Cellular Genetic Algorithms: Understanding the Behavior of Using Neighborhoods

Carolina Salto[1] and Enrique Alba[2]

[1] Facultad de Ingeniería, Universidad Nacional de La Pampa, CONICET, Argentina

[2] Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Málaga, Spain,
e-mail: {saltoc}@ing.unlpam.edu.ar

Abstract. In this paper, we analyze the neighborhood effect in the selection of parents on an evolutionary algorithm. In this line, we compare a cellular genetic algorithm (cGA), which intrinsically uses the neighbor notion in the mating process, with a modified genetic algorithm including the concept of neighborhood in the selection of parents. Additionally, we analyze the neighborhood size considered for the selection of parent, trying to discover if a quasi-optimal size exists. All the analysis is carried out from a traditional analytic sense to a theoretical point of view regarding evolvability measures. The experimental results suggest that the neighbor effect is important in the performance of an evolutionary algorithm and could provide the cGA with higher chances of success in well-known optimization problems. Regarding the neighborhood size, there is an evidence that a range of neighbors of six, plus/minus two, individuals leads to the cGA to perform more efficiently than other considered sizes.

1 Introduction

Most EAs maintain panmictic populations (single population of individuals), in which there are no particular structures: genetic operators are applied on them as a whole. On the other hand, there exist structured EAs, in which the population is decentralized and any given individual interacts with a smaller set of individuals (denominated *neighborhood*). This neighborhood is smaller, sometimes much smaller, than the size of the population. There are two main ways for structuring the population in EAs, namely distributed EAs (dEAs), and cellular EAs (cEAs) (Alba and Dorronsoro, 2008b). In Figure 1 we show the typical population structures of an EA with centralized population or panmixia -Figure 1a)-, a descentralized EA -Figure 1b)-, and a cellular EA -Figure 1c). In this work we concentrate our attention to cEAs.

In cEAs the concept of (small) *neighborhood* is intensively used; this means that an individual may only interact with its nearby neighbors in the breeding loop. The overlapped small neighborhoods of cEAs help to explore the search space because the induced slow diffusion of solutions through the population provides a kind of exploration (diversification), while exploitation (intensification) takes

place inside each neighborhood by genetic operations Alba and Dorronsoro (2008a). In fact, adding neighborhoods to EAs take them mostly to the domain of swarm intelligence algorithms, where an emergent behavior comes out of the new configuration of the algorithm, much in the sense of other swarm techniques Sudholt (2017); Tomassini (2006).

If we think the population of an EA in terms of graphs Alba and Dorronsoro (2008b), being the individuals the vertices of the graph and their potential relationships the edges, a panmictic EA is a completely connected graph (see Figure 2a). On the other hand, a cEA is a lattice graph, as one individual can only interact with its nearest neighbors, for example in Figure 2b) each individual has eight neighbors.

The concept of neighborhood seems to play an important role. However, we actually do not have any clue on which size of the neighborhood is the best choice, or if the successful issue is the use of a neighborhood topology at all, or maybe the fact that only a few individuals are used. All these questions motivated us to explore in depth the benefits of using neighborhoods as well as to analyse different topology configurations. Therefore, the objective of this article is to discover whether a quasi-optimal arrangement of neighbors exists in the topology or if the use of a neighborhood is actually promoting a higher probability of escaping from local optima. Then, our initial hypothesis is that nontraditional neighborhood structures, where individuals can only interact with their closest neighbors in the population, may provide new essential information about the search process, hence leading the cEA to perform more efficiently than the use of traditional neighborhood configurations, for a number of well-known optimization problems.

For this purpose, we have designed a variation of a traditional cEA that uses a C9 (compact nine) neighborhood shape (one of the most commonly used neighborhoods Alba and Dorronsoro (2008a); Jie et al. (2017); Whitley (1993)), but having as a free parameter the number k of considered direct neighbors from which the second parent is selected (the first parent is always the center of the considered neighborhood). As the main loop of the cEA corresponds to a genetic algorithm (GA), we have denominated this algorithm as cGAk. The methodology devised in this work is targeted to characterize algorithms regarding the structure or conformation of the neighborhood and to generalize the number of neighbors used in a cGAk, and it can be summarized as follows. We evaluate a cGAk with all possible configurations of neighborhood sizes. We then will compare this algorithm against a panmictic GA incorporating the concept of neighbor for the parent selection with a free conformation of the neighborhood (at random) and a panmictic GA. The planned analyses and comparisons will help us to claim if the neighborhood configuration leads the algorithm to increase its success rate and to obtain good quality solutions in a lower number of evaluations. In order to get some meaningful conclusions, we present some studies from several angles between theory and experimentation, thus

giving a better insight into the internal behavior of each algorithm not only using static metrics but also from a dynamical point of view. This work extends the analysis carried out in Alba and Dorronsoro (2008a) where cellular GAs are compared to panmictic GAs and distributed GAs to show the effects of structuring a population, but how the neighborhood sizes affect the performance of a cGA was not considered in this preliminary work. Moreover, other previous studies lye on particle swarm optimization (PSO) and the number of informants García and Alba (2012, 2015). The intention of the present article is to analyze whether there is an effect in the performance of a cGA regarding the size of the neighborhood. So this is new: another metaheuristic is analyzed in order to determine if the observation on PSO also hold in cGAs.

