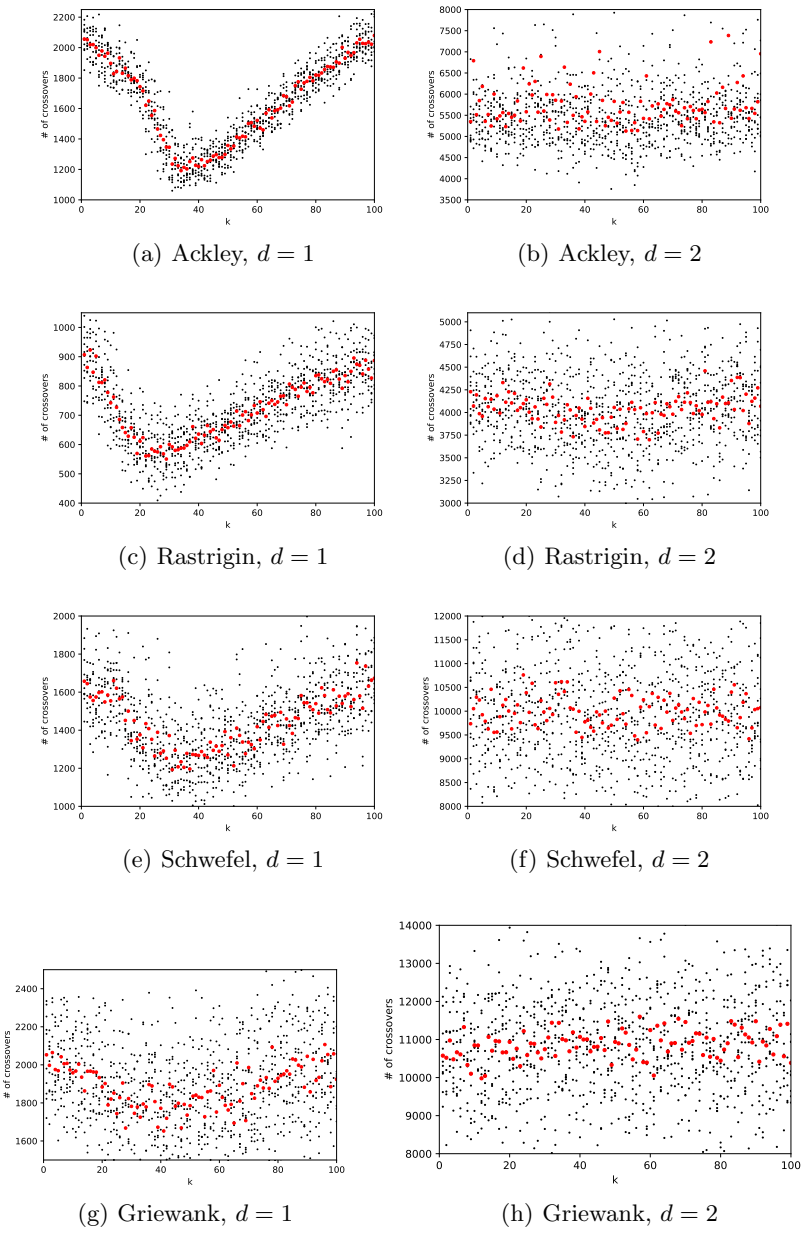
(c) Sum Squares, $d = 1$ (d) Sum Squares, $d = 2$

(e) Zakharov, $d = 1$ (f) Zakharov, $d = 2$

(g) Rotated Hyper-Ellipsoid, $d = 1$ (h) Rotated Hyper-Ellipsoid, $d = 2$

Fig. 2. Average running time of the cGA over some uni-modal functions.

these last figures, we cannot make any strong conjecture. Thus, we left a more rigorous analysis as future work.



(a) Ackley, $d = 1$ (b) Ackley, $d = 2$

(c) Rastrigin, $d = 1$ (d) Rastrigin, $d = 2$

(e) Schwefel, $d = 1$ (f) Schwefel, $d = 2$

(g) Griewank, $d = 1$ (h) Griewank, $d = 2$

Fig. 3. Average running time of the cGA over some multi-modal functions.

4 Conclusions and Future Work

In this work, we attempted to explore the tradeoff between asynchronous and synchronous update policies in cGAs. To do so, we introduce an update policy,

named k-Fixed Line Sweep (kFLS). This policy was applied over a cGA for solving different single-objective benchmark continuous optimization functions. The results show that, for some cases, kFLS can accelerate convergence to near-optimal solutions if the value of k is appropriately adjusted. As a first approach, the performed empirical analysis may not be rigorous enough. Thus, in the future, we would like to perform a more robust and rigorous statistical analysis. Besides, we would like to explore the effect of different neighborhoods, topologies, and strategies, such as Cellular Memetic Algorithms (cMA). Finally, we are especially interested in performing an analysis of kFLS for $\mathcal{NP}$-hard combinatorial optimization problems.

References

1. Alba, E., Dorronsoro, B.: Cellular Genetic Algorithms, vol. 42. Springer Science & Business Media (2009)
2. Alba, E., Giacobini, M., Tomassini, M., Romero, S.: Comparing synchronous and asynchronous cellular genetic algorithms. In: International Conference on Parallel Problem Solving from Nature. pp. 601–610. Springer (2002)
3. Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms. John Wiley & Sons, 2 edn. (2004)
4. Luque, G., Alba, E., Dorronsoro, B.: An asynchronous parallel implementation of a cellular genetic algorithm for combinatorial optimization. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. p. 1395–1402. GECCO '09, Association for Computing Machinery, New York, NY, USA (2009), https://doi.org/10.1145/1569901.1570088
5. Nebro, A.J., Durillo, J.J., Luna, F., Dorronsoro, B., Alba, E.: Mocell: A cellular genetic algorithm for multiobjective optimization. International Journal of Intelligent Systems 24(7), 726–746 (2009), https://onlinelibrary.wiley.com/doi/abs/10.1002/int.20358
6. Zhang, B., Xu, L., Zhang, J.: A multi-objective cellular genetic algorithm for energy-oriented balancing and sequencing problem of mixed-model assembly line. Journal of Cleaner Production 244, 118845 (2020), https://www.sciencedirect.com/science/article/pii/S0959652619337151