Learning and Nonlinear Models - Journal of the Brazilian Society on Computational Intelligence (SBIC), Vol. 17, Iss. 2, pp. 15–26, 2019

BAYESIAN PARTITION CROSSOVER FOR PSEUDO-BOOLEAN OPTIMIZATION

Diogenes Laertius and Renato Tinós
Department of Computing and Mathematics, FFCLRP, University of São Paulo
{dlaertius}@usp.br, {rtinos}@ffclrp.usp.br

Abstract – The recombination of solutions is important for population metaheuristics and other optimization algorithms. Recently, an efficient recombination operator that preserve the interaction between the decision variables was proposed for pseudo-Boolean optimization. Partition Crossover (PX) groups decision variables in order to allow the decomposition of the evaluation function. PX allows to find, with computational cost proportional to the cost of evaluating one solution of the problem, the best solution among a number of offspring solutions that grows exponentially with the number of recombining components found by the operator. PX has been so far used only in problems where the information about the linkage between the decision variables is known *a priori*. This information is stored in a graph, know as variable interaction graph. We propose a new PX for pseudo-Boolean optimization problems that can be used when the variable interaction graph is not known *a priori*. For this purpose, it is necessary to estimate the linkage between the decision variables by using procedures generally employed in estimation of distribution algorithms. The experimental results show that the new recombination operator generally improves the number of offspring that are better than their parents when compared to traditional recombination operators. However, generating better offspring does not necessarily imply in better performance for the evolutionary algorithm.

Keywords – Genetic Algorithms, combinatorial optimization, recombination.

1. Introduction

Genetic algorithms (GAs) are evolutionary algorithms initially proposed by John Holland [1], which combine the use of a population of solutions, heredity of characteristics, and selection and transformation operators inspired by biological mechanisms. In evolutionary algorithms, individuals are interpreted as candidate solutions for a problem and the fittest individual represents the best solution found. GA, in a simple way, involves three types of operators: selection, recombination and mutation. The selection operators choose the individuals in a population for transformation according to their fitness. The recombination operator, also called crossover operator, recombine decision variables from the parents to generate new individuals (offspring). The mutation operator randomly transforms decision variables of a solution.

The recombination assumes an important role in population metaheuristics, particularly in GAs [2]. However, recombination is not used only in population metaheuristics, but in other optimization paradigms too. For example, multi-trial LKH (Lin-Kernigan-Helsgaun), that is one of the most successful heuristics for the Traveling Salesman Problem (TSP), recombines solutions generated in different trials of local search [3]. Multi-trial LKH obtained success in finding the global optimal in all instances researched by K. Helsgaun and holds records for several instances where optimal solutions are not known [4].

The process of recombining locally optimal solutions found in different runs of an algorithm, or by different algorithms, makes the research for efficient recombination operators of great importance for the area of optimization in general. In evolutionary computation, recombination operators are generally random and do not preserve the relationship between decision variables. In GAs, the term *genetic linkage* is used to represent the interaction between decision variables.

Partition Crossover (PX) is a deterministic crossover that explores the decomposition of the evaluation function and the linkage between decision variables. PX was first proposed by Whitley et al. [5, 6] for the Travelling Salesman Problem. In [7], PX was extended for pseudo-Boolean problems and in [8] it was used in a clustering problem where the decision variables are integers. Basically, partition crossover groups the decision variables into p subsets. The subsets are defined through the identification of connected components in a recombination graph. The recombination graph is obtained by removing, from the variable interaction graph, common features found in two parents .

The variable interaction graph stores the information of the linkage between decision variables in a given instance of an optimization problem. The decomposition of the interaction graph allows to find the best solution among 2^p possible offspring solutions in computational cost similar to evaluating two solutions[1], where p is the number of components found in the recombination graph. This is possible because PX explores the linear decomposition of the evaluation function (or *fitness function*).

All partition crossover presented in the literature require that the variable interaction graph be known *a priori*, i.e., before starting the optimization procedure. In all PX proposed in the literature by now, the variable interaction graph is obtained by analyzing the evaluation function of the instance of the problem. However, there are many problems where it is not possible to build the variable interaction graph by analyzing the evaluation function. Problems where information of the fitness function is

[1]For example, in TSP, the cost of evaluating a solution is O(N), where N is the number of the graph vertices.

not explicitly explored by the algorithm are know as black-box problems. On the other hand, problems where this information is explicitly explored are known as gray-box problems [9]. All PX proposed in the literature by now can be applied only to gray-box problems.

In this work, we propose a PX that can be used in black-box pseudo-Boolean optimization problems, i.e., it is not necessary to know *a priori* the variable interaction graph. In the proposed PX, the variable interaction graph is estimated during the optimization procedure. In order to estimate the interaction graph, procedures employed in estimation of distribution algorithms (EDAs) are used. We use procedures of a popular EDA: the Bayesian optimization algorithm (BOA). BOA was proposed by Pelikan et al. [10] and makes use of Bayesian Networks (BNs) to encode and represent the probabilistic model that is used here as the variable interaction graph [11, 12]. The PX proposed here is called Bayesian Partition Crossover (BPX). BPX uses the same recombination procedures used by the PX for pseudo-Boolean problems proposed in [7]. However, it uses BNs to model the variable interaction graph.

2. Proposed Method

An important concept related to the theory of GAs is the concept of hyperplanes (or schemata). Hyperplanes are subsets of decision variables that can be recombined and transformed in order to improve the fitness of the solutions [13]. The GAs implicitly manipulate large amounts of hyperplanes in order to improve solutions.

The first step of a GA is the initialization of candidate solutions; in general, individual of the initial population are randomly generated. Then, the population is evaluated using the fitness function. The fitness of the individuals is used to select parents that will be reproduced. The offspring are then added to the current population. This process is repeated until a stop criterion is met.

There are different genetic operators employed in GAs. Regarding crossover, the traditional operators in literature, e.g. 2-point crossover and uniform crossover, are random and "blind", which means that they cut and recombine subsets of decision variables without identifying possible good hyperplanes. In other words, the genetic linkage, represented by hyperplanes, are not necessarily preserved. PX for pseudo-Boolean problems, on other hand, preserve good hyperplanes by decomposing the interaction graph according to the information present in the parents.

2.1. PX for Pseudo-Boolean Problems

A pseudo-Boolean function has the form $f : \mathbb{B}^n \to \mathbb{R}$. If the evaluation function can be written as a sum of subfunctions that depends on the maximum *k* variables, then we say the pseudo-Boolean function is also k-restrict. An example of k-restrict pseudo-Boolean function is the NK landscape model. The fitness function for the NK landscape model is given by:

$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}, \mathbf{m}_i) \tag{1}$$

where $\mathbf{x} \in \mathbb{B}^N$ is the candidate solution and the mask $\mathbf{m}_i \in \mathbb{B}^N$ indicates the elements of $\mathbf{x}$ that influence subfunction f_i. Each mask $\mathbf{m}_i$ has $k = K + 1$ ones. The element of $\mathbf{x}$ at position i and K other elements are equal to one, i.e., they influence subfunction f_i.

