Toward the Best Combination of Optimization with Fuzzy Systems to Obtain the Best Solution for the GA and PSO Algorithms Using Parallel Processing

Submitted: 20th December 2019; accepted: 30th March 2020

Fevrier Valdez, Yunkio Kawano, Patricia Melin

DOI: 10.14313/JAMRIS/1-2020/7

Abstract: *In general, this paper focuses on finding the best configuration for PSO and GA, using the different migration blocks, as well as the different sets of the fuzzy systems rules. To achieve this goal, two optimization algorithms were configured in parallel to be able to integrate a migration block that allow us to generate diversity within the subpopulations used in each algorithm, which are: the particle swarm optimization (PSO) and the genetic algorithm (GA). Dynamic parameter adjustment was also performed with a fuzzy system for the parameters within the PSO algorithm, which are the following: cognitive, social and inertial weight parameter. In the GA case, only the crossover parameter was modified.*

Keywords: *Genetic Algorithms, Particle Swarm Optimization, Fuzzy Logic, Parallel Processing*

1. Introduction

In this paper, we are dedicated to finding the best configuration that will help us obtain the minimum error for the pair of optimization algorithms that we use. We consider different sets of fuzzy system rules and also with the integration of different migration blocks to share information between the two algorithms.

Also, some benchmark functions are too complex, and they can take too long for the optimization algorithms, so we configure the algorithms to be used in parallel and thereby improve the runtime. This allows us to be able to use the migration blocks so that information is shared between each of the algorithms and to find the global minimum in less time.

The biggest problem with metaheuristic algorithms is that when they are searching there comes a time when they could stagnate and not reach the specific global minimum of each benchmark function. That is why we combine several strategies to avoid this situation.

As for the strategies to improve the methods, we refer to the dynamic adaptation of parameters for each algorithm, and also to the migration blocks that allow us to find a global minimum between the two algorithms (GA and PSO).

Previously, some other researchers have worked with the same optimization algorithms as us, but some of them have focus on the reduction of the runtime of the algorithms, others use algorithm with dynamic parameters adjustment with fuzzy systems. As in our previous paper in which we were working with the improvement of the performance of the same way to the algorithms of PSO and GA, but of individuals form each and with the use of processing inside of GPU (Graphics Processing Unit). In which we focus on reducing runtime [1].

In the current case in which we take care of finding the global minimum in less time with the integration of GPUs, we can find other people in the research community who work on similar research[2]–[5]. Also, some researchers have dedicated themselves to the use of fuzzy systems to find the parameters and others researchers are working with the use the others topologies of each algorithm to reach the best global minimum [6].

In comparison to the other mentioned researchers, the difference of our work is that we combine different strategies to find the best solution for each of the benchmark functions, such as: the parallel execution that allows us to share information with the migration blocks between the optimization algorithms [7]–[9].

Below a summary of each of the sections contained in the paper is presented. Within Section 2 there are three subsections which are part of the theoretical framework we use, such as concepts and pseudocode of the GA and PSO algorithms, and as a third subsection, our contribution is better described, which is, the joint version of the PSO and GA in parallel with the description of the different fuzzy systems and the migration blocks that were used. Section 3 shows the experiments we performed and the results obtained from them, shown separately from each of the two cases used where one focused on the use of fuzzy systems of one input and one output, as well as one input and two outputs. Finally, in Section 4, we find our conclusions based on the results obtained.

2. Theory of the Optimization Algorithms and Improvement Strategies Used

In this section you will see all the theory related to the optimization algorithms and strategies used where it will be divided by subsections for better understanding.

2.1. Genetic Algorithms

Once the pseudocode of the genetic algorithm is shown, Genetic algorithms (GA) are algorithms that are based on natural selection, where we can find the inherited characteristic of the parent chromosomes to the children, where the individuals are eliminating the weakest chromosomes according to the principles of Charles Darwin were the fittest survives. As in nature, the rivalry between people for scarce resources results in the fittest people ruling over the weakest [2], [4], [10].

In Figure 1 the pseudocode for the genetic algorithm is presented.

```
Begin algorithm
Initialize the population of random individuals
    For i=1 to TotalDimensions*30
        For 1 to MaxGenerations
            Assign fitness-value to entire population
            Select the best Solution
            Crossover
            Mutation
            Update population
        End for
    Save all Results in array.
    End for of i
Send Results to file (Runtime and all variables).
End algorithm
```

Fig. 1. Pseudocode of the Genetic Algorithm

The Figure 1, represent a cycle with an analysis of the process than can be parallelized and then can be implemented on the video card for improving runtime. The first process that is sent to the video card is "Assign fitness value to entire population" and these "Select the best solution". The code modification is very small, but with that, you can get an improvement in the runtime.

2.2. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is apopulation-basedstochasticoptimizationtechniquedevelopedbyEberhartandKennedyin1995, inspired by social behavior of bird flocking or fish schooling [7], [11]–[14].

PSO has many processes similar to those that work with genetic algorithms. This algorithm initiates a swarm of random particles, where each of the contained particles could be a solution to the problem that is being worked on. These possible solutions are evaluated in each of the iterations that we have.

Figure 2 shows the pseudocode of the PSO algorithm. [6], [14]–[16]:

```
Begin Algorithm
Initialize the positions and velocities of all particles
    For i=1 to TotalDimensions*30
        For 1 to MaxIterations|
            Compute Fitness value
            Update best local position
            Update best global position
            Update Velocity
            Update Position
        End for
    Save all Results in array.
    End for of i
Send Results to file (Runtime and all variables).
End algorithm
```

Fig. 2. Pseudocode of PSO algorithm

The following equation represents the update of the velocity vector of each particle $i\ i$ in each iteration k.

$$v_i^{k+1} = \omega \cdot v_i^k + \varnothing_1 \cdot rand_1 \cdot pBest_i - x_i^{k\Sigma} + \varnothing_2 \cdot rand_2 \cdot g_i - x_i^{k\Sigma}$$

A more detailed description of each factor is made below:

v_i^k = Is the velocity of the particle i in the iteration k.

