Cellular GAs with Active Components of PSO: Mutation and Crossover

A. Villagra[1], G. Leguizamón[2] and E. Alba[3]

[1] Universidad Nacional de la Patagonia Austral, Argentina
avillagra@uaco.unpa.edu.ar
[2] Universidad Nacional de San Luis, Argentina
legui@unsl.edu.ar
[3] Universidad de Málaga, España
eat@lcc.uma.es

Abstract. Two powerful metaheuristics being used successfully since their creation for the resolution of optimization problems are Cellular Genetic Algorithm (cGA) and Particle Swam Optimization (PSO). Over the last years, interest in hybrid metaheuristics has risen considerably in the field of optimization. Combinations of operators and metaheuristics have provided very powerful search techniques. In this work we incorporate active components of PSO into the cGA. We replace the mutation and the crossover operators by concepts inherited by PSO internal techniques. We present four hybrid algorithms and analyze their performance using a set of different problems. The results obtained are quite satisfactory in efficacy and efficiency.

1 Introduction

Optimization is important in all branches of engineering due to limited resources available. Evolutionary Algorithms (EAs) are very popular optimization techniques [2], [3]. They work by evolving a population of individuals (potential solutions), emulating the biological processes of selection, mutation, and recombination found in Nature, so that individuals (i.e., solutions) are improved. Most EAs in the classical literature are panmictic, although restricting the mating among individuals has appeared as an important research line. To this end, some kind of structure is added to panmictic (unrestricted) mating by defining neighborhoods among them. Among the many types of structured EAs (where the population is somehow decentralized), distributed and cellular algorithms are the most popular optimization tools [2], [4], [19]. A cGA, is a class of a decentralized population in which the tentative solutions evolve in overlapped neighborhoods [17],[19]. In a cGA individuals are conceptually set in a toroidal mesh, and are allowed to recombine with nearbyindividuals. The overlapping of neighborhoods provides to cGA an implicit slow diffusion mechanism.

Over the last years, interest in hybrid metaheuristics has risen considerably in the field of optimization [2]. The use of hybrid metaheuristics applied to combinatorial optimization problems received a continuously increasing attention in the literature. Hybrid approaches in fact usually seem both to combine complementary strengths and to overcome the drawbacks of single methods by embedding in them one or more steps based on different techniques.

In this work we intend to generate new functional and efficient hybrid algorithms in a methodological and structured way. In particular we use as the base core technique the cGA algorithm and apply "active components" of PSO in that cGA.

PSO was originally designed and introduced by Eberhart and Kennedy in 1995 [7], [10]. The PSO is a population based search algorithm inspired in the social behavior of birds, bees or a shoal of fishes. Each individual within the swarm is represented by a vector in multidimensional search space. It has been shown that this simple model can deal with difficult optimization problems efficiently.

The paper continues as follows. In Section 2 we show basic concepts of cGA and PSO. In Section 3 we present the hybrid algorithms proposed. In Section 4 we show the experimental tests and results, and finally in Section 5, we give some conclusions and analyze future research directions.

2 Basic Concepts

In this section we briefly describe the metaheuristic techniques used in this work: Cellular Genetic Algorithm and Particle Swarm Optimization.

2.1 Cellular Genetic Algorithm

Genetic Algorithms (GAs) are a particular class of EAs. In turn, cGAs are a subclass of Genetic Algorithms (GAs) with a spatially structured population, i.e. the individuals can only mate with their neighboring individuals [1]. These overlapped small neighborhoods help in exploring the search space because the induced slow diffusion of solutions through the population provides a kind of exploration, while the exploitation takes place inside each neighborhood by genetic operators. In cGAs the population is usually structured in a 2D toroidal grid. The most commonly used neighborhood is called L5 [19]. This neighborhood always contains five individuals: the considered one (position(x,y)) plus the North, East, West, and South individuals.