The NK model is considered hard for genetic algorithms with traditional recombination operators [14]. Traditional recombination operators do not explore the interaction between decision variables to define which elements should be inherited from one or another parent. In this way, the offspring generated by traditional recombination are usually worse than the parents because good hyperplanes are broken. As a consequence, good combinations for the subsets of decision variables that influence a subfunction of the fitness function are broken.

The epistasis between decision variables in a pseudo-Boolean problem can be represented by a directed graph known as variable interaction graph (VIG)[2]. The vertices of the VIG are the decision variables and the edges represent the interaction between variables. When available, the information between the genetic linkage present in the fitness function can be used to build the VIG. In the NK landscape model for example, if a decision variable i influences subfunction j in the fitness function (Equation 1), then an edge is created from vertex i to vertex j.

The main idea of PX is the use of the information about common variables, found in the parents, to decompose the VIG and, as a consequence, the fitness function. If the bits for the parents in a given vertex are equal (i.e., the vertex is common), then the impact of inheriting the bit from one or another parent is the same. Thus, this vertex can be removed from the VIG. The resulting graph is known as recombination graph. The connected components in the recombination graph indicate the bits that should be inherited together from the parents. The connected components, known as recombining components, can be individually evaluated.

The offspring's fitness is computed as a sum of partial evaluations for the connected components inherited from the parents. Figure 1 shows an example of recombination by PX. The VIG (Figure 1.a) is modified by removing edges from common vertices (an alternative is to remove the common vertices). By applying a procedure (e.g., breadth-first search) to find the connected components in the resulting graph (recombination graph), recombining components are found. Each component is defined by elements of the solution influenced only by other elements within the component or by elements outside the component with similar bis for both parents (Figure 1.b). Thus, the partial evaluation for each component is independent from the choice of one

[2]A undirected graph can also be used, depending on the implementation of PX.

or another parent for the elements outside the component. The bits of the offspring are chosen according to the best evaluation, from the bits of one or another parent, inside each component (Figure 1.c).

As a condition for PX to work, it is necessary that the VIG of the problem be known. In the NK landscape model, the VIG is built by analyzing the fitness function; in particular, by analyzing the masks $\mathbf{m}_i$. However, this is not possible for all pseudo-Boolean functions.

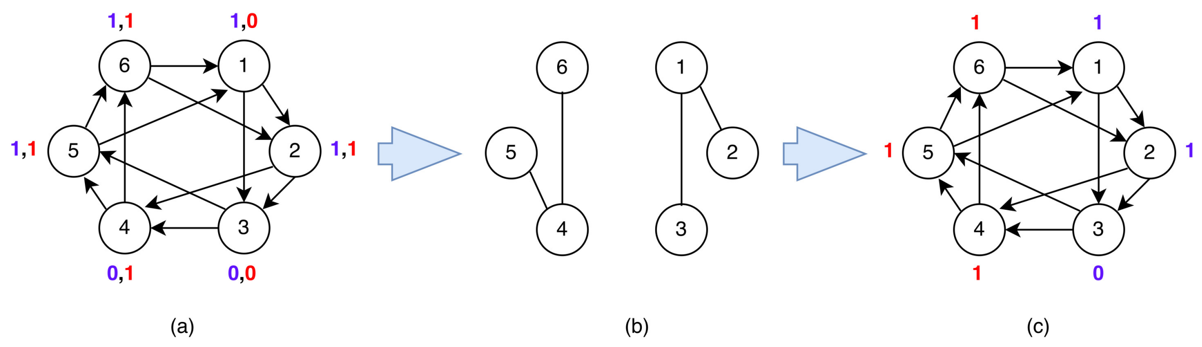

Figure 1: Example of PX for the NK landscape model with $N = 6$ and $K = 2$. a) VIG. In this example, the parents are **x** $=$ $[1, 1, 0, 0, 1, 1]^T$ (in blue) and **y**$=$ $[0, 1, 0, 1, 1, 1]^T$ (in red). In order to identify the recombining components, an undirected graph (recombination graph) is created from the VIG by removing all the edges from vertices i in which $x_i = y_i$. The recombining components are the connected components of the resulting graph. Here two recombining components are identified: the first contains vertices 4, 5 and 6; the second contains vertices 1, 2 and 3. c) The best offspring is generated by the recombination of the best partial solutions (inside the recombining components). In this example, the offspring inherits the variables of the first component from **y** and the variables of the second component from **x**.

2.2. Learning the Problem Structure

EDAs generate new solutions without making use of traditional operators employed in evolutionary algorithms [15]. Offspring are generated by finding partial solutions common to individuals with good fitness. The generation of new individuals occurs through sampling using a probabilistic model based on the knowledge of the problem structure represented by BNs. BOA uses BNs to represent the dependence between variables. Each variable corresponds to a vertex of the graph and the edges indicate the dependence between variables.

Here, we propose to use the BNs created by BOA to represent the VIG in problems where the VIG cannot be built from the analysis of the fitness function. The VIG is used to create the recombination graph in PX. Thus, only the procedure to build the BN from BOA is employed in the proposed recombination operator.

The way BOA estimates the probabilistic model combines the knowledge *a priori* of the problem and the sampling of promising solutions selected from a initial population [10]. When the knowledge is not available, BOA estimates the BN only from the sampled solutions [16, 17].

2.3. Bayesian Partition Crossover (BPX)

The BPX, proposed in this paper, is based on the PX operator and uses procedures of BOA to estimate the VIG of an instance of a pseudo-Boolean problem. When the graph is estimated, PX can be applied. The main advantage is that, unlike traditional PX, this operator can be applied to any pseudo-Boolean problem, since it is not necessary to know a *priori* the VIG. The procedure used in BOA to build the BN is used here to build the VIG, but, unlike in BOA, the model is not built in every generation.

Here, a new VIG is built every n_g generations. The parameter n_g is defined in the beginning of the run. As in BOA, it is expected that the network model generated in BPX will also be improved along the generations. The BN is created using Algorithm K2 [18]. Starting with a BN without edges and N vertices, representing the problem variables, K2 uses a score metric to evaluate how good a BN represents the dataset. A greedy search method is then used to maximize the score metric. In each step of the greedy search, news edges are added to the BN. The gain of each insertion is calculated and used to accept or reject the changes. Here, the BN is used to build the VIG employed in PX. After the VIG is created, the PX is used as described in Algorithm 1.