The remainder of this article is structured as follows. Section 2 presents the cGAk and describes the modified GA to include the neighbor concept. Section 3 introduces the test problem and the parameterizations used in the experimentation. Section 4 presents and examines the results validating our proposal. Section 5 summarizes our conclusions and sketches our future work.

2 Proposals

In this section, a basic cGAk is described. The possibility of considering different neighborhood sizes enables us to generalize the number of neighbors, from 1 to $k-1$ (being k the size of the neighborhood shape without considering the central individual). Therefore, a number of different versions of cGAs can be generated, each one of them with neighborhoods containing k individuals at distance one of the current individual. These cGA variants will be later experimentally compared versus the genGAk by incorporating the idea of neighborhood in the selection of parents for mating. We will include in our study the well-known panmictic GA where individuals can mate with any other individual in the population. Last section discusses on genGAk, a hybrid step between a generational panmictic GA and a cellular GA. In genGAk we also use the concept of neighborhoods, but dynamically and randomly (uniformly) defined on a given individual every time that the evolutionary loop is computed.

2.1 The cGAk Algorithm

As previously mentioned, we will consider in this work a cGA with k neighbors (cGAk for short), where an individual may only interact with its direct neighbors at distance one in the breeding loop. The neighborhood shape used in this work is the C9 Alba and Dorronsoro (2008a) which contains the 8 nearest individuals to the considered one (in horizontal, vertical, and diagonal directions), as shown in Figure 3(a). The pseudocode of cGAk is introduced in Algorithm 1. As a traditional cGA, it starts by generating and evaluating an initial population. During the reproductive cycle, for each individual i (first parent) in the population, a set of k neighbours (K_i^t) randomly selected from C9

Algorithm 1 cGA with k neighbors (cGAk)

```
1: t = 0; {current evaluation}
2: initialize(Pop);
3: evaluate(Pop);
4: while (t < max_generations) do
5:    for (i=1, i < pop_size, i++) do
6:       K = select_neighboor(Pop[i], k); {random selection of neighborhood K with k individuals from C9, k in [2, ..., 8]}
7:       parent = select(K); {select the second parent by binary tournament selection}
8:       offspring = recombine(Pop[i], parent, p_c); {only one child is generated}
9:       offspring = mutate(offspring, p_m);
10:      Pop_aux[i] = replace(Pop[i],offspring); {select offspring if it is equal or better than Pop[i], in other case Pop[i]
         goes to next generation}
11:   end for
12:   Pop = Pop_aux;
13:   t = t + 1;
14: end while
15: return (best individual from Pop);
```

neighborhood shape without repetition is generated. The other parent is chosen from K_i^t according to binary tournament selection Miller and Goldberg (1995), which involves running a tournament between two individuals chosen at random from the K_i^t, being the winner the individual with the best fitness. After that, the variation operators (recombination and mutation) are applied to them, and the considered individual i is replaced by the recently created offspring if the new one represents a better solution than the considered individual. The already mentioned genetic operators (selection, recombination, mutation, and replacement) are iteratively applied to each individual until the termination condition is met. In Figure 3a) we can see how the reproductive cycle is applied in the neighborhood of an individual in the cGAk. Therefore, the principal difference with a canonical cGA lies in the conformation of the neighborhood K_i^t of each individual i. The presented cGAk corresponds to a synchronous cGA Alba (2005), as the individuals composing the population of the next generation are stored in an auxiliary population and, when completed, replace in an atomic step the current population. Therefore, in this model all the individuals in the population are updated simultaneously and, equivalently, the creation of individuals is made only from the individuals in the current population (not merging solutions created in different generations of the previous search).

2.2 The genGAk Algorithm

For comparison purposes, and also for analyzing how the neighborhood conformation and size affect the success of the GAs, we have propose in this work a new GA, as a means to restrict the interaction of the individuals to a subset of them in the reproductive cycle, similar as a cGA does. In general, they take the whole current population into account and the choices for mating purposes of a given individual always consider k individuals selected from all the rest of individuals, without restriction. We name this algorithm as genGAk. Indeed, we propose to analyze the behavior of the genGAk, where each individual in the population is selected to be parent, but for each of them, the second one is selected from a neighbor of k random (uniform) individuals of the population (Pop), by using binary tournament selection (see Figure 3b). Using this approach, for each individual i, and at each

Algorithm 2 Genetic Algorithm with k neighbors (genGAk)

```
1: t = 0; {current evaluation}
2: initialize(Pop);
3: evaluate(Pop);
4: while (t < max_generations) do
5:    for (i=1, i < pop_size, i++) do
6:       K = select_neighboor(Pop, k); {random selection of neighborhood K with k individuals}
7:       parent = select(K); {select the second parent by binary tournament selection}
8:       offspring = recombine(Pop[i], parent, p_c); {only one child is generated}
9:       offspring = mutate(offspring, p_m);
10:      Pop_aux[i] = replace(Pop[i],offspring); {select offspring if it is equal or better than Pop[i], in other case Pop[i]
         goes to next generation}
11:   end for
12:   Pop = Pop_aux;
13:   t = t + 1;
14: end while
15: return (best individual from Pop);
```

generation t, a different subset (K_i^t) with k individuals is generated. Formally, we can represent that subset as shown in Equation 1.

$$K_i^t = \{n_1, n_2, \ldots, n_K\} | K_i^t \subset Pop, \forall n_j, n_h \in K_i^t \;\; n_j \neq n_h \neq i \tag{1}$$

Algorithm 2 presents the pseudocode of genGAk to show the changes introduced to a traditional GA. Lines 6 to 8 show the significative differences with a traditional GA. The first step (Line 6) consists of the random selection of k individuals (from the whole population) to define the neighbor K_i^t for the individual i (first parent). After that, the second parent is selected from this neighbor K_i^t by using binary tournament selection (Line 7). Finally, the second parent and the individual i are recombined and only one offspring is generated. Therefore, genGAk has two principal differences with a traditional GA: the selection operator for mating does not work at population level and all individuals in the population participate in the mating loop as the first parent.

3 Experimental Setup

In this section we present the necessary information to reproduce the experiments that have been carried out in this article. First we will introduce the problems used to assess the performance of our proposals. Second, we will justify the parameters that the algorithms (cGAk and genGAk) will use.

3.1 Problems

In this section we present the set of problems chosen for our study. The benchmark is composed of combinatorial problems with many different features in the optimization domain, such as epistasis, multimodality, problem generators, or parameter fitting. This guarantees a high level of confidence in the results. The problems used are the following ones: OneMax Schaffer and Eshelman (1991) (or BitCounting) -instance with 500 bits-; P-PEAKS problem Jong et al. (1997), with 300 peaks; the Maximum Cut of a Graph (MAXCUT) problem Khuri et al. (1994) -instance with 20 vertices-; and

Minimum Tardy Task Problem (MTTP) Stinson (1985) -instance of 100 tasks-. Table 1 presents for each of them the following information: fitness function, number of variables (n), and optimal solution (a detailed description on these problems can be found in Alba and Dorronsoro (2008b)). All of them are maximization problems.

3.2 Parameters

In our experiments, the global pool of solutions of all the algorithms is set to 400 solutions. The tentative solutions for all the problems are encoded as binary strings. One of the two parents is the current individual i and the other one is selected from the neighborhood (K_i^t) using binary tournament. . The bit flip mutation is applied to the child with a rate (p_m) equals to $1/n$ (where n is the length of the solutions). The stop condition is to find the optimum fitness of each problem or to reach 10^6 evaluations. In the case of genGAk, proportional selection is used to build up the next population from the set of parent and offspring solutions. Specific parameters for the cGAk's are the C9 neighborhood, and a lattice shape of 20×20 individuals. Table 2 summarizes the parameters used in the experimentation. All parameter values are taken from Alba and Dorronsoro (2008a), where the justification is carried out. Notice that we are not using highly specialized cGAs and GAs, since our goal is not to outperform other algorithms, for the considered problems, but to analyze the influence of the different neighborhood sizes (k) in the behavior of the proposed algorithms. The k values vary from 2 to 8, but in the case of genGAk we also want to know the effect of increasing the k value from 50 (incremented by 50) until reaching the population size; when k=400 the genGAk behaves as a traditional GA.

The code was developed using JCELLAlba and Dorronsoro (2008b), a Java software library for working with cEAs and GAs. The considered hardware resource is an Intel I7 at 2.30 GHz, 6 GB of RAM, under Windows 7.

Due to the stochastic nature of the algorithms, we perform 50 independent runs to get reliable statistical results. We use the non-parametric test, since the resulting distributions could not follow the conditions of normality and homoskedasticity García et al. (2009). In particular, we have considered the application of the Friedman's ranking test, and used the Wilcoxon test as post-hoc procedure.

4 Results

In this section, we first present an analysis concerning the computational effort of the different neighborhood sizes (k) in cGAk and genGAk. Then, additional analyses concerning some evolvability measures (such as fitness distance analysis and escape probability) of each algorithm are performed.

4.1 Computational Effort

All the algorithms and for all neighborhood sizes were able to find the optimum value for the considered problems. Therefore, we concentrate the analysis in the numerical effort of each algorithm to locate that optimum value, by measuring the number of evaluations of the objective function made during the search. Figure 4 shows the distribution of the number of evaluations for each algorithm.

Regardless of the neighborhood size, cGAk is the fastest algorithm (minimum number of evaluations), meanwhile genGAk is the slowest one. A traditional GA (genGA400) remains in an intermediate position. Except for MAXCUT problem, in the rest of the problems genGAk with $50 \leq k \leq 350$ presents a slight decrease in the number of evaluations compared to genGA with k varying from 2 to 8. Moreover, cGA6 is the most promising cGAk variant: it obtains quickly the optimum (lower number of mean evaluations) in the majority of the problems. This observation is similar to the given one by Garcia et al. García and Alba (2012, 2015) in their study about PSO and the number of informant neighbors, where they show that a number of 6 ± 2 informant particles lead the algorithm to perform more accurately than other existing versions of it (the standard "two" and "all" PSO). Given that the p-value of Friedman test is lower than the level of significance considered $\alpha = 0.01$, we can estate that there are significant differences among the algorithms. Attending to this conclusion, Wilcoxon text, as a post-hoc statistical analysis, helps to determine which groups of behavior are emerging from the point of view of similar performance.

The Wilcoxon test generates groups of algorithms which are displayed in Figure 5 (in addition to assign letters to the different mean groups as usually does the test, we use a scale of grays in order to show more clearly the information). The algorithms grouped in the same column do not show significantly different results from each other. If overlapping of columns is not observed, then the algorithms have significant differences. A common observed behavior for all problems is that the cGAk algorithms belong to groups which present significant statistical differences with genGAk. The exception is the MAXCUT problem. Particularly, cGAk with $k = 6 \pm 2$ do not present significant differences between them, as they lie in the same group.

In the same line of reasoning, we can infer from these results that the neighborhood topology also has an important impact in the computational effort of the algorithm to solve the problems. There is a big difference between using a restricted neighborhood, represented by the cGAk, and using a panmictic EA incorporating the concept of neighbor for the parent selection (genGAk algorithm), where the second parent is selected from the whole population regardless of the distance in the population position of the first parent.

4.2 Fitness-Distance Analysis

In this section, we analyze an evolvability measure such as fitness-fitness clouds. Consequently, before starting with the discussion of the results, we describe it. Fitness-distance analysis quantifies the relation between the fitness of the individuals $f(x_i)$ in the landscape and its distances to the nearest global optimum x_{opt} Lu et al. (2011). Fitness distance correlation can also be visualized with the fitness distance plot, where the genotypic distance of a solution to the optimum is plotted against their fitness.

Figure 6 presents the fitness-distance plots for the algorithms with $k = 6$ (other k values present similar graphs). Due to the characteristics of the OneMax problem, the distance to the optimum is inversely proportional to the solution's fitness value. Consequently, the shape of the curves are lines, and no differences between the algorithms can be observed for this problem. For MAXCUT, no correlation can be observed between genotypic distance and fitness values, regardless of the algorithm used to solve the problem. Many solutions with good fitness values (near the 56.5) having different genotypic distances to the optimum (between 0 and 20) can be observed (upper points in the graph). This observation suggests that the algorithms to solve the MAXCUT may require stronger diversification mechanisms than for the rest of the problems. For MTTP and P-PEAKS, a very strong negative correlation can be observed. In the case of P-PEAKS, the right shapes like bags" have important differences in its heights and this is due to the good quality of the solutions sampled by cGA6. In other words, the difference in the bags indicates that the cGA6 sample many solutions with fitness near to the optimum (points near to 1.0) but with high genotypic distance to that optimum (average distances of 150). This happens because in many (hard) problems, optima of the same fitness have different genotypic representation (are located in different parts of the search space. Regarding MTTP, genGA6 presents a thickening in the bottom left part of the correlation shape, meaning that the algorithm samples more solutions not only with low fitness but also with low genotypic diversity respect to the optimum. All these observations indicate that the cGAk sampling behavior is better in the amount of solutions with good quality and close to the optimum value, endorsing the idea that the use of structured neighborhoods helps in the optimization process.

4.3 Escape Probability

An Escape Probability (ep) analysis studies the number of steps required to escape from a local optimum. It is defined as $P(f_i) = 1/S_i$, where S_i denotes the mean number of steps required to find an improving move starting in an individual with fitness value f_i. In our work, the escape probability for a fitness value f_i throughout the iteration process is computed as the average of the improving intervals (measured in number of evaluations) of each new individual with fitness value f_i. If the

escape probability is high for a particular fitness value, then it is easy to improve the fitness quality. Consequently, the escape probability $P(f_i)$ is a good indication of the degree of evolvability for individuals of fitness value f_i.

Figure 7 plots the Escape Probability (*ep*)for cGAk and genGAk (with k equals to 6 because similar graphs are obtained with the rest of k values) for all the problems. The horizontal axis shows the number of iterations of the algorithm variants, whereas the vertical axis presents the $P(f_i)$ value computed for the best solution found in iteration i of the algorithm. In all cases the cGA6 curve is shorter than the genGA6 one because cGA6 needs less number of iterations to locate the optimum (as explained in Section 4.1). The first evident conclusion is that, for all problems, cGAk has a higher *ep* value than genGAk throughout the evolution, indicating that the use of structured neighborhoods allows the algorithm to easily escape from local optimum. This observation is congruent with the results shown in Section 4.1: the cGAk obtains the optimum in less number of evaluations than genGAk for all the problems, and this is because cGAk gets trapped in local optima less frequently or for a less number of steps than genGAk (as evidenced by the high *ep* values of cGAk).).

Following analysis goes into detail of what happened with the *ep* values for the different configurations of neighborhoods for cGAk. Figure 8 plots the *ep* for all the problems, considering cGAk with k equal to 2, 4, 6, and 8. The first conclusion here is that all cGAk variants present high values of *ep* for OneMax problem at the beginning of the search, meaning that those k values provides the cGAk variants with high exploration ability, declining gradually to the end the evolution. In the case of MaxCut, the algorithms present a faster reduction in the *ep* values during the first stages of the search. After that the cGAk variants show a moderated *ep* progress, which means that the search is trapped in local basins (ie. it is costly to improve current solutions). It is important to note that, for this problem, the optimum is found very quickly as indicated in Section 4.1. P-PEAKS presents a similar situation to MaxCut but with low initial *ep* values. MTTP deserves a special consideration: cGAks present a fluctuation in the *ep* values from the middle of the search, which is more important at the end of the evolution. This indicates that the variants have the ability to escape from local minimal.

Anyway, the curves representing the different k values are overlapped in most of the evolution. Values of $k = 6 \pm 2$ present a slight higher *ep* values than the case of $k = 2$, indicating that the cGAk with $k = 6 \pm 2$ has higher chances to escape from local optima than genGA2. These results validate the ones obtained in Section 4.1: cGA2 needs more evaluations to obtain the optimum value than the rest of cGAk's.

In summary, we can detect some trends in the *ep* of the structured cGA cases, but still their behaviour is too similar each other in most problems: we probably should need a better metric to understand their internal differences.

5 Conclusions and Future Work

This article analyzes the benefits of incorporating the neighborhood concept to an EA, and how that notion affects to their success. For that purpose we considered cellular GAs (structured neighborhood), GAs incorporating the concept of neighborhood (non-structured neighborhood) and traditional panmictic GAs. Also, the influence of the number of neighbors in the performance of the algorithms were analyzed with the aim of generalizing the study to a wider spectrum of algorithms. A series of experiments and comparisons have been carried with several different combinatorial problems using metrics, such as fitness evaluations, and different evolvability measures.

In short, as to accuracy, the cGAk algorithms solved instances better than non structured ones, specially those with k=6. As to the computational effort (number of evaluations), cGAk performs a search in a faster way than the genGAk, and those differences are statistically significant for all the tackled problems. Consequently, the high exploration/exploitation capabilities of cGAk are clear, and that advantage may be attributed to the neighbor notion and their conformation.

Using a kind of probability of escape analysis, we have shown that solutions evolved by an EA using structured neighborhoods such as cGAk shows higher *ep* values than genGAk (EA using non-structured neighborhoods), which indicates that cGAk has the ability to evolve solutions escaping from local basins, a main hypothetical reason now visualized and confirmed on their behavior.

As future work, we will experiment with other combinatorial problems to extend the study presented in this work. This is important so as to better understand patterns of search based in internal metrics related to the search landscape. It is also clear that we need to add other metrics that better help to differentiate the behaviours in the different neighborhoods: most probably these metrics should account for such neighborhood peculiarities if we want a deeper understanding.

Acknowledgments

Dr. Salto acknowledge the UNLPam, the ANPCYT, CONICET and PICTO-UNLPam-0278 in Argentina from which receives regular support. The work of Prof. Alba has been funded by the Spanish project TIN2014-57341-R (moveON), University of Málaga, International Campus of Excellence, Andalucía Tech, Spain.

Bibliography

E. Alba. *Parallel Metaheuristics: A New Class of Algorithms.* Wiley, 2005.

E. Alba and B. Dorronsoro. On the effects of structuring the population. In *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces Series*, pages 37–46. Springer US, 2008a.

E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces Series.* Springer, 2008b.

J. García and E. Alba. Why six informants is optimal in PSO. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 25–32. ACM, 2012.

J. García and E. Alba. Hybrid PSO6 for hard continuous optimization. *Soft Computing*, 19(7): 1843–1861, 2015.

S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: A case study on the CEC2005 special session onreal parameter optimization. *Journal of Heuristics*, 15(6):617–644, 2009.

L. Jie, W. Liu, Z. Sun, and S. Teng. Hybrid fuzzy clustering methods based on improved self-adaptive cellular genetic algorithm and optimal-selection-based fuzzy c-means. *Neurocomputing*, 249:140 – 156, 2017.

K. De Jong, M. Potter, and W. Spears. Using problem generators to explore the effects of epistasis. In *Proceedings of the Seventh International Conference of Genetic Algorithms*, page 338345. Morgan Kaufmann, 1997.

S. Khuri, T. Bäck, and J. Heitkötter. An evolutionary approach to combinatorial optimization problems. In *Proceedings of the 22Nd Annual ACM Computer Science Conference on Scaling Up : Meeting the Challenge of Complexity in Real-world Computing Applications*, CSC '94, pages 66–73. ACM, 1994.

G. Lu, J. Li, and X. Yao. Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms. In Peter Merz and Jin-Kao Hao, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 6622 of *Lecture Notes in Computer Science*, pages 108–117. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20363-3.

B. Miller and D. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9:193–212, 1995.

J.D. Schaffer and L.J. Eshelman. On Crossover as an Evolutionarily Viable Strategy. In R.K. Belew and L.B. Booker, editors, *International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann, 1991.

D. Stinson. *An Introduction to the Design and Analysis of Algorithms.* Manitoba : The Charles Babbage Research Centre, 1985.

D. Sudholt. Theory of swarm intelligence. In *GECCO 2017 - Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 902–921, 2017.

M. Tomassini. *Spatially structured evolutionary algorithms: artificial evolution in space and time.* Springer Science & Business Media, 2006.

L.D. Whitley. Cellular genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 658–, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1-55860-299-2. URL http://dl.acm.org/citation.cfm?id=645513.657598.

Table 1. Benchmark of combinatorial optimization problems

Problem	Fitness function	n	Optimum
OneMax	$f_{OneMax}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i$	500	500
MAXCUT	$f_{MAXCUT}(\boldsymbol{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij}[x_i(1-x_j) + x_j(1-x_i)]$	20	56.740064
P-PEAKS	$f_{P-PEAKS}(\boldsymbol{x}) = \frac{1}{n} \max_{1<i<p}(N - HammingD(\boldsymbol{x}, Peak_i))$	300	1.0
MTTP	$f_{MTTP}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i \cdot w_i$	100	0.005

Table 2. Experimental parameters of all GAs

	cGAk	genGAk
Population Size	400 individuals	
Parent Selection	current individual + Binary tournament	
Recombination	Two-point, $pc = 1.0$	
Bit mutation	Bit-flip, $pm = 1/n$	
Stop condition	Find the optimum or reach 10^6 evaluations	
Replacement	Rep_if_not_Worse	–
Neighborhood	C9	–
Lattice	20×20	–
Replacement	–	$(\mu + \lambda)$-prop. selection

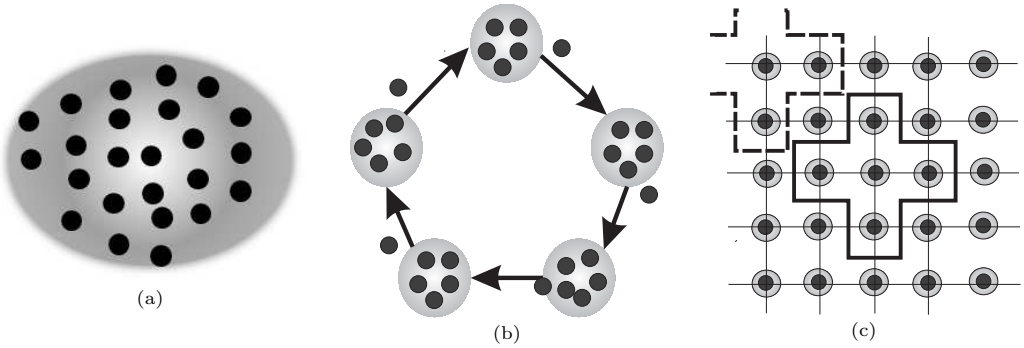
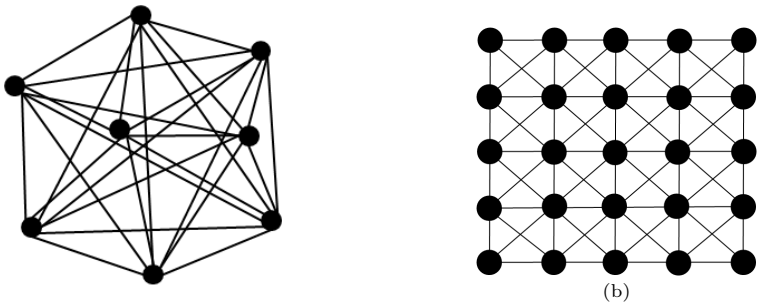


Fig. 1. Panmictic (a), distributed (b), and cellular (c) EAs.



(a) (b)

Fig. 2. Connectivity graph among individuals for panmictic (a), and cellular EAs (b).

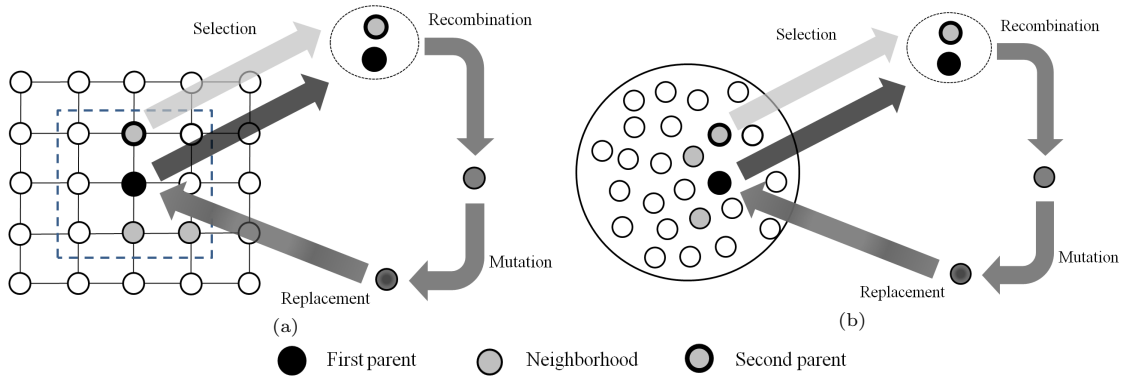
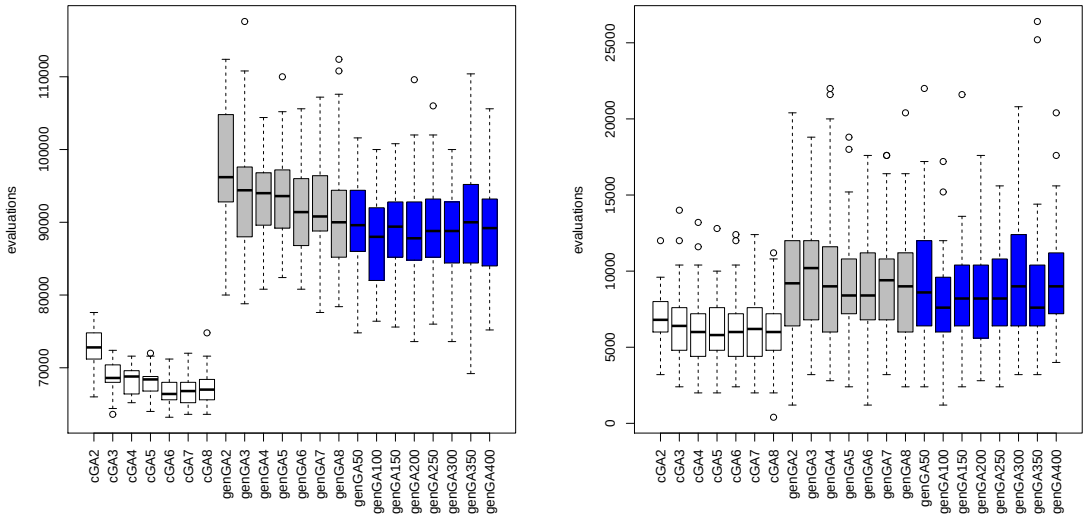
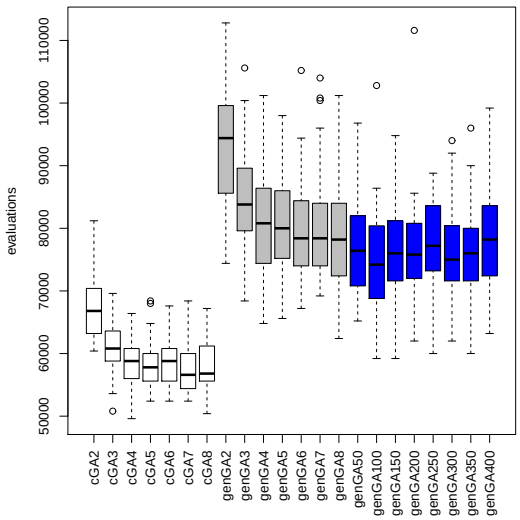


Fig. 3. Reproductive cycle of each individual in a cellular GA (a) and a panmictic GA (b) with $k = 3$.

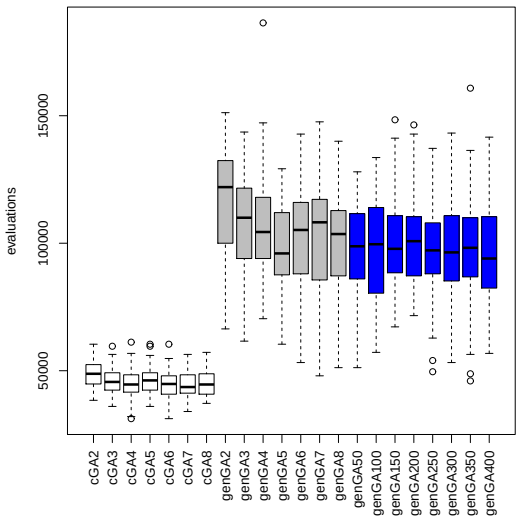


(a) OneMax

(b) MAXCUT

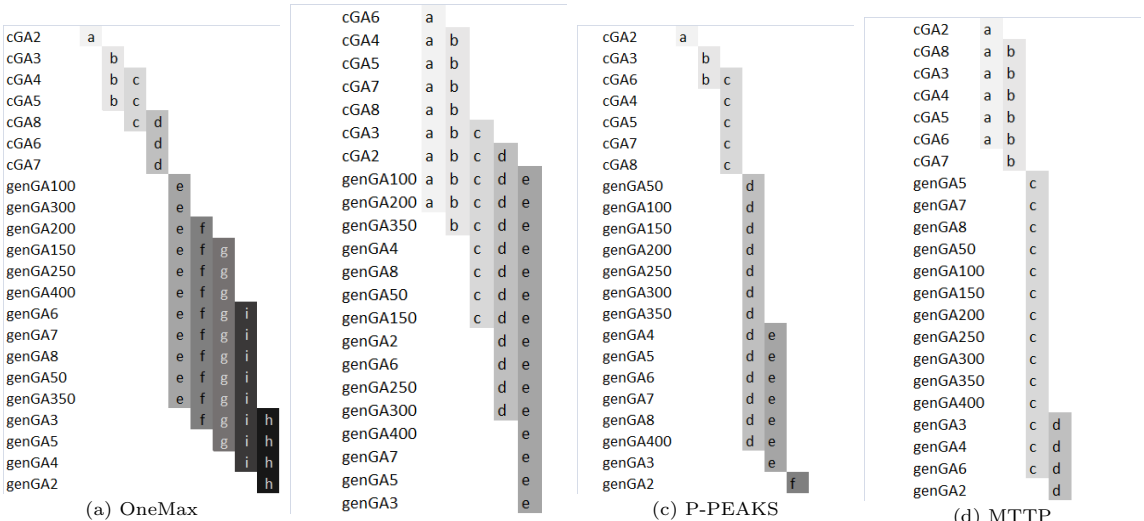


(c) P-PEAKS



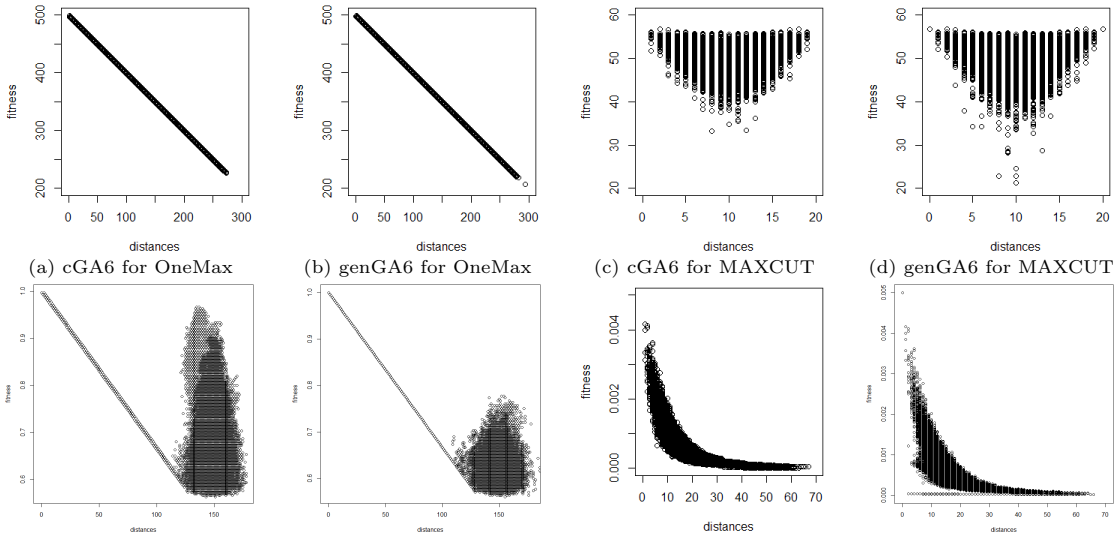
(d) MTTP

Fig. 4. Box-plots of the number of evaluations required for the algorithms under different k sizes to solve the problems.



(a) OneMax (b) MAX_CUT20_09 (c) P-PEAKS (d) MTTP

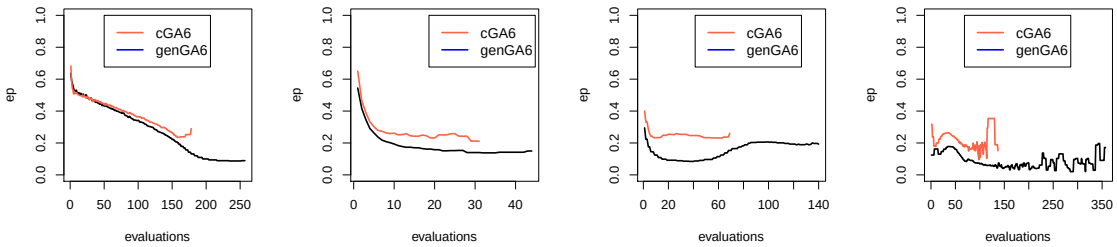
Fig. 5. Wilcoxon's multiple range output.



(a) cGA6 for OneMax (b) genGA6 for OneMax (c) cGA6 for MAXCUT (d) genGA6 for MAXCUT

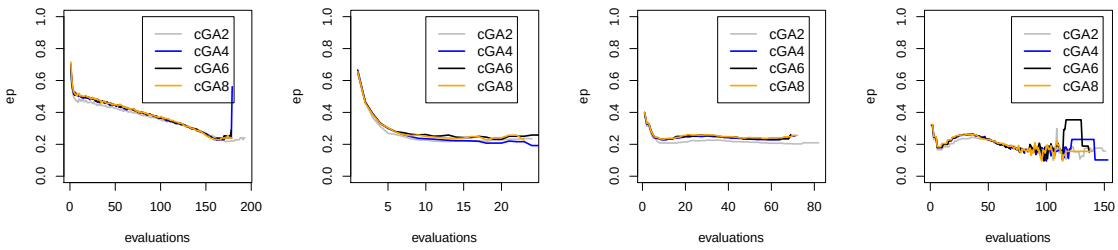
(e) cGA6 for P-PEAKS (f) genGA6 for P-PEAK (g) cGA6 for MTTP (h) genGA6 for MTTP

Fig. 6. Fitness distance plots of cGAk and genGAk ($k = 6$) for all the problems.



(a) OneMax (b) MAXCUT (c) P-PEAKS (d) MTTP

Fig. 7. Escape probability versus iterations of cGAk and genGAk ($k = 6$) for all the problems.



(a) OneMax (b) MAXCUT (c) P-PEAKS (d) MTTP

Fig. 8. Escape probability versus iterations of different cGAk's to solve the problems.