In Algorithm 1 we present the pseudo-code of a canonical cGA. It starts by generating and evaluating an initial population. After that, genetic operators (selection, recombination, mutation, and replacement) are iteratively applied to each individual until the termination condition is met. The population is structured in a two-dimensional (2-D) toroidal grid, and the neighborhood defined on it (line 6) contains five individuals. The considered individual itself is always selected for being one of the two parents (line 7). The second parent is selected by Tournament Selection (line 8). Genetic operators are applied to individuals in lines 10 and 12. We use in this chapter a two point crossover operator (DPX1)

Algorithm 1 Pseudocode of a cGA

```
1: /* Algorithm parameters in 'cga' */
2: Steps-Up(cga)
3: for $s \longleftarrow$ 1 to $MAX\_STEPS$ do
4:     for $x \longleftarrow$ 1 to $WIDTH$ do
5:         for $y \longleftarrow$ 1 to $HEIGHT$ do
6:             nList ⟵ ComputeNeigh (cga,position(x,y));
7:             parent1 ⟵ IndividualAt(cga,position(x,y));
8:             parent2 ⟵ LocalSelect(nList);
9:             /* Recombination */
10:            DPX1(cga.Pc,nList[parent1],nList[parent2],auxInd.chrom);
11:            /* Mutation */
12:            BitFlip(cga.Pm,auxInd.chrom);
13:            auxInd.fit ⟵ cga.Fit(Decode(auxInd.chrom));
14:            InsertNewInd(position(x,y),auxInd,[ if_not_worse ], cga, auxPop);
15:        end for
16:    end for
17:    cga.pop ⟵ auxPop;
18:    UpdateStatistics(cga)
19: end for
```

and traditional binary mutation operator - *bit-flip*.

2.2 Particle Swarm Optimization

The Particle Swarm Optimization was developed by Kennedy and Eberhart in 1995 [10]. This is a population-based technique inspired by social behavior of the movement of flocks of birds or schools of fish. In PSO the potential solutions, called particles, "fly" or "move" through the problem space by following some simple rules. All of the particles have fitness values based on their position and have velocities which direct the flight of the particles. PSO is initialized with a group of random particles (solutions), and then searches for optima by updating them through generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (according to fitness) that particle has found so far. This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value obtained so far by any particle in the population. This best value is a global best and thus it is called *gbest*.

Every particle updates its velocity and position with the following equations:

$$v_{n+1} = \omega_i v_n + \underbrace{\varphi_1 * rand * (pbest_n - x_n)}_{cognitive} + \underbrace{\varphi_2 * rand * (gbest_n - x_n)}_{social} \quad (1)$$

$$x_{n+1} = x_n + v_{n+1} \quad (2)$$

ω_i is the inertia coefficient which avoid big fluctuations over time; v_n is the particle velocity; x_n is the current particle position in the search space; $pbest_n$ and $gbest_n$ are defined as the "personal" best and global best seen so far; $rand$ is a random number between (0,1); and φ_1, φ_2 are learning factors.

It is important to highlight in Equation 1 that the second term represents what the particle itself has learned, and it is sometimes referred to as the "cognitive" term. The cognitive component in the velocity equation represents the best performance of the particle so far. The third term, sometimes referred as "social term" represents the group best solution so far.

The velocities of the particles are defined in terms of probabilities that a bit will change to one. Using this definition, a velocity must be a binary vector. So a transformation is used to map all real valued numbers of velocity to the range $[0,1]$ [11]. The normalization function used here is a sigmoid function s:

$$s(v_n^j) = \frac{1}{1 + exp(-v_n^j)} \tag{3}$$

where j represents the $j-th$ component of the velocity.
Also, the Equation 1 is used to update the velocity vector of the particle. And the new position of the particle is obtained using Equation 4:

$$x_{n+1}^j = \begin{cases} 1 & \text{if } \; r < s(v_n^j) \\ 0 & \quad \text{otherwise} \end{cases} \tag{4}$$

where r is a random number in the range $[0,1]$.

3 Our Proposed Algorithms

Our algorithms will have a general structure as a cGA and then mutation and crossover will be performed based in operations found in PSO.