ω = Is the inertial factor.

$\varnothing_1, \varnothing_2$ = The learning ratios (weights) that control the cognitive and social components.

$rand_1, rand_2$ = The random numbers between 0 and 1.

x_i^k = Current position of particle i in iteration k.

$pBest_i$ = Best position (solution) found by the particle i so far g_i represents the position of the particle with the *pBest_fitness* of the environment of g_i*(localbest)*. Eq. 2.

2.3. Parallel PSO and GA

The parallel PSO and GA is formed by the combination of the algorithms described above, to create a parallel version to be able to integrate a migration block, which allows us to generate diversity in the populations of each of algorithms. The algorithms share among them, either, the entire population of the best individuals. A fuzzy system is also proposed to dynamically adjust the PSO and some GA parameters [8], [12], [17], [18].

Although we don't use graphics cards in our algorithm, certain articles helped us to know what parameters of the algorithm we can use to share information in parallel with another algorithm.

Figure 3 illustrates the flow of information in proposed method.

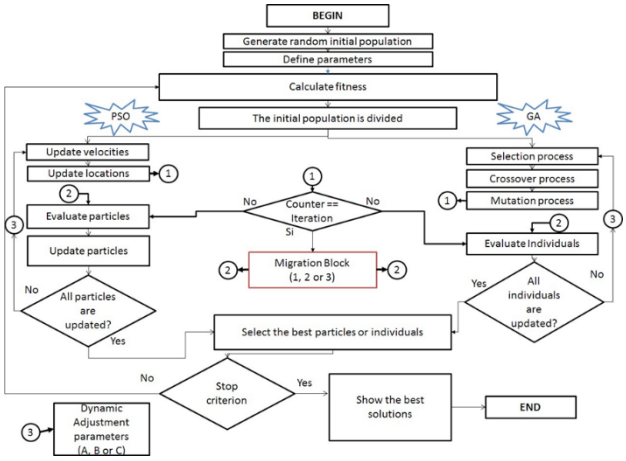

Fig. 3. Parallel PSO and GA

Figure 3 shows the flow chart that is used to obtain the results of our experiments. Which is composed of a block in which the main parameters are defined for each of the algorithms that are used, after having created the population then it is divided so that each of the algorithms works. From there each one of the algorithms begins to work simultaneously and each performs its respective processes. We can see in the center of Figure 3 that there is a migration block which refers to each of the following blocks (Figures 4, 5 and 6). On each of the sides of the diagram the union point number 3 is observed, which joins the block that is in the lower-left part of Figure 3 in which the dynamic adjustment of parameters is made according to the fuzzy systems which are shown in Figures 7, 8, 9, 10 and 11 [16], [19]–[23].

2.4. Migration Blocks

The migration blocks are more focused on improving the results obtained by the genetic algorithm, which migrates individuals between the two algorithms to add more diversity within each population [24].

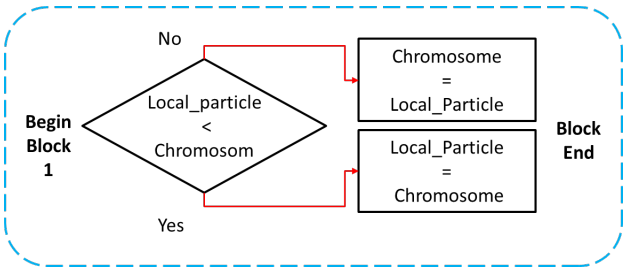

Fig. 4. Migration block number 1

In Figure 4 the migration block number 1 is shown in which the comparison is made among all the PSO particles and all GA individuals, in the case that the individuals are greater than the particles a population change is made between the algorithms.

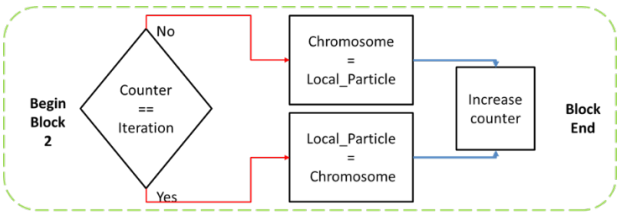

Fig. 5. Migration block number 2

In Figure 5 the migration block number 2 is shown in which the same population change is made as the previous block, but now it is activated every certain number of iterations.

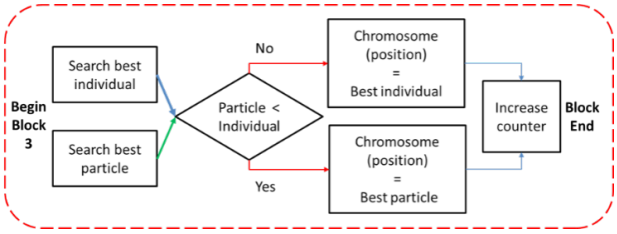

Fig. 6. Migration block number 3

Finally, in Figure 6 is observed that the best solutions are found for each of the populations, these are compared and only the best ones are exchanged.

Only the parameters of the PSO variables were adjusted, which are the cognitive, social and the inertial weight parameters. This is why in each of the fuzzy systems shown below, only the rules of each one change.

2.5. Dynamic Adjustment of Parameters With Fuzzy Systems

CASE 1 WITH FUZZY SYSTEMS OF ONE INPUT AND ONE OUTPUT:

The fuzzy systems used for dynamic adjustment are the following models.

Fuzzy System 1_Cognitive		
	Input= Iterations	Output= Cognitive Parameter
Membership Functions	Low	Low
	Medium	Medium
	High	High

Fuzzy System 2_Social		
	Input= Iterations	Output= Social Parameter
Membership Functions	Low	Low
	Medium	Medium
	High	High

Fuzzy System 3_Inertial		
	Input= Iterations	Output= Inertial Parameter
Membership Functions	Low	High
	Medium	Medium
	High	Low

Fig. 7. Rules for each fuzzy systems type A

In Figure 7 the set of rules that are used in the fuzzy system type A is shown, in this case, it's the same membership functions for the cognitive and social parameters.