Algorithm 1 presents PX for pseudo-Boolean problems. In Algorithm 1, we assume that the VIG is available. The first step (line 6) is transferring all bits that are common to both parents to the offspring. Then, in line 7, the recombination graph is built and the subset of connected components π is found using Breadth First Search[3] In lines 8 to 16, the partial evaluation, g^i, for the partial solution inside the connected component, defined by π, for each parent is employed to select the corresponding bits from one or another parent. Inside each recombining (connected) component, the offspring inherits the bits of only one parent. PX is

[3]Other algorithm to find connected components in a graph can be used in this step.

greedy: for each component, PX selects the partial solution that results in the best partial evaluation g^i. Thus, if p recombining components are found, PX returns the best of 2^p reachable offspring. For example, if $p = 10$, the best of $2^{10} = 1024$ individuals is found. The time complexity for one application of PX is O(N) when the maximum degree of the VIG is limited by a constant K.

Here, the maximum degree of vertices of the BN (and also of the VIG) is given by $K = 5$. Thus, each variable is linked to a maximum of 5 other variables. A high degree implies a higher computational cost for Algorithm K2 [19]. High vertex degrees also generally implies in finding less connected components for PX. This is a limitation of PX: when the VIG is highly connected, few recombining components are found, implying in a small number of recombination opportunities. By limiting K here, this problems is minimized. The parameter K used in this work was defined through initial experiments.

The new BPX differs from the traditional PX [7] in two points: i) it uses an estimated VIG, as described before; ii) the partial evaluations of the recombining components are estimated (Line 8 in Algorithm 1). In the traditional PX, the partial evaluations of the recombining components are computed according to the knowledge of the subfunctions in the fitness function of the problem. Here, we consider that these subfunctions are not available and should be estimated. In the method proposed here, the subfunctions estimation is dependent on the problem. We will present the method for estimating the partial evaluations for the recombining components for specific problems in Section 3.

Algorithm 1 Partition Crossover (PX) - Assuming Maximization

```
1:  function PX($f, G, \mathbf{x}, \mathbf{y}$)
2:  Input: $f$                                        ▷ pseudo-Boolean function
3:  Input: $G$                                        ▷ VIG
4:  Input: $\mathbf{x}$ and $\mathbf{y}$              ▷ parents
5:  Output: $\mathbf{z}$                              ▷ offspring
6:  $\mathbf{z} \leftarrow \mathbf{x} \wedge \mathbf{y}$   ▷ transferring common bits
7:  $\pi \leftarrow findConnectedComponents(G[\mathbf{x} \oplus \mathbf{y}])$
8:  Let $f(\mathbf{x}) = \sum_{i=1}^{|\pi|} g^i(\mathbf{x})$ and $f(\mathbf{y}) = \sum_{i=1}^{|\pi|} g^i(\mathbf{y})$
9:      for i = 1 until $|\pi|$ do
10:         if $g^i(\mathbf{x}) > g^i(\mathbf{y})$ then
11:             $\mathbf{z} \leftarrow \mathbf{z} \vee (\mathbf{x} \wedge \pi_i)$
12:         else
13:             $\mathbf{z} \leftarrow \mathbf{z} \vee (\mathbf{y} \wedge \pi_i)$
14:         end if
15:     end for
16: end function
```

Algorithm 2 shows the hybrid GA with BPX that is used in the experiments. The first step of algorithm is to randomly generating the individuals in the initial population (line 3). The individuals in the initial population are improved by first-improvement local search, LSFI (step 4). LSFI transforms random bits and checks if there is an improvement. This process is repeated until all bits are tested or an improvement is detected. The BN is estimated by Algorithm K2 (line 11), using the individuals of the current population selected according to their fitness (line 10). The BN is used here as the VIG G. PX uses G to recombine two parents, $\mathbf{p_1}$ and $\mathbf{p_2}$, in order to create an offspring $Q(i)$ (line 19).

In the hybrid algorithm, the offspring is generated by recombination (line 19) or flip mutation (line 21). In order to preserve the diversity of the population, part of the population is replaced by random individuals every 15 generations (lines 28 -30). Here, 10% of individuals are replaced by random individuals. All solutions are then improved by local search (line 31). The hybrid algorithm was used on previous works with the PX [7].

3. Problems

Experimental results for three pseudo-Boolean problems are presented in Section 4: 0-1 knapsack problem, deceptive function based on trap functions, and NK landscape model. The NK landscape model was used because the VIG is available by analyzing the fitness function; as a consequence, BPX can be compared to PX in this problem. In the other two problems, PX cannot be used; thus, BPX is compared only to traditional crossover. The next subsection present the three problems and how the partial evaluation of the recombining components (Line 8 in Algorithm 1) are computed for each problem.

3.1. 0-1 Knapsack Problem

The 0-1 knapsack problem is a binary variation of a traditional combinatorial optimization problem. The problem is NP-hard for the general case. Hybrid algorithms, composed of genetic algorithms and local search, have presented interesting results for this problem [20]. Variation of this problem can be found in different real-world applications, e.g., in project selection and economic planning [21, 22]. The objective of the 0-1 knapsack problem is to maximize the sum of the profits of items in the knapsack (V) without, however, exceeding the weight capacity (C) of it. The sum of profits items in the knapsack is given by:

$$V(\mathbf{x}) = \sum_{i=1}^{N} p_i x_i \tag{2}$$

Algorithm 2 Hybrid GA with BPX

```
1: Input: $f$                                                    ▷ pseudo-Boolean function
2: for $i \leftarrow 1, n_{pop}$ do
3:     $P(i) = randomIndividual()$;
4:     $P(i) = LSFI(P(i))$;                                      ▷ Local Search with First Improvement
5: end for
6: $g \leftarrow 0$
7: $r \leftarrow 0$
8: while terminated condition is not satisfied do
9:     if $g = 0$ or $g = n_g$ then
10:        $S \leftarrow selectPromisingSolutions(P)$;
11:        $G \leftarrow BayesianNetwork(S)$;                    ▷ building the VIG
12:        $g \leftarrow 0$
13:    end if
14:    $g \leftarrow g + 1$
15:    $Q(1) = bestSolution(P)$;
16:    for $i \leftarrow 2, n_{pop}$ do
17:        $(\mathbf{p_1}, \mathbf{p_2})$ = selection (P);
18:        if $random() < p_{cross}$ then
19:            $Q(i) \leftarrow px(f, G, \mathbf{p_1}, \mathbf{p_2})$;   ▷ execute PX with the graph generated by BN
20:        else
21:            $Q(i) = mutation(\mathbf{p_1})$;
22:        end if
23:    end for
24:    if r=15 then
25:        $Q(1) = bestSolution(Q)$;
26:        $Q(1) = LSFI(Q(1))$;
27:        for $i \leftarrow 2, n_{pop}$ do
28:            if $i < 0.1 * n_{pop}$ then
29:                $Q(i) = randomIndividual()$;
30:            end if
31:            $Q(i) = LSFI(Q(i))$;
32:        end for
33:        $r \leftarrow 0$
34:    end if
35:    $r \leftarrow r + 1$
36:    $P \leftarrow Q$;
37: end while
```

where p_i is the profit for the i-th item and $\mathbf{x} \in \mathbb{B}^N$ is the candidate solution where:

$$x_i = \begin{cases} 1 & \text{, if the } i\text{-th object is selected} \\ 0 & \text{, otherwise} \end{cases} \tag{3}$$