We define the active components of a particular metaheuristic as those components that characterize or define its behavior with respect to the manner in which performs the search space. For example, in PSO a basic component is represented by the use of social and personal knowledge of the particles. Or, in ACO the pheromone structure that modified through a process of feedback to bias the exploration, can be considered an active component. From this concept of active component, it is possible to design hybrid metaheuristics that are formed by a host algorithm (i.e, a canonical version) and the incorporation of another active component/s of other metaheuristic/s that would allow improve from some perspective, the performance of the core metaheuristic.

To incorporate active components from PSO we maintain information about cognitive and social factors during the execution of a regular synchronous cGA with the intention of improving its performance.

In this work we propose four algorithms called hyCPM-local, hyCPM-global, hyCPR-local and hyCPR-global. In all algorithms we will treat each individual as a particle. We maintain its velocity, position and information about its personal (*pbest*), and social (*gbest*) knowledge to update the information (velocity and position). For the first two algorithms (hyCPM-local and hyCPM-global) a mutation based on PSO is used and line 12 (mutation in the canonical cGA Algorithm 1) is replaced with the following lines:

```
1: UpdateVelocity;
2: UpdateIndividual (cga.Pm, auxInd.chrom);
```

The first line updates the velocity of the particle using Equation 1. The second line modifies the individual taking into account the mechanism with the sigmoid function using Equation 4.

Both algorithms will apply this mutation based on PSO, with the difference that hyCPM-local uses the local neighborhood (Linear5), and then selects one neighbor from there as *gbest*. For hyCPM-global the global optimum of the all population is used as *gbest*.

For the last two algorithms (hyCPR-local and hyCPR-global) the concepts of PSO replace the recombination operator. The same mechanism as described above is used. The mutation operation is maintained without modification an in the canonical cGA and line 10 (recombination in the canonical cGA Algorithm 1) is replaced with the following lines:

```
1: UpdateVelocity;
2: UpdateIndividual (cga.Pc, auxInd.chrom);
```

4 Experiments and Analysis of Results

In this section we present the set of problems chosen for this study. We have chosen a representative set of problems to better study our proposal. The benchmarks contains many different interesting features in optimization, such as epistasis, multimodality, and deceptiveness. The problems used are Massively Multimodal Deceptive Problem (MMDP) [8], Frequency Modulation Sounds (FMS) [18] , Multimodal Problem Generator (P-PEAKS) [9], COUNTSAT [6] (an instance of MAXSAT [14]), Error Correcting Code Design (ECC)[13], and Maximum Cut of a Graph (MAXCUT) [12]; The minimum tardy task problem (MTTP)[16]. Finally the OneMax Problem [15](or BitCounting).

The common parameterization used for all algorithms is described in Table 1, where L is the length of the string representing the chromosome of the individuals. One parent is always the individual itself while the other one is obtained by using *Tournament Selection* (TS). The two parents are forced to be different in the same neighborhood.
In the recombination operator, we obtain just one offspring from two parents.

Table 1. Parameterization used in our algorithms.

Population Size	400 individuals
Selection of Parents	itself + Tournament Selection
Recombination	DPX1, $p_c = 1.0$
Bit Mutation	(Bit-flip for cGA), $p_m = 1/L$
Replacement	Replace If Not Worse
Inertia coefficient	w = 1
Leaning factors	$\varphi_1, \varphi_2 = 1$
Random value	$rand = UN(0,1)$

The *DPX1* recombination is always applied (probability $p_c = 1.0$). The bit mutation probability is set to $p_m = 1/L$. The exceptions are COUNTSAT, where we use $p_m = (L-1)/L$ and the FMS problem, for which a value of $p_m = 1/(2*L)$ is used. These two values are needed because the algorithms had a negligible solution rate with the standard $p_m = 1/L$ probability in our preliminary set of experiments. We here measure hit rate as the number of experiments. These parameter values have been the best values found in previous work with cGA [5].

Table 2. Percentage of Success obtained by hyCPM-local, hyCPM-global, hyCPR-local, hyCPR-global and cGA for a set of problems

Problem	hyCPM-local	hyCPM-global	hyCPR-local	hyCPR-global	cGA
ECC	**100%**	**100%**	**100%**	**100%**	**100%**
P-PEAKS	**100%**	**100%**	**100%**	**100%**	**100%**
MMDP	58%	**61%**	**100%**	**100%**	54%
FMS	**83%**	81%	0%	0%	25%
COUNTSAT	**80%**	36%	**100%**	**100%**	0%
"cut20.01"	**100%**	**100%**	**100%**	**100%**	**100%**
"cut20.09"	**100%**	**100%**	**100%**	**100%**	**100%**
"cut100"	38%	**48%**	**100%**	**100%**	45%
"mttp20"	**100%**	**100%**	**100%**	**100%**	**100%**
"mttp100"	**100%**	**100%**	0%	0%	**100%**
"mttp200"	**100%**	**100%**	0%	0%	**100%**
OneMax	**100%**	**100%**	0%	0%	**100%**
Average	**88%**	86%	67 %	67%	77%

We will replace the considered individual on each generation by the newly created individual in the same neighborhood only if the offspring fitness is not worse than the selected individual. The cost of solving a problem is analyzed by measuring the number of evaluations of the objective function made during the search. The stop condition for all algorithms is to find an optimum solution or to achieve a maximum of one millon function evaluations. The last three rows of Table 1 represent the values used only for the algorithms based on PSO. Throughout the paper all best values are **bolded**.
All algorithms are implemented in Java, and run on a 2.53 GHz Intel i5 processor under Windows 7.

In Table 2 we show the percentage of success in 150 independent runs for the five algorithms. We can observe that the success rate for our hybrids is higher (or equal in some cases) than for cGA algorithm. The average success rate for hyCPM-local (88%) and hyCPM-global (86%) are better than the average success rate for cGA (77%). Moreover, cGA obtained a very undesirable (0%) hit rate for the COUNTSAT problem. Also two of our hybrids (hyCPR-local and hyCPR-global) obtained a very undesirable (0%) hit rate for last four problems (FMS, "mttp100", "mttp200" and OneMax), meanwhile the other two hybrids (hyCPM-local and hyCPM-global) have always found success. As regards the average success rate HyCP-local obtains the highest value.

To analyze the selected variables number of evaluations (Evals) and time (Time) we grouped the results in three tables taking into account the percentage of success of each algorithms.

Table 3 shows the results of the five algorithms that obtained 100% of success for ECC, P-PEAKS, "cut20.01", "cut20.09" and "mttp20" and the following information is shown. The first column (Problem) represents the name of the problem resolved, the second column (Best) has the better found solution and then for each algorithm (hyCPM-local, hyCPM-global, hyCPR-local, hyCPR-global and cGA) the number of evaluations (columns Evals) needed to solve each problem, and the time in ms consumed (columns Time). Finally, the last column (ANOVA|K-W) represents the *p*-values computed by performing ANOVA or Kruskal-Wallis tests as appropriate, on the time and evaluations results, in order to assess the statistical significance of them (columns Evals and Time). We will consider a 0.05 level of significance. Statistical significant differences among the algorithms are shown with symbols "(+)", while non-significance is shown with "(-)".

Table 3. Results obtained by cGA and our hybrid algorithms for a set of problems

		hyCPM-local		hyCPM-global		hyCPR-local		hyCPR-global		cGA		ANOVA|K-W	
Problem	Best	Evals	Time	Evals	Time	Evals	Time	Evals	Time	Evals	Time	Evals	Time
ECC	0.07	**141400**	3369	157600	4370	429000	26596	417200	24341	150000	**2512**	(+)	(+)
P-PEAKS	1.00	39600	**3126**	**38200**	3376	176600	33661	176000	33128	39200	3283	(+)	(+)
"cut20.01"	10.12	**4800**	31	5200	33	8000	117	6400	78	5200	**26**	(+)	(+)
"cut20.09"	56.74	7600	**41**	**7000**	51	13800	172	13400	157	8000	49	(+)	(+)
"mttp20"	0.0244	4800	31	**4600**	**27**	10200	117	11200	125	5600	28	(+)	(+)

We can observe that our hybrid algorithms reduce the number of evaluations required to reach the optimum value in all problems and also these differences are statisticaly significant. Meanwhile, for the time required to obtain the optimum in general, cGA obtained the minimum values being slightly faster than the hybrids, who have a slight overhead compared to the canonical algorithm. This is an expected behavior since the mutation based on PSO requires some additional calculations to keep updated the particles, and to be able of using individual and social knowledge as appropriate. It should be noted that there are differences between the two algorithms (hyCPM-local and hyCPM-global) that use concepts of PSO in the mutation and CGA with respect to the two algorithms (hyCPR-local and hyCPR-global) that use the same concept replacing recombination. This is clearly shown in Figure 1. We chose the first problem (ECC) in representation of the other problems since the behavior is similar.

Figure 1 shows the number of evaluations needed for each algorithm to reach the optimum value in ECC problem. We can observe the median values and how the results are distributed. The minimum values are obtained by hyCPM-local; nevertheless, the difference among the results are not statically significant between hyCPM-global and cGA.

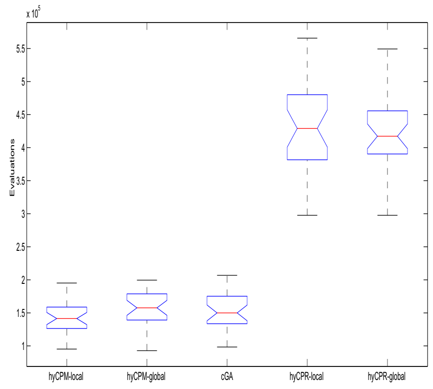

Fig. 1. Box-plots of the number of evaluations required for the algorithms hyCPM-local, hyCPM-global, cGA, hyCPR-local and hyCPR-global to solved ECC problem

Table 4 shows the results obtained by hyCPM-local, hyCPM-global and cGA for three problems because they obtained a 100% of success in all runs (see Table 2). The columns represent the same values as in Table 3. We can observe that our hybrids reduce the number of evaluations required to reach the optimum value in two out of three problems, nevertheless these differences are not statistically significant. The cGA was equally good for reducing the time.

Table 4. Results obtained by hyCPM-local, hyCPM-global and cGA for a set of problems

		hyCPM-local		hyCPM-global		cGA		ANOVA|K-W	
Problem	Best	Evals	Time	Evals	Time	Evals	Time	Evals	Time
"mttp100"	0.005	150200	1905	**141000**	1759	152800	**1084**	(-)	(-)
"mttp200"	0.0025	**440400**	10228	450800	10265	459600	**5487**	(-)	(+)
OneMax	500	199200	10236	228000	11683	**128200**	**3581**	(+)	(+)

Finally, Table 5 shows the results obtained by hyCPR-local and hyCPR-global for three problems where only these hybrids obtained 100% of success (see Table 2). The columns of the Table contain the same information as the previous Tables, the only difference in the last column (t-test) that represents the p-values computed by performing t-test as appropriate (with a normal distribution of the results), on the time and evaluations results, in order to assess the statistical significance of them (columns Evals and Time). We will consider a 0.05 level of significance. Statistical significant differences among the algorithms are shown with symbols "(+)", while non-significance is shown with "(-)".

In Table 5 we can observe that the minimum values are always obtained by hyCPR-global for both variables analyzed (number of evaluations and time, but the differences are not statically significant significant to the rest of algorithms and this is just an observed trend.

Table 5. Results obtained by hyCPR-local and hyCPR-global for a set of problems

		hyCPR-local		hyCPR-global		*t*-test	
Problem	Best	Evals	Time	Evals	Time	Evals	Time
MMDP	40	535600	37215	**527600**	**34152**	(-)	(-)
COUNTSAT	6860	13200	151	**12400**	**125**	(-)	(-)
"cut100"	1077	535600	37215	**527600**	**34152**	(-)	(-)

5 Conclusions

In this work we present four hybrid algorithms, called hyCPM-local, hyCPM-global, hyCPR-local and hyCPR-global. The motivation for this work was to improve the performance of a basic cGA with the addition of components that make efficient other metaheuristics, with the goal of getting even better results than those already obtained by the core technique.

In eleven out of twelve problems analyzed our hybrids obtained a 100% of success in obtaining the optimum. In nine out of the twelve problems analyzed the best performance in terms of the number of evaluations was also obtained by our hybrids. This means that our hybridization framework can effectively improve the efficiency of the basic cGA. Meanwhile, in all problems where the percentage of success in achieving the optimum is different, our hybrid algorithms obtained the highest success percentage (so they are also very accurate). Taking into account the time required to reach the optimum values, only in five of the twelve problems discussed our hybrids run for a faster execution. This behavior is also good, since the introduction of the PSO concepts into cGA also introduces more processing time and this affects the time required to reach the optimum values.

These results encourage us to expand the set of problems discussed in future works and to incorporate other active components from other metaheuristics.

Acknowledgments We acknowledge the constant support by the Universidad Nacional de la Patagonia Austral. The second author also acknowledges the constant support afforded by the Universidad Nacional de San Luis and the ANPCYT that finances his current researches. The third author acknowledges funds from the CICE of the Junta de Andalucia, under contract P07-TIC-03044 (DIRICOM http://diricom.lcc.uma.es) and Spanish Ministry of Sciences and Innovation (MICINN) and FEDER under contract TIN2011-28194 (RoadMe http://roadme.lcc.uma.es).

References

1. E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms.* Springer, 2008.
2. E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.
3. T. Bäck, D.B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation.* Oxford University Press, 1997.
4. E. Cantú-Paz. *Eficient and Accurate Parallel Genetic Algorithms*, volume 1 of *Book Series on Genetic Algorithms and Evolutionary Computation.* Kluwer Academic, 2nd edition, 2000.
5. B. Dorronsoro. *Diseño e Implementación de Algoritmos Genticos Celulares para Problemas Complejos.* PhD thesis, Universidad de Málaga, España, 2006.
6. S. Droste, T. Jansen, and I. Wegener. A natural and simple function which is hard for all evolutionary algorithms. In *3rd SEAL*, pages 2704–2709, 2000.
7. R. Eberhart and J. Kennedy. A new optimizer using particles swarm theory. In *Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway*, pages 39–43, 1995.
8. D. Goldberg, K. Deb, and J. Horn. Massive multimodality, deception, and genetic algorithms. In R. Männer and B. Manderick, editors, *Int. Conf. Parallel Prob. Solving from Nature II*, pages 37–46, 1992.
9. K. De Jong, M. Potter, and W. Spears. Using problem generators to explore the effects of epistasis. In *7th Int. Conf. Genetic Algorithms*, pages 338–345. Morgan Kaufmann, 1997.
10. J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *IEEE Int. Conf. Neural Netw.*, volume 4, pages 1942–1948, 1995.
11. J. Kennedy and R. Eberhart. A discrete binary version of the particle swarm algorithm. *A discrete binary version of the particle swarm algorithm*, 1997.
12. S. Khuri, T. Bäck, and J. Heitkötter. An evolutionary approach to combinatorial optimization problems. In *22nd annual ACM C.S. conf.*, pages 66–73, 1994.
13. F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, 1977.
14. C. Papadimitriou. *Computational Complexity.* Adison-Wesley, 1994.
15. J.D. Schaffer and L.J. Eshelman. On Crossover as an Evolutionary Viable Strategy. In R.K. Belew and L.B. Booker, editors, *Proceedings of the 4th ICGA*, pages 61–68. Morgan Kaufmann, 1991.
16. D. Stinson. *An Introduction to the Design and Analysis of Algorithms.* St. Pierre, Manitoba : The Charles Babbage Research Centre, 1985.
17. M. Tomassimi. The parallel genetic cellular automata: Application to global function optimization. In R.F. Albrecht, C.R. Reeves, and N.C. Steele, editors, *International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 385–391. Springer-Verlag, Heidelberg, 1993.
18. S. Tsutsui and Y. Fujimoto. Forking genetic algorithm with blocking and shrinking modes. In S. Forrest, editor, *5th ICGA*, pages 206–213, 1993.
19. D. Whitley. Cellular genetic algorithms. In S. Forrest, editor, *5th ICGA*, page 658. Morgan Kaufmann, 1993.