Fuzzy System 1_Cognitive		
	Input= Iterations	Output= Cognitive Parameter
Membership Functions	Low	High
	Medium	Medium
	High	Low

Fuzzy System 2_Social		
	Input= Iterations	Output= Social Parameter
Membership Functions	Low	Low
	Medium	Medium
	High	High

Fuzzy System 3_Inertial		
	Input= Iterations	Output= Inertial Parameter
Membership Functions	Low	High
	Medium	Medium
	High	Low

Fig. 8. Rules for fuzzy system type B

In Figure 8 the membership functions of the cognitive parameter are varied.

Fuzzy System 1_Cognitive		
	Input= Iterations	Output= Cognitive Parameter
Membership Functions	Low	Low
	Medium	Medium
	High	High

Fuzzy System 2_Social		
	Input= Iterations	Output= Social Parameter
Membership Functions	Low	High
	Medium	Medium
	High	Low

Fuzzy System 3_Inertial		
	Input= Iterations	Output= Inertial Parameter
Membership Functions	Low	High
	Medium	Medium
	High	Low

Fig. 9. Rules for fuzzy system type C

In Figure 7, the used rules are the same for all variables (cognitive, social and inertial weight parameters). In Figure 8, the rules change for the fuzzy systems of the cognitive parameter. Finally, in the case of Figure 9, it is similar to the previous one, but now the rules are changed for the social parameter.

CASE 2 WITH FUZZY SYSTEMS OF ONE INPUT AND TWO OUTPUTS:

The following fuzzy systems refer to those in case 2 of the experiments that are composed of one input and two outputs.

Fuzzy System D			
	Input= Iterations	Output 1= Cognitive Parameter	Output 2= Social Parameter
Membership Functions	Low	Low	Low
	Medium	Medium	Medium
	High	High	High

Fig. 10. Rules for fuzzy system type D

In Figure 10, each of the rules is like the rules used in Figure 7.

Fuzzy System E			
	Input= Iterations	Output 1= Cognitive Parameter	Output 2= Social Parameter
Membership Functions	Low	High	Low
	Medium	Medium	Medium
	High	Low	High

Fig. 11. Rules for fuzzy system type E

In Figure 11, each of the rules is like the rules used in Figure 8.

Fuzzy System F			
	Input= Iterations	Output 1= Cognitive Parameter	Output 2= Social Parameter
Membership Functions	Low	Low	High
	Medium	Medium	Medium
	High	High	Low

Fig. 12. Rules for the fuzzy system type F

In Figure 12, each of the rules is like the rules used in Figure 9.

In Figure 13, are presented the rules that contain the fuzzy system of one input and two outputs. The output variables are the cognitive and social parameters of the PSO algorithm. These are influenced by the input variable, which is the number of iterations.

Fuzzy System with nine rules				
	Rules	Input = Iterations	Output = Cognitive Parameter	Output = Social Parameter
Membership Functions	1	Low	Low	Low
	2	Medium	Medium	Medium
	3	High	High	High
	4	Low	Medium	Medium
	5	Low	High	High
	6	Medium	Low	Low
	7	Medium	High	High
	8	High	Low	Low
	9	High	Medium	Medium

Fig. 13. Nine rules for the fuzzy system

Fuzzy System with 27 rules				
	Rules	Input = Iterations	Output = Cognitive Parameter	Output = Social Parameter
Membership Functions	1	Low	Low	Medium
	2	Low	Low	High
	3	Low	Medium	Low
	4	Low	Medium	Medium
	5	Low	Medium	High
	6	Low	High	Low
	7	Low	High	Medium
	8	Low	High	High
	9	Medium	Low	Low
	10	Medium	Low	Medium
	11	Medium	Low	High
	12	Medium	Medium	Low
	13	Medium	Medium	Medium
	14	Medium	Medium	High
	15	Medium	High	Low
	16	Medium	High	Medium
	17	Medium	High	High
	18	High	Low	Low
	19	High	Low	Medium
	20	High	Low	High
	21	High	Medium	Low
	22	High	Medium	Medium
	23	High	Medium	High
	24	High	High	Low
	25	High	High	Medium
	26	High	High	High
	27	Low	Low	Medium

Fig. 14. Twenty-seven rules for the fuzzy system (it is all possible)

In Figure 14, in this case, all the possible rules that can be created for the fuzzy system of one input and two outputs with three membership functions are used.

Different number of rules was used to experiment if they influence the obtained results.

3. Experiments and Results

The computer on which the experiments were made has the following hardware components. It has an Intel Core i7-4770 to 3.4 GHz with 4 cores and 8 threads, and the amount of 16 gigabytes of memory RAM (Random Access Memory) to 1600 MHz.

The benchmark functions that were use are in the following figure.

Figure 15 shows the benchmark mathematical functions that were used for the experiments, the first column is the identification list of each the functions, on the other hand, all functions have the objective of reaching zero[25], [26].

The results shown below are made based on a combination of different fuzzy systems varying

the number of rules, as well as different migration blocks.

#	Benchmark Functions	Function Limits
1	De Jong's	$-5.12 \leq xi \leq 5.12$
2	Rotated Hyper-ellipsoid	$-65.536 \leq xi \leq 65.536$
3	Rosenbrock's	$-2.048 \leq xi \leq 2.048$
4	Rastrigin's	$-5.12 \leq xi \leq 5.12$
5	Schwefel's	$-500 \leq xi \leq 500$
6	Ackley's Path	$-32.768 \leq xi \leq 32.768$
7	Axis parallel hyper-ellipsoid	$-5.12 \leq xi \leq 5.12$
8	Griewangk's	$-600 \leq xi \leq 600$

Fig. 15. List of benchmark functions

Tab. 1. PSO and GA parameters used

PSO and GA parameters	
Population	100
Dimensions	5, 10, 20, 40, 80, 160, 320, 640, 1280
Iterations / Generations	1000
Cognitive parameter	Dynamic
Social parameter	Dynamic
Inertial weight	Dynamic
Percentage of Crossing	0.8
Assignment Fitness Value	Ranking
Selection	Universal Stochastic Sampling
Recombination	Multipoint crossing
Mutation	(0.7 / Chrome length)

3.1. Experiments for CASE 1 With One Input and One Output

This refers to the combination of the two optimization algorithms with dynamic adjustment of parameters with a fuzzy system for each of the variables. Each of the following figures is equivalent to a set of experiments of each benchmark functions used.

Figure 16 to 23 illustrate the results of the experiments for case 1 that includes a comparison between the different configurations of fuzzy systems (Figures 7, 8 and 9) as well as the different migration blocks (Figures 4, 5 and 6). Zero refers to the fact that it does not include a migration block.

In Figure 16 is observed that in the CASE-1-B-1, which the case of one input and one output is where the fuzzy system is of type B and the migration block is the number 1, better results were obtained for all the dimensions that were used, which are from 5 to 1280 dimensions.

It can be see that in the cases where the number 1 indicates that block 1 was used are those that obtain a lower value, especially case 1 with fuzzy system type B.

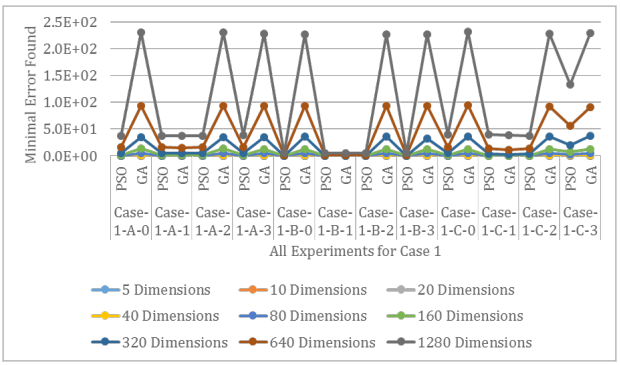

Fig. 16. Comparison of the results of the different combination used in Case 1: with the fuzzy systems A, B and C, as well as with the integration of migration blocks 0, 1, 2 and 3. These evaluating the benchmark function 1

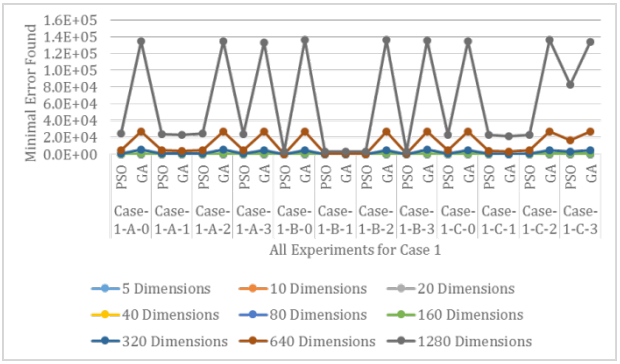

Fig. 17. Comparison of the results of the different combinations used in Case 1: with the fuzzy systems A, B and C, as well as with the integration of migration blocks 0, 1, 2, and 3. These evaluating the benchmark function 2

In the same way, Figure 17 shows that the fuzzy system type B with the migration block 1 is the winner for the benchmark function number 2 (Figure 15).

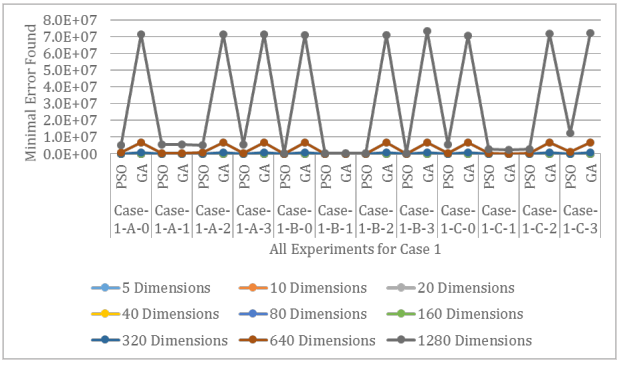

Fig. 18. Comparison of the results of the different combinations used in Case 1: with the fuzzy systems A, B and C, as well as with the integration of migration blocks 0, 1, 2 and 3. These evaluating the benchmark function 3

In Figure 18, the values found are extremely high, but as in the previous case type B is the winner, although it does not reach zero, it is less than the other ones in this figure. The highest value is around of 70,900,000 for GA and the lowest value is around 360,000 for PSO, it's a great difference this value is

when we working with 1280 dimensions and configuration "CASE-1-B-2". But if we are running the algorithm with other configuration as "CASE-1-B-1" we obtain a great difference with values around 2,900 where we working with 1280 dimensions.

In Figure 19, in other cases we saw that the combination with the migration block improved the results but for the benchmark function number 4, we could notice that this same migration block was beneficial for all the fuzzy systems used.

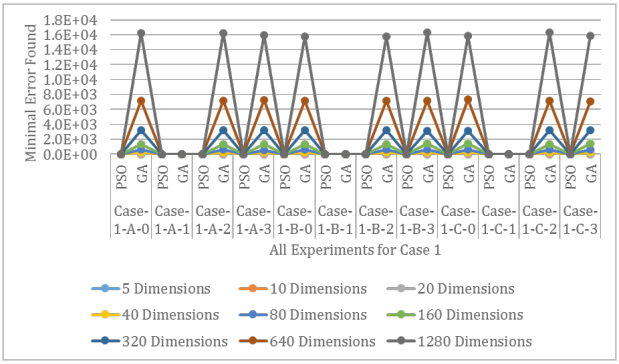

Fig. 19. Comparison of the results of the different combinations used in Case 1: with the fuzzy systems A, B and C, as well as with the integration blocks 0, 1, 2 and 3. These evaluating the benchmark function 4

In Figure 20, good results are observed in the same cases as the previous figure although the values close to zero are at low dimensions. The best values obtained in this case for the benchmark function 5 are around 1,300 for the "Case-1-B-1".

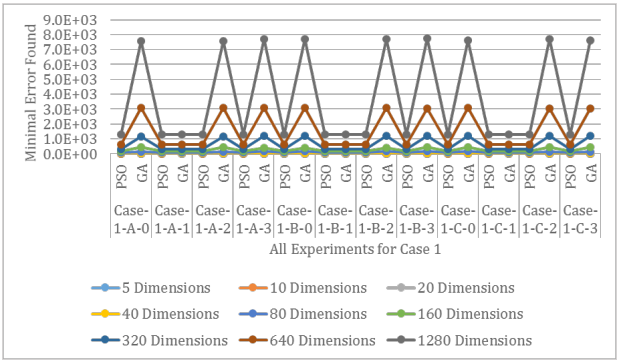

Fig. 20. Comparison of the results of the different combinations used in Case 1: with the fuzzy systems A, B and C, as well as with the integration blocks 0, 1, 2 and 3. These evaluating the benchmark function 5

For function 6, in Figure 21, good results are obtained with low dimensions because when we are working with 1,280 dimensions the results obtained are around 1,100.

In function 7 in Figure 22, it is a good result with "Case-1-B-1". The best solutions are between 0 and 0.5 for all dimensions used and this is due to the benchmark function.

In Figure 23, the minimum values is between 0 and 0.28 for function 8. We can note that the results using block 1 are the best.

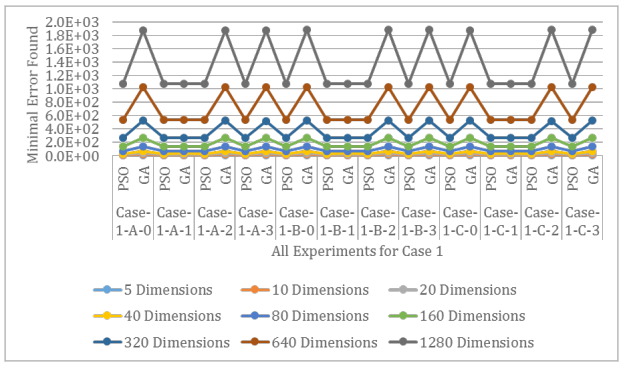

Fig. 21. Comparison of the results of different combination used in Case 1: with the fuzzy systems A, B and C, as well as with the integration of migration blocks 0, 1, 2 and 3. These evaluating the benchmark function 6

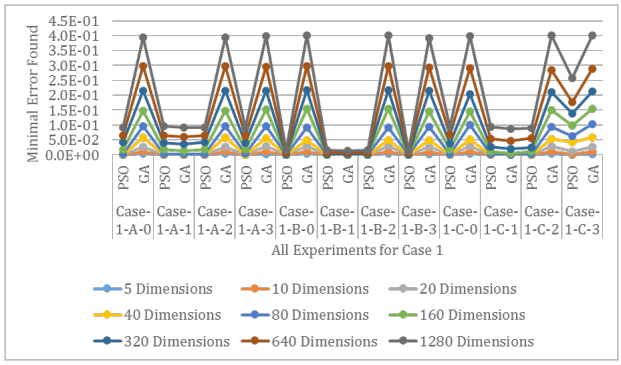

Fig. 22. Comparison of the results of different combination used in Case 1: with the fuzzy systems A, B and C, as well as with the integration of migration blocks 0, 1, 2 and 3. These evaluating the benchmark function 7

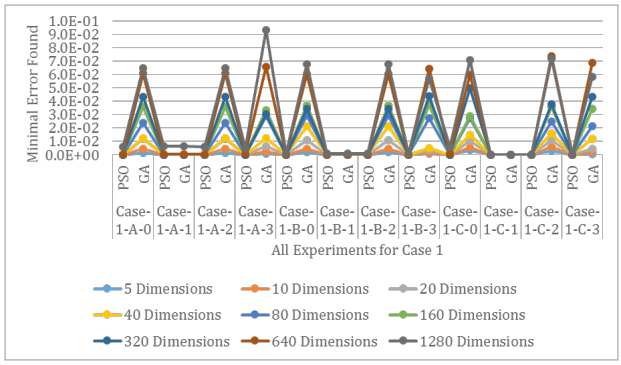

Fig. 23. Comparison of the results of different combination used in Case 1: with the fuzzy systems A, B and C, as well as with the integration of migration blocks 0, 1, 2 and 3. These evaluating the benchmark function 8

The following figures between Figure 24 to Figure 31 show the results of case 2 that focuses on the use of fuzzy systems of one input and two outputs.

3.2. Experiments for CASE 2 With One Input and Two Outputs

In the same way that in case 1 it can be observed that the use of the migration blocks is the one with the best results. In figure 24, the winning column is "Case-2-B-1" although it goes hand in hand with the "Case-2-B-0"

which is the one that does not integrate migration block. The best value is around 5 for these columns.

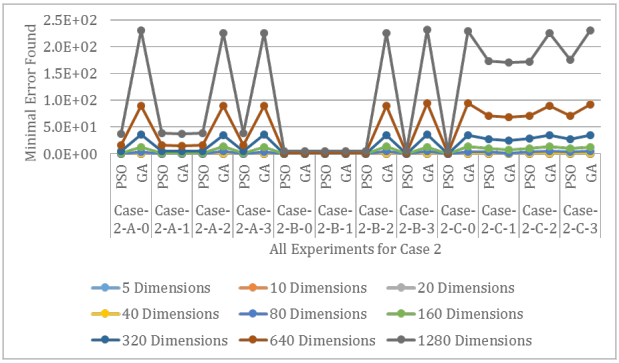

Fig. 24. Comparison of the results of the different combinations used in Case 2: with fuzzy systems A, B and C, as well as with the integration of migration blocks 0, 1, 2 and 3. These evaluation the benchmark function 1

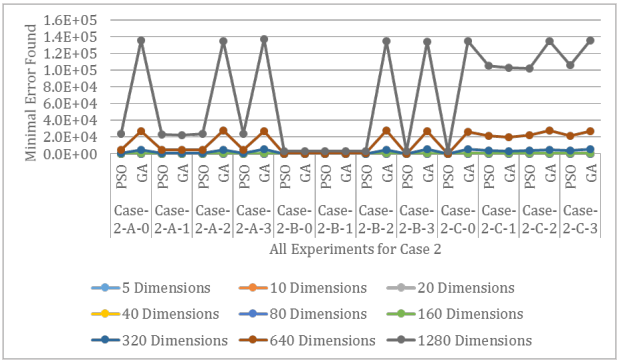

Fig. 25. Comparison of the results of the different combinations used in Case 2: with fuzzy systems A, B and C, as well as with the integration of migration blocks 0, 1, 2 and 3. These evaluations the benchmark function 2

In Figure 25, the comparison is made between all the versions of case 2, which shows as a result that the cases where fuzzy type B is used, with the use of migration block 1 as well as where no block is used, being block 1 the one that gets numbers closer to zero when we are working with low dimensions, but if we work with high dimensions the results go up to almost 3,000.

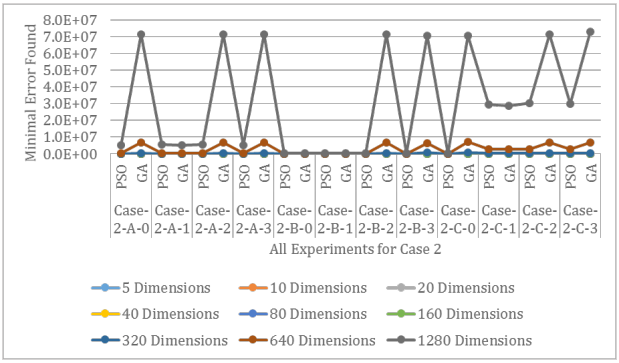

Fig. 26. Comparison of the results of the different combinations used in Case 2: with the fuzzy systems A, B, and C, as well as with the integration of migration blocks 0, 1, 2, and 3. These evaluating the benchmark function 3

In Figure 26, for the experiment of case 2 for the evaluation function 3 it is shown that in the same way as the previous cases the configuration with the case where it uses the migration block 1 is the closest to zero, arriving at the use of 80 dimensions.

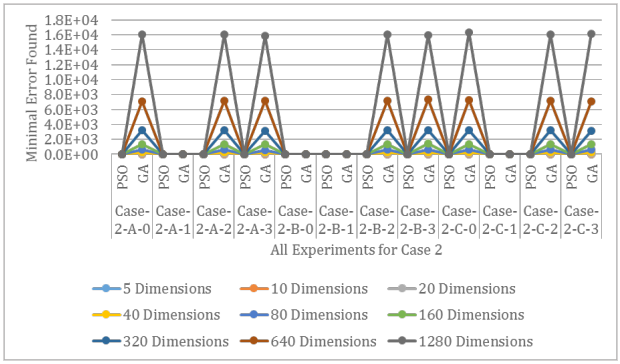

Fig. 27. Comparison of the results of the different combinations used in Case 2: with the fuzzy systems A, B, and C, as well as with the integration of migration blocks 0, 1, 2, and 3. These evaluating the benchmark function 4

Evaluating function number 4 in Figure 27, it is shown that in a greater number of configurations the objective of the function that is zero is reached. Although it is observed that in most of the cases is that it is gained by the use of the migration block 1.

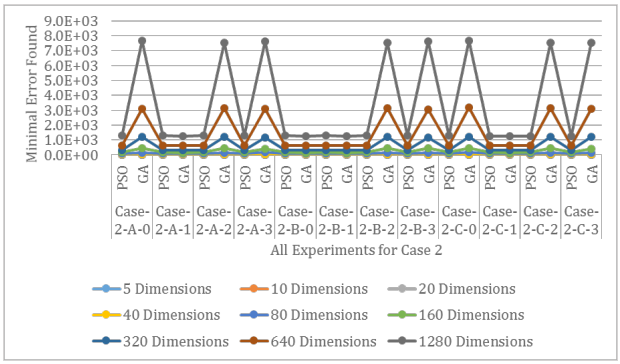

Fig. 28. Comparison of the results of the different combinations used in Case 2: with the fuzzy systems A, B, and C, as well as with the integration of migration blocks 0, 1, 2, and 3. These evaluating the benchmark function 5

Figure 28, in this evaluation of function 5, it was not possible to reach zero in any experiment, even though we were working with low dimensions although it was close to the minimum value.

In Figure 29 compared to Figure 28, worse results are observed which are large in value. Even when working up to 1280 dimensions are lower than the results observed in Figure 28.

In Figure 30, the results are very low but it is because it depends on the function with which you are working. At low dimensions, you can get some zeros in the values.

Figure 31, if we compare it with case 1 it can be seen that the values, in general, are a little lower depending of the number of dimensions that are being used.

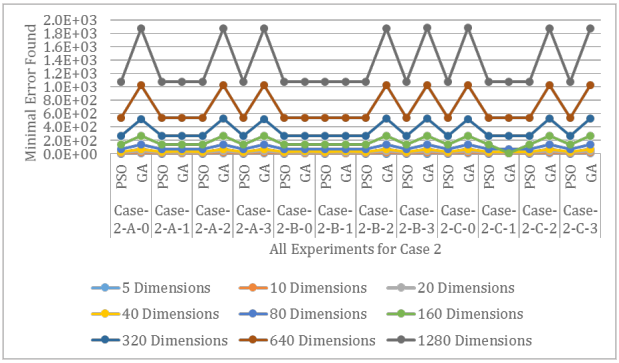

Fig. 29. Comparison of the results of the different combinations used in Case 2: with the fuzzy systems A, B, and C, as well as with the integration of migration blocks 0, 1, 2, and 3. These evaluating the benchmark function 6

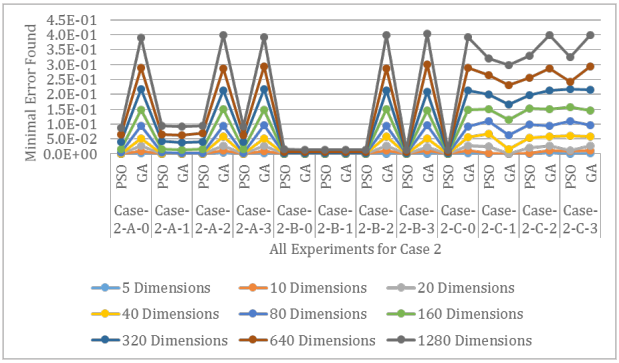

Fig. 30. Comparison of the results of the different combinations used in Case 2: with the fuzzy systems A, B, and C, as well as with the integration of migration blocks 0, 1, 2, and 3. These evaluating the benchmark function 7

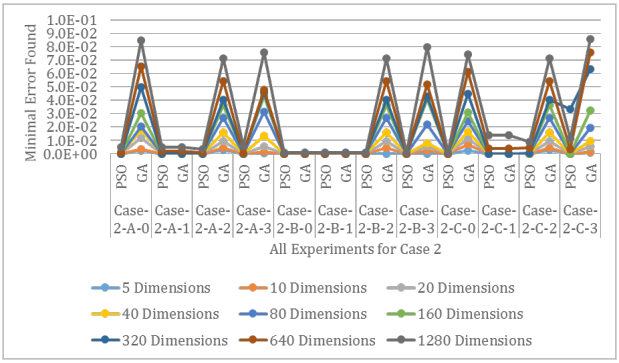

Fig. 31. Comparison of the results of the different combinations used in Case 2: with the fuzzy systems A, B, and C, as well as with the integration of migration blocks 0, 1, 2, and 3. These evaluating the benchmark function 8

4. Conclusion

As a general conclusion, after analyzing all the results of the performed experiments out, it can be stated that the use of the parallel algorithm combining PSO and GA with the integration of migration block 1 (Fig. 4) and the used of the fuzzy system type B (Fig. 8), is the most suitable for working at high dimensions.

In certain benchmark functions it was observed that the other types of fuzzy systems were good, but they cannot beat type B.

Regarding the comparison between cases 1 and case 2, the results show that there are very similar although on some situations case 2 wins case 1 and vice versa.

In the case of the fuzzy system of nine and twenty-seven rules, good results were not obtained, we believe that they are too many rules and saturates, giving as output a similar values for all the input modifications to the fuzzy.

Comparing Case 1 and 2 for the experiments, we observed that the global minimum values found are very similar although in some benchmark functions, the results obtained were better, but in other cases the other one won. In cases where nine and twenty-seven rules were used within the fuzzy system, they were not the best, we thought that membership functions were spliced and that is why they did not help and caused them to give the same output value regardless of the value of the input that was used.

As future work we can make use of Type-2fuzzy systems[27]–[32], as well as an algorithm that helps us optimize the rules of fuzzy systems so that they are the most appropriate and help us improve the local minimum found; although this will make the runtime of the experiment is higher, that is why it could also integrate the use of GPU to improve the performance of the parallel algorithm in general.

ACKNOWLEDGEMENTS

The authors would like to thanks CONACYT and Tijuana Institute of Technology for the support during this research work.

AUTHORS

Fevrier Valdez* – Division of Graduate Studies and Research, Tijuana Institute of Technology, Tijuana, Mexico, e-mail: fevrier@tectijuana.mx.

Yunkio Kawano – Division of Graduate Studies and Research, Tijuana Institute of Technology, Tijuana, Mexico, e-mail: monicoyunkio89@gmail.com.

Patricia Melin – Division of Graduate Studies and Research, Tijuana Institute of Technology, Tijuana, Mexico, e-mail: pmelin@tectijuana.mx.

* Corresponding author

REFERENCES

[1] Y. Kawano, F. Valdez and O. Castillo, "Performance Evaluation of Optimization Algorithms based on GPU using CUDA Architecture". In: 2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI), 2018, 1–6, DOI: 10.1109/LA-CCI.2018.8625236.

[2] G. R. Harik, F. G. Lobo and D. E. Goldberg, "The compact genetic algorithm", IEEE Transactions on Evolutionary Computation, vol. 3, no. 4, 1999, 287–297, DOI: 10.1109/4235.797971.

[3] X. H. Shi, Y. H. Lu, C. G. Zhou, H. P. Lee, W. Z. Lin and Y. C. Liang, "Hybrid evolutionary algorithms based on PSO and GA". In: The 2003 Congress on Evolutionary Computation, 2003. CEC '03, vol. 4, 2003, 2393–2399, DOI: 10.1109/CEC.2003.1299387.

[4] S. Debattisti, N. Marlat, L. Mussi, S. Cagnoni, "Implementation of a Simple Genetic Algorithm within the CUDA Architecture", GPUs for Genetic and Evolutionary Computation Competition at 2009 Genetic and Evolutionary Computation Conference, 2009.

[5] L. Mussi and S. Cagnoni, "Particle swarm optimization within the CUDA architecture", 2009.

[6] J. C. Vazquez and F. Valdez, "Fuzzy logic for dynamic adaptation in PSO with multiple topologies". In: 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013, 1197–1202, DOI: 10.1109/IFSA-NAFIPS.2013.6608571.

[7] F. Olivas, F. Valdez and O. Castillo, "Fuzzy Classification System Design Using PSO with Dynamic Parameter Adaptation Through Fuzzy Logic". In: O. Castillo and P. Melin (eds.), Fuzzy Logic Augmentation of Nature-Inspired Optimization Metaheuristics: Theory and Applications, 2015, 29–47, DOI: 10.1007/978-3-319-10960-2_2.

[8] F. Valdez, P. Melin and O. Castillo, "Fuzzy control of parameters to dynamically adapt the PSO and GA Algorithms". In: International Conference on Fuzzy Systems, 2010, 1–8, DOI: 10.1109/FUZZY.2010.5583934.

[9] F. Valdez, P. Melin and O. Castillo, "Fuzzy Logic for Combining Particle Swarm Optimization and Genetic Algorithms: Preliminary Results". In: A. H. Aguirre, R. M. Borja and C. A. R. Garciá (eds.), MICAI 2009: Advances in Artificial Intelligence, 2009, 444–453, DOI: 10.1007/978-3-642-05258-3_39.

[10] J. Carnahan and R. Sinha, "Nature's algorithms [genetic algorithms]", IEEE Potentials, vol. 20, no. 2, 2001, 21–24, DOI: 10.1109/45.954644.

[11] Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources". In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, 2001, 81–86, DOI: 10.1109/CEC.2001.934374.

[12] F. Olivas, F. Valdez and O. Castillo, "Particle swarm optimization with dynamic parameter adaptation using interval type-2 fuzzy logic for benchmark mathematical functions". In: 2013 World Congress on Nature and Biologically Inspired Computing, 2013, 36–40, DOI: 10.1109/NaBIC.2013.6617875.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization". In: Proceedings of ICNN'95 – International Conference on Neural Networks, vol. 4, 1995, 1942–1948, DOI: 10.1109/ICNN.1995.488968.

[14] R. Poli, J. Kennedy and T. Blackwell, "Particle swarm optimization", Swarm Intelligence, vol. 1, no. 1, 2007, 33–57, DOI: 10.1007/s11721-007-0002-0.

[15] F. Olivas, L. Amador-Angulo, J. Perez, C. Caraveo, F. Valdez and O. Castillo, "Comparative Study of Type-2 Fuzzy Particle Swarm, Bee Colony and Bat Algorithms in Optimization of Fuzzy Controllers", Algorithms, vol. 10, no. 3, 2017, DOI: 10.3390/a10030101.

[16] F. Valdez, P. Melin and O. Castillo, "Parallel Particle Swarm Optimization with Parameters Adaptation Using Fuzzy Logic". In: I. Batyrshin and M. G. Mendoza (eds.), Advances in Computational Intelligence, 2013, 374–385, DOI: 10.1007/978-3-642-37798-3_33.

[17] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization". In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), vol. 1, 2001, 101–106, DOI: 10.1109/CEC.2001.934377.

[18] J. Kaur, S. Singh and S. Singh, "Parallel Implementation of PSO Algorithm Using GPGPU". In: 2016 Second International Conference on Computational Intelligence Communication Technology (CICT), 2016, 155–159, DOI: 10.1109/CICT.2016.38.

[19] F. Valdez, P. Melin and O. Castillo, "An improved evolutionary method with fuzzy logic for combining Particle Swarm Optimization and Genetic Algorithms", Applied Soft Computing, vol. 11, no. 2, 2011, 2625–2632, DOI: 10.1016/j.asoc.2010.10.010.

[20] A. S. Radhamani and E. Baburaj, "Performance evaluation of parallel genetic and particle swarm optimization algorithms within the multicore architecture", International Journal of Computational Intelligence and Applications, vol. 13, no. 4, 2014, DOI: 10.1142/S1469026814500242.

[21] Z.-X. Wang and G. Ju, "A parallel genetic algorithm in multi-objective optimization". In: 2009 Chinese Control and Decision Conference, 2009, 3497–3501, DOI: 10.1109/CCDC.2009.5192490.

[22] Z. Dingxue, G. Zhihong and L. Xinzhi, "On Multipopulation Parallel Particle Swarm Optimization Algorithm". In: 2007 Chinese Control Conference, 2007, 763–765, DOI: 10.1109/CHICC.2006.4347299.

[23] X. Lai and G. Tan, "Studies on migration strategies of multiple population parallel particle swarm optimization". In: 2012 8th International Conference on Natural Computation, 2012, 798–802, 10.1109/ICNC.2012.6234614.

[24] H. Pohlheim, "Genetic and Evolutionary Algorithm Toolbox for Matlab". In: Evolutionäre Algorithmen, 2000, 157–170,
DOI: 10.1007/978-3-642-57137-4_6.

[25] J. G. Digalakis and K. G. Margaritis, "An Experimental Study of Benchmarking Functions for Genetic Algorithms," International Journal of Computer Mathematics, vol. 79, no. 4, 403–416, 2002,
DOI: 10.1080/00207160210939.

[26] "GEATbx: Example Functions (single and multiobjective functions) 2 Parametric Optimization". H. Pohlheim, http://www.geatbx.com/docu/fcnindex-01.html. Accessed on: 2020-05-28.

[27] E. Bernal, O. Castillo, J. Soria and F. Valdez, "Interval Type-2 fuzzy logic for dynamic parameter adjustment in the imperialist competitive algorithm". In: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2019, 1–5,
DOI: 10.1109/FUZZ-IEEE.2019.8858935.

[28] J. R. Castro, O. Castillo and P. Melin, "An Interval Type-2 Fuzzy Logic Toolbox for Control Applications". In: 2007 IEEE International Fuzzy Systems Conference, 2007, 1–6,
DOI: 10.1109/FUZZY.2007.4295341.

[29] R. Martinez, A. Rodriguez, O. Castillo and L. T. Aguilar, "Type-2 Fuzzy Logic Controllers Optimization Using Genetic Algoritms and Particle Swarm Optimization". In: 2010 IEEE International Conference on Granular Computing, 2010, 724–727,
DOI: 10.1109/GrC.2010.43.

[30] N. C. Long and P. Meesad, "Meta-heuristic algorithms applied to the optimization of type-1 and type 2 TSK fuzzy logic systems for sea water level prediction". In: 2013 IEEE 6th International Workshop on Computational Intelligence and Applications (IWCIA), 2013, 69–74,
DOI: 10.1109/IWCIA.2013.6624787.

[31] P. Melin, J. Urias, D. Solano, M. Soto, M. Lopez and O. Castillo, "Voice Recognition with Neural Networks, Type-2 Fuzzy Logic and Genetic Algorithms", Engineering Letters, vol. 13, no. 2, 2006.

[32] F. Gaxiola, P. Melin, F. Valdez, J. R. Castro and O. Castillo, "Optimization of type-2 fuzzy weights in backpropagation learning for neural networks using GAs and PSO", Applied Soft Computing, vol. 38, 2016, 860–871,
DOI: 10.1016/j.asoc.2015.10.027.