The sum of weights should be constrained by:

$$W(\mathbf{x}) = \sum_{i=1}^{N} w_i x_i \leq V \tag{4}$$

where w_i is the weight of the i-th item. In our experiments, we use the following penalty function [23, 24]:

$$R(x) = \begin{cases} 0, & \text{if } W(\mathbf{x}) \leq V \\ \alpha\left(W(\mathbf{x}) - C\right), & \text{otherwise} \end{cases} \tag{5}$$

where $\alpha = max_{i=1,\dots,N}(p_i w_i)$. Thus, the 0-1 knapsack fitness function is given by:

$$f(\mathbf{x}) = V(x) - R(x) \tag{6}$$

In BPX, the partial evaluation (see Algorithm 1) for each recombining component π_i is computed by summing the profits for the decision variables inside the recombining component:

$$g^i(\mathbf{x}) = \sum_{j \in \pi_i} p_j x_j \tag{7}$$

3.2. Deceptive Functions

Unatation functions depend only on the number of 1s in a string $\mathbf{x} \in \mathbb{B}^N$, regardless of the position in the same [25]. The study of this type of function is interesting since its understanding and manipulation are easy. There are only $N+1$ different function values for a search space with size 2^N. Unatation functions can be used to create trap functions and deceptive functions for GAs. In a trap function, a candidate solution with all alleles equal to 1 is the global optimum and another candidate solution with all alleles equal to 0 is a local optimum with a large basin of attraction [26]. A trap function for a candidate solution $\mathbf{x} \in \mathbb{B}^N$ is created when:

$$f(\mathbf{x}) = \begin{cases} \frac{a}{b}(z - u(\mathbf{x}), & \text{if} \quad u(\mathbf{x}) \leq z \\ \frac{b}{N-z}(u(\mathbf{x}) - z), & \text{otherwise} \end{cases} \tag{8}$$

where $u(\mathbf{x}) = \sum_{i=1}^{N} x_i$ is the unatation function, b and a are respectively the fitness of the global and local optima, and z controls the size of the basin of attraction for the local optimum.

In BPX, the partial evaluation (see Algorithm 1) for each recombining component π_i is computed by summing the contributions in Equation 8 for the decision variables inside the recombining component:

$$g^i(\mathbf{x}) = \begin{cases} \frac{a}{b}(z - u^i(\mathbf{x}), & \text{if} \quad u^i(\mathbf{x}) \leq z \\ \frac{b}{N-z}(u^i(\mathbf{x}) - z), & \text{otherwise} \end{cases} \tag{9}$$

where:

$$u^i(\mathbf{x}) = \sum_{j \in \pi_i} x_j \tag{10}$$

is the unitation function for the decision variables inside recombining component π_i.

3.3. NK Model

The NK Model, or NK Landscape, is a problem introduced by Stuart Kauffaman [27] and is one of the most popular classes of problems for testing optimization algorithms on random instances [28]. In this model, N represents the number of genes in the chromosome and each gene interacts with K other genes. The genes interactions is given by two types of neighborhood in the model: adjacent and random. In adjacent neighborhood, each genes interact with K other adjacent genes adjacent. In random neighborhood, each gene interacts with K other random genes.

The fitness function is computed by using Equation 1. By changing K, the difficult of the problem changes because the number of local optima in the fitness landscape changes.

In BPX, the partial evaluation (see Algorithm 1) for each recombining component π_i is computed by summing the contributions in Equation 1 for the decision variables inside the recombining component:

$$g^i(\mathbf{x}) = \frac{1}{N} \sum_{j \in \pi_i} f_j(\mathbf{x}, \mathbf{m}_j) \tag{11}$$

4. Experimental Results

Results of experiments with the three problems described in Section 3 are presented here. In all problems, BPX was compared to uniform (UN) and 2-point (2PT) crossover. The same hybrid GAs are used; the only difference among the algorithms is the recombination operator. As commented before, BPX was compared to PX only when it was possible to compute the VIG *a priori*, i.e., for the NK Landscapes. For the two other problems, we also show results where BPX is combined with UN and 2PT. The combined strategies are respectively called BPX+UN and BPX+2PT. These combinations have been tested in order to investigate the potential of combining exploratory characteristics of the traditional operators with the exploitative characteristics of BPX. We still investigated the impact of using LSFI or not in these two problems.

In the experiments, a new BN is generated every $n_g = 300$ generations (Algorithm 2). Elitism and tournament selection are used; the size of the tournament pool is 3. The crossover rate is $p_{cross} = 0.6$ and the mutation rate is $p_{mut} = \frac{1}{N}$. The size of the population is $n_{pop} = 50$ for the experiments with the NK model and $n_{pop} = 200$ for the experiments with other problems. The number of runs is $n_{runs} = 50$ and the stop criteria is the number of generations when equal to or greater then twenty thousand generations.

The algorithms were evaluated according to the following criteria:

1. **Mean fitness**: for each run, the best fitness is stored. The mean fitness value is the mean of the best fitness over all runs.

2. **Percentage of individuals better than parents**: for each run, the total number of offspring generated by crossover that are better than their respective parents is stored. This number value stored is divided by the total number of recombination operations performed during the run. This is done only for the 1000 first generations because improvements are rare in late generations. At the end, a percentage of this operations is presented.

3. **Percentage of individuals better than best**: for each run, the total number of offspring generated by crossover that are better than the current best individual (found by the GA in the current run). At the end, the total number is divided by the total number of recombination operations performed during the run. Again, this is done only for the 1000 first generations and a percentage of this operations is presented.

The Wilcoxon Signed-Rank Test with significance level equal to 95% was used to statistically compare the results of the GA with the different recombination operators.

4.1. NK Model

The hybrid GA with different crossover operators was applied to the NK Model with $N = \{100, 300, 500\}$ and with $K = \{1, 2, 3, 5, 10\}$ for the adjacent and random neighborhood models. The results for the mean fitness are presented in tables 1 and 2. The results for the percentage of individuals better than parents are presented in tables 3 and 4, while the percentage of individuals better than best are presented in tables 5 and 6.

Table 1: Mean fitness (and standard deviation) for the hybrid GA with different recombination operators for the experiments with the NK Landscapes with adjacent neighborhood. A symbol $+$ or $-$ for Alg. A indicates that the result for BPX is respectively better or worse than the result for Alg. A. The symbol s indicates that the result is statistically significant. The best results are in bold.

N	K	UX	2PT	PX	BPX
100	1	0.7169 ± 0.0172 (-)	0.7184 ± 0.0173 (-)	**0.7187** ± **0.0172** (-)	0.7162 ± 0.0173
100	2	0.7385 ± 0.0136 (+)	0.7459 ± 0.0140 (s-)	**0.7479** ± **0.0138** (s-)	0.7394 ± 0.0160
100	3	0.7469 ± 0.0135 (-)	0.7577 ± 0.0113 (s-)	**0.7630** ± **0.0112** (s-)	0.7442 ± 0.0141
100	5	0.7444 ± 0.0139 (-)	0.7620 ± 0.0111 (s-)	**0.7670** ± **0.0113** (s-)	0.7417 ± 0.0120
100	10	0.7296 ± 0.0118 (-)	**0.7407** ± **0.0100**(s-)	0.7245 ± 0.0145 (+)	0.7288 ± 0.0122
300	1	0.7158 ± 0.0100 (-)	0.7172 ± 0.0099 (-)	**0.7201** ± **0.0098** (s-)	0.7155 ± 0.0102
300	2	0.7323 ± 0.0109 (+)	0.7374 ± 0.0103 (s-)	**0.7473** ± **0.0095** (s-)	0.7336 ± 0.0106
300	3	0.7379 ± 0.0092 (-)	0.7438 ± 0.0080 (s-)	**0.7632** ± **0.0067** (s-)	0.7379 ± 0.0082
300	5	0.7345 ± 0.0085 (+)	0.7415 ± 0.0071 (s-)	**0.7596** ± **0.0059** (s-)	0.7351 ± 0.0076
300	10	0.7144 ± 0.0080 (s+)	**0.7220** ± **0.0077** (s-)	0.7138 ± 0.0080 (s+)	0.7178 ± 0.0077
500	1	0.7114 ± 0.0082 (-)	0.7132 ± 0.0090 (-)	**0.7199** ± **0.0084** (s-)	0.7112 ± 0.0085
500	2	0.7250 ± 0.0079 (-)	0.7291 ± 0.0086 (s-)	**0.7473** ± **0.0074** (s-)	0.7247 ± 0.0091
500	3	0.7280 ± 0.0063 (+)	0.7341 ± 0.0069 (s-)	**0.7641** ± **0.0047** (s-)	0.7286 ± 0.005
500	5	0.7259 ± 0.0070 (-)	0.7289 ± 0.0058 (s-)	**0.7575** ± **0.0055** (s-)	0.7241 ± 0.0063
500	10	0.7046 ± 0.0058 (+)	**0.7090** ± **0.0066** (s-)	0.7038 ± 0.0063 (+)	0.7057 ± 0.0065

Table 2: Mean fitness of the GA for the NK Landscapes with random neighborhood.

N	K	UX	2PT	PX	BPX
100	1	0.7119 ± 0.0184 (+)	0.7121 ± 0.0186 (+)	**0.7146** ± **0.0179** (-)	0.7121 ± 0.0187
100	2	0.7391 ± 0.0120 (-)	0.7375 ± 0.0136 (-)	**0.7420** ± **0.0132** (-)	0.7366 ± 0.0144
100	3	**0.7498** ± **0.0164** (-)	0.7496 ± 0.0168 (-)	0.7456 ± 0.0150 (-)	0.7450 ± 0.0162
100	5	0.7478 ± 0.0184 (+)	**0.7525** ± **0.0143** (-)	0.7470 ± 0.0140 (+)	0.7498 ± 0.0146
100	10	0.7272 ± 0.0137 (-)	**0.7295** ± **0.0128**(-)	0.7268 ± 0.0146 (+)	0.7268 ± 0.0153
300	1	0.7120 ± 0.0112 (-)	0.7115 ± 0.0112 (-)	**0.7157** ± **0.0109** (s-)	0.7111 ± 0.0111
300	2	0.7308 ± 0.0105 (-)	0.7297 ± 0.0103 (+)	**0.7352** ± **0.0102** (s-)	0.7305 ± 0.0098
300	3	**0.7395** ± **0.0087** (-)	0.7390 ± 0.0111 (-)	0.7366 ± 0.0091 (+)	0.7372 ± 0.0086
300	5	0.7407 ± 0.0079 (-)	**0.7411** ± **0.0090** (-)	0.7399 ± 0.0082 (-)	0.7391 ± 0.0083
300	10	**0.7252** ± **0.0112** (-)	0.7245 ± 0.0084 (-)	0.7239 ± 0.0088 (-)	0.7230 ± 0.0100
500	1	0.7059 ± 0.0081 (+)	0.7056 ± 0.0082 (+)	**0.7142** ± **0.0076** (s-)	0.7060 ± 0.0078
500	2	0.7252 ± 0.0071 (-)	0.7252 ± 0.0080 (-)	**0.7342** ± **0.0071** (s-)	0.7240 ± 0.0080
500	3	0.7316 ± 0.0068 (+)	0.7320 ± 0.0074 (+)	**0.7329** ± **0.0056** (-)	0.7323 ± 0.0065
500	5	0.7352 ± 0.0094 (+)	**0.7366** ± **0.0061** (-)	0.7362 ± 0.0073 (+)	0.7358 ± 0.0075
500	10	0.7211 ± 0.0073 (-)	**0.7230** ± **0.0070** (-)	0.7200 ± 0.0079 (+)	0.7204 ± 0.0077

One can observe that PX presents the the best results for the percentage of individuals better than parents and for the percentage of individuals better than best. This is as expected result, because PX employs the exact VIG for the different instances of the

Table 3: Percentage of individuals better than parents for the NK Model with adjacent neighborhood.

N	K	UX	2PT	PX	BPX
100	1	0.0855 ± 0.0068 (-)	0.0854 ± 0.0072 (-)	**0.3842 ± 0.0163** (s-)	0.0821 ± 0.0209
100	2	0.0739 ± 0.0069 (s-)	0.0802 ± 0.0067 (s-)	**0.3454 ± 0.0180** (s-)	0.0585 ± 0.0154
100	3	0.0616 ± 0.0058 (s-)	0.0754 ± 0.0066 (s-)	**0.3016 ± 0.0165** (s-)	0.0486 ± 0.0130
100	5	0.0491 ± 0.0069 (s-)	0.0708 ± 0.0071 (s-)	**0.1409 ± 0.0179** (s-)	0.0355 ± 0.0083
100	10	0.0346 ± 0.0043 (s-)	**0.0607 ± 0.0051** (s-)	0.0423 ± 0.0115 (s-)	0.0244 ± 0.0061
300	1	0.1175 ± 0.0085 (s+)	0.0981 ± 0.0067 (s+)	**0.3979 ± 0.0144** (s-)	0.1400 ± 0.0175
300	2	0.1045 ± 0.0078 (s+)	0.0917 ± 0.0074 (s+)	**0.3559 ± 0.0182** (s-)	0.1124 ± 0.0200
300	3	0.0926 ± 0.0072 (s+)	0.0887 ± 0.0064 (s+)	**0.3217 ± 0.0167** (s-)	0.0998 ± 0.0116
300	5	0.0802 ± 0.0072 (s+)	0.0808 ± 0.0063 (s+)	**0.2881 ± 0.0179** (s-)	0.0874 ± 0.0133
300	10	0.0574 ± 0.0062 (s+)	0.0710 ± 0.0059 (s-)	**0.1260 ± 0.0191** (s-)	0.0624 ± 0.0131
500	1	0.1320 ± 0.0074 (s+)	0.1030 ± 0.0056 (s+)	**0.4048 ± 0.0154** (s-)	0.1596 ± 0.0169
500	2	0.1178 ± 0.0065 (s+)	0.0963 ± 0.0054 (s+)	**0.3779 ± 0.0183** (s-)	0.1288 ± 0.0170
500	3	0.1108 ± 0.0066 (s+)	0.0924 ± 0.0045 (s+)	**0.3359 ± 0.0173** (s-)	0.1220 ± 0.0144
500	5	0.0946 ± 0.0071 (s+)	0.0860 ± 0.0065 (s+)	**0.3112 ± 0.0195** (s-)	0.1049 ± 0.0111
500	10	0.0738 ± 0.0075 (s+)	0.0768 ± 0.0057 (s+)	**0.1722 ± 0.0180** (s-)	0.0806 ± 0.0108

Table 4: Percentage of individuals better than parents for the NK Model with random neighborhood.

N	K	Uniform	2-Points	PX	BPX
100	1	0.0894 ± 0.0082 (s+)	0.0767 ± 0.0070 (s+)	**0.3815 ± 0.0154** (s-)	0.0974 ± 0.0201
100	2	0.0758 ± 0.0072 (s+)	0.0661 ± 0.0064 (s+)	**0.2901 ± 0.0191** (s-)	0.0667 ± 0.0155
100	3	0.0662 ± 0.0068 (s+)	0.0561 ± 0.0051 (s+)	**0.1507 ± 0.0209** (s-)	0.0572 ± 0.0116
100	5	0.0515 ± 0.0062 (s+)	0.0471 ± 0.0052 (s+)	**0.0713 ± 0.0155** (s-)	0.0394 ± 0.0119
100	10	0.0346 ± 0.0047 (s-)	**0.0347 ± 0.0048** (s-)	0.0168 ± 0.0082 (s+)	0.0264 ± 0.0066
300	1	0.1186 ± 0.0080 (s+)	0.0938 ± 0.0056 (s+)	**0.3963 ± 0.0142** (s-)	0.1827 ± 0.0239
300	2	0.1060 ± 0.0065 (s+)	0.0850 ± 0.0049 (s+)	**0.3671 ± 0.0200** (s-)	0.1339 ± 0.0163
300	3	0.0941 ± 0.0074 (s+)	0.0761 ± 0.0063 (s+)	**0.2841 ± 0.0199** (s-)	0.1076 ± 0.0166
300	5	0.0807 ± 0.0074 (+)	0.0631 ± 0.0055 (s+)	**0.1776 ± 0.0171** (s-)	0.0827 ± 0.0106
300	10	0.0558 ± 0.0065 (s+)	0.0476 ± 0.0052 (s+)	**0.0989 ± 0.0162** (s-)	0.0639 ± 0.0136
500	1	0.1366 ± 0.0067 (s+)	0.1017 ± 0.0049 (s+)	**0.3956 ± 0.0181** (s-)	0.2114 ± 0.0202
500	2	0.1218 ± 0.0072 (s+)	0.0909 ± 0.0050 (s+)	**0.3859 ± 0.0161** (s-)	0.1567 ± 0.0160
500	3	0.1082 ± 0.0071 (s+)	0.0821 ± 0.0054 (s+)	**0.3427 ± 0.0175** (s-)	0.1252 ± 0.0187
500	5	0.0935 ± 0.0072 (s+)	0.0709 ± 0.0045 (s+)	**0.2257 ± 0.0165** (s-)	0.1031 ± 0.0119
500	10	0.0714 ± 0.0057 (s+)	0.0552 ± 0.0049 (s+)	**0.1464 ± 0.0158** (s-)	0.0751 ± 0.0095

Table 5: Percentage of individuals better than best for the NK Model with adjacent neighborhood.

N	K	Uniform	2-Points	PX	BPX
100	1	0.0186 ± 0.0038 (s+)	0.0181 ± 0.0040 (s+)	**0.0707 ± 0.0114** (s-)	0.0318 ± 0.0109
100	2	0.0159 ± 0.0035 (s+)	0.0159 ± 0.0031 (s+)	**0.0779 ± 0.0108** (s-)	0.0221 ± 0.0076
100	3	0.0127 ± 0.0029 (s+)	0.0135 ± 0.0027 (s+)	**0.0630 ± 0.0130** (s-)	0.0171 ± 0.0057
100	5	0.0090 ± 0.0029 (s+)	0.0109 ± 0.0027 (s+)	**0.0371 ± 0.0124** (s-)	0.0112 ± 0.0044
100	10	0.0050 ± 0.0019 (+)	0.0067 ± 0.0021 (s-)	**0.0202 ± 0.0078** (s-)	0.0051 ± 0.0029
300	1	0.0408 ± 0.0053 (s+)	0.0337 ± 0.0037 (s+)	**0.1022 ± 0.0117** (s-)	0.0675 ± 0.0103
300	2	0.0363 ± 0.0041 (s+)	0.0303 ± 0.0035 (s+)	**0.1257 ± 0.0127** (s-)	0.0525 ± 0.0121
300	3	0.0314 ± 0.0043 (s+)	0.0288 ± 0.0037 (s+)	**0.1219 ± 0.0118** (s-)	0.0444 ± 0.0080
300	5	0.0268 ± 0.0043 (s+)	0.0227 ± 0.0031 (s+)	**0.1002 ± 0.0136** (s-)	0.0363 ± 0.0089
300	10	0.0176 ± 0.0035 (s+)	0.0157 ± 0.0035 (s+)	**0.0705 ± 0.0150** (s-)	0.0206 ± 0.0069
500	1	0.0514 ± 0.0043 (s+)	0.0406 ± 0.0029 (s+)	**0.1228 ± 0.0140** (s-)	0.0819 ± 0.0114
500	2	0.0449 ± 0.0033 (s+)	0.0365 ± 0.0028 (s+)	**0.1613 ± 0.0124** (s-)	0.0640 ± 0.0101
500	3	0.0424 ± 0.0041 (s+)	0.0337 ± 0.0029 (s+)	**0.1483 ± 0.0121** (s-)	0.0588 ± 0.0090
500	5	0.0358 ± 0.0037 (s+)	0.0293 ± 0.0037 (s+)	**0.1274 ± 0.0146** (s-)	0.0460 ± 0.0065
500	10	0.0260 ± 0.0039 (s+)	0.0223 ± 0.0033 (s+)	**0.1002 ± 0.0129** (s-)	0.0304 ± 0.0058

NK Landscapes. PX preserves the linkage between the decision variables. BPX estimates the VIG, as well the contribution of the subfunctions. Thus, it presents worse results than PX. However, the results are better than those for the traditional operators for both neighborhood models.

It is also important to observe that the impact of increasing K is bigger for PX than for BPX. As discussed before, BPX uses a constant value for vertices degree of the BN. Higher mean vertices degree implies in less connected components in the

Table 6: Percentage of individuals better than best for the NK Model with random neighborhood.

N	K	Uniform	2-Points	PX	BPX
100	1	0.0201 ± 0.0033 (s+)	0.0169 ± 0.0042 (s+)	**0.0696 ± 0.0090** (s-)	0.0370 ± 0.0112
100	2	0.0164 ± 0.0038 (s+)	0.0135 ± 0.0034 (s+)	**0.0704 ± 0.0127** (s-)	0.0254 ± 0.0087
100	3	0.0154 ± 0.0038 (s+)	0.0105 ± 0.0029 (s+)	**0.0584 ± 0.0133** (s-)	0.0226 ± 0.0076
100	5	0.0104 ± 0.0034 (s+)	0.0084 ± 0.0022 (s+)	**0.0363 ± 0.0114** (s-)	0.0131 ± 0.0071
100	10	0.0051 ± 0.0016 (+)	0.0048 ± 0.0018 (+)	**0.0078 ± 0.0054** (-)	0.0058 ± 0.0031
300	1	0.0415 ± 0.0046 (s+)	0.0335 ± 0.0033 (s+)	**0.1032 ± 0.0123** (s-)	0.0895 ± 0.0147
300	2	0.0371 ± 0.0037 (s+)	0.0302 ± 0.0030 (s+)	**0.1489 ± 0.0181** (s-)	0.0626 ± 0.0109
300	3	0.0322 ± 0.0042 (s+)	0.0264 ± 0.0030 (s+)	**0.1311 ± 0.0121** (s-)	0.0491 ± 0.0096
300	5	0.0276 ± 0.0034 (s+)	0.0210 ± 0.0031 (s+)	**0.1004 ± 0.0130** (s-)	0.0341 ± 0.0076
300	10	0.0183 ± 0.0033 (s+)	0.0141 ± 0.0027 (s+)	**0.0531 ± 0.0107** (s-)	0.0238 ± 0.0077
500	1	0.0531 ± 0.0038 (s+)	0.0407 ± 0.0030 (s+)	**0.1178 ± 0.0134** (s-)	0.1089 ± 0.0120
500	2	0.0469 ± 0.0043 (s+)	0.0362 ± 0.0027 (s+)	**0.1790 ± 0.0145** (s-)	0.0779 ± 0.0115
500	3	0.0421 ± 0.0036 (s+)	0.0324 ± 0.0031 (s+)	**0.1685 ± 0.0135** (s-)	0.0609 ± 0.0120
500	5	0.0359 ± 0.0037 (s+)	0.0270 ± 0.0024 (s-)	**0.1340 ± 0.0130** (s-)	0.0452 ± 0.0076
500	10	0.0267 ± 0.0034 (s+)	0.0195 ± 0.0025 (s+)	**0.0827 ± 0.0111** (s-)	0.0289 ± 0.0058

recombination graph. As a consequence, less recombining components are found.

It is clear from the analysis of Tables 3 to 6, that BPX generates better offspring than UX and 2PT for the NK Landscapes. However, better offspring does not necessarily implies in better performance for the algorithm. This can be observed in tables 1 and 2. This occurs because BPX is too greedy for this problem. Thus, the diversity of the population is decreased. Diversity is important in order that individuals be improved by mutation and local search. BPX is strongly exploitative, while UX and 2PT are strongly explorative. A combination of such characteristics would be desirable.

4.2. 0-1 Knapsack Problem

In the experiments with the 0-1 Knapsack Problem, the weights and profits are positive real numbers randomly chosen from a defined range of 5 to 20 for weights and from 40 to 100 for profit. The knapsack capacity is equal to $C = 0.5S_w$, where S_w is the sum of the weights of all items. The chromosome size is $N = 100$.

The results for the experiments with the 0-1 Knapsack Problem are presented in tables 7, 8, and 9. Experiments with the hybrid GA with and without local search are presented.

In the experiments with the 0-1 Knapsack Problem, UX obtained the better performance, followed by 2PT and BPX. The results are explained because this problem has constraints. BPX estimates the contributions of the recombining components because they cannot be directly computed by analyzing the fitness function. However, solutions are penalized in the fitness function (Equation 6) and the penalties are not computed when analyzing the recombining components in BPX. This is an disadvantage of BPX, i.e., offspring are evaluated without taking in consideration the penalties. A possible improvement for BPX in this case is to estimate the subfunctions taking in consideration the penalties. However, this would imply in using a different method for estimation, that can be explored in future works.

Table 7: Mean fitness for the 0-1 Knapsack Problem.

	UX	2PT	BPX	BPX+UN	BPX+2PT
With LSFI	**4557.15 ± 106.11** (-)	4556.85 ± 105.03 (-)	4555.05 ± 103.50	4556.20 ± 106.70 (-)	4555.25 ± 104.75 (-)
Without LSFI	4553.26 ± 106.74 (+)	4553.64 ± 105.17 (+)	**4553.87 ± 103.51**	4553.05 ± 105.23 (+)	4551.69 ± 102.45 (+)

Table 8: Percentage of individuals better than parents for the 0-1 Knapsack Problem.

	UX	2PT	BPX	BPX+UN	BPX+2PT
With LSFI	**0.0795 ± 0.007** (s-)	0.0554 ± 0.004 (s-)	0.0109 ± 0.002	0.0386 ± 0.004 (s-)	0.0324 ± 0.0037 (s-)
Without LSFI	**0.0353 ± 0.007** (s-)	0.0220 ± 0.004 (s-)	0.0100 ± 0.002	0.0257 ± 0.004 (s-)	0.0202 ± 0.004 (s-)

Table 9: Percentage of individuals better than best for the 0-1 Knapsack Problem.

	UX	2PT	BPX	BPX+UN	BPX+2PT
With LSFI	0.0031 ± 0.000 (s-)	0.0027 ± 0.000 (s-)	0.0016 ± 0.000	**0.0038 ± 0.001** (s-)	0.0029 ± 0.001 (s-)
Without LSFI	**0.0084 ± 0.002** (s-)	0.0062 ± 0.001 (s-)	0.0022 ± 0.001	0.0067 ± 0.001 (s-)	0.0059 ± 0.002 (s-)

4.3. Deceptive Functions

In the experiments with deceptive functions, $N = 50$, $a = 0.80$, $b = 1$, and $z = 48$ [26]. The results are presented in tables 10-12.

All algorithms presented the same mean fitness (Table 10). This is a hard problem for the GAs and the local optimum was reached in all runs. However, tables 11 and 12 indicate that BPX resulted in a better performance for the percentage of individuals better than parents and individuals better than best. This result indicates that BPX was able to find good hyperplanes that improved the population.

Table 10: Mean fitness for the deceptive functions.

	UX	**2PT**	**BPX**	**BPX+UN**	**BPX+2PT**
Without LsFi	0.8000 ± 0.00	0.8000 ± 0.00	0.8000 ± 0.00	0.8000 ± 0.00	0.8000 ± 0.00

Table 11: Percentage of individuals better than parents for the deceptive functions.

	UX	**2PT**	**BPX**	**BPX+UN**	**BPX+2PT**
Without LsFi	0.0511 ± 0.0068 (s+)	0.0389 ± 0.007 (s+)	$\mathbf{0.0643 \pm 0.012}$	0.0618 ± 0.011 (s+)	0.0579 ± 0.009 (s+)

Table 12: Percentage of individuals better than best for the deceptive functions.

	UX	**2PT**	**BPX**	**BPX+UN**	**BPX+2PT**
Without LsFi	0.0133 ± 0.003 (s+)	0.0107 ± 0.003 (s+)	$\mathbf{0.0206 \pm 0.007}$	0.0179 ± 0.006 (s+)	0.0186 ± 0.006 (+)

5. Conclusion and Future Work

The investigation of recombination operators that preserve good hyperplanes is recent in evolutionary computing. In this work, we presented a new recombination operator (BPX). BPX is a partition crossover, but unlike other operators of this type, it can be used in problems where the variable interaction graph is not known *a priori*. BPX estimates the variable interaction graph by using methods employed in BOA to estimate Bayesian Networks that model the interaction between decision variables. BPX uses the common information contained in parents to partition the variable interaction graph. The connected components of the resulting graph are recombining components, i.e., they indicate the recombination mask.

The experimental results indicated that BPX generated better offspring, when compared to traditional crossover operators, in two of the three problems investigated in this paper. BPX increased the percentage of offspring better than parents and offspring better than the best current solution in the population. The only problem where BPX did not result in better offspring was the 0-1 knapsack problem. This is a problem with penalties and the way the recombining components are evaluated for this problem proved to be misleading.

The main limitation of the proposed algorithm is that BNs need to be constructed and this consumes computational time. This affects the scalability of the algorithm when the method is compared to simple GAs. When the termination criterion is based on a fixed time, BPX presented inferior performance compared to other crossover operators. This behaviour is mainly due to the creation of a centralized storage model of genetic linkage information (VIG) during the execution of the algorithm. The investigation of new ways of building the VIG is a possible future work.

In this way, a future work is to investigate better ways to evaluate recombining components, especially for optimization problems with constraints. Now, the recombining components are evaluates using the knowledge provided by the fitness function. However, this can be avoided by using, instead, the individuals of the population for sampling the partial evaluations of the recombining components.

In the experiments, generating better offspring did not necessarily imply in better performance for the evolutionary algorithm. In fact, BPX resulted in worse performance for the NK Landscapes. BPX is too greedy for this problem, what resulted in low diversity populations. BPX is strongly exploitative, while traditional recombination operators are strongly explorative. Another relevant future work is to explore the combination of such characteristics in a new recombination operator.

6. Acknowledgements

This work was supported by The São Paulo Research Foundation (FAPESP) under grant 2016/16769-0.

References

[1] J. H. Holland. "Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence". *Ann Arbor, MI: University of Michigan Press*, 1975.

[2] A. E. Eiben, J. E. Smith *et al.*. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[3] W. Cook. *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press, 2012.

[4] K. Helsgaun. "General k-opt submoves for the Lin–Kernighan TSP heuristic". *Mathematical Programming Computation*, vol. 1, no. 2-3, pp. 119–163, 2009.

[5] D. Whitley, D. Hains and A. Howe. "Tunneling between optima: partition crossover for the traveling salesman problem". In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pp. 915–922. ACM, 2009.

[6] R. Tinós, D. Whitley and G. Ochoa. "A new generalized partition crossover for the traveling salesman problem: tunneling between local optima". *Evolutionary Computation*, pp. 1–31, 2019.

[7] R. Tinós, D. Whitley and F. Chicano. "Partition crossover for pseudo-boolean optimization". In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pp. 137–149. ACM, 2015.

[8] R. Tinós, L. Zhao, F. Chicano and D. Whitley. "NK hybrid genetic algorithm for clustering". *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 748–761, 2018.

[9] D. Whitley. "Next Generation Genetic Algorithms: A User's Guide and Tutorial". In *Handbook of Metaheuristics*, pp. 245–274. Springer, 2019.

[10] M. Pelikan, D. E. Goldberg and E. Cantú-Paz. "BOA: The Bayesian optimization algorithm". In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pp. 525–532. Morgan Kaufmann Publishers Inc., 1999.

[11] F. M. Nyikosa, M. A. Osborne and S. J. Roberts. "Bayesian Optimization for Dynamic Problems". *arXiv preprint arXiv:1803.03432*, 2018.

[12] M. Nishio, M. Nishizawa, O. Sugiyama, R. Kojima, M. Yakami, T. Kuroda and K. Togashi. "Computer-aided diagnosis of lung nodule using gradient tree boosting and Bayesian optimization". *PloS one*, vol. 13, no. 4, pp. e0195875, 2018.

[13] M. Melanie. *An introduction to genetic algorithms*, volume 3. 1999.

[14] R. B. Heckendorn, S. Rana and D. Whitley. "Test function generators as embedded landscapes". *Foundations of Genetic Algorithms*, vol. 5, pp. 183–198, 1999.

[15] M. K. Crocomo. "Bayesian optimization algorithm with community detection". Ph.D. thesis, Universidade de São Paulo, 2012.

[16] A. Eiben and J. Smith. *Introduction to evolutionary computing (natural computing series)*. Springer, 2008.

[17] M. Pelikan and D. E. Goldberg. "Bayesian optimization algorithm: From single level to hierarchy". Ph.D. thesis, PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2002. Also IlliGAL Report, 2002.

[18] G. F. Cooper and E. Herskovits. "A Bayesian method for the induction of probabilistic networks from data". *Machine learning*, vol. 9, no. 4, pp. 309–347, 1992.

[19] M. Scanagatta, C. P. de Campos, G. Corani and M. Zaffalon. "Learning Bayesian networks with thousands of variables". In *Advances in neural information processing systems*, pp. 1864–1872, 2015.

[20] J. Zhao, T. Huang, F. Pang and Y. Liu. "Genetic algorithm based on greedy strategy in the 0-1 knapsack problem". In *Genetic and Evolutionary Computing, 2009. WGEC'09. 3rd International Conference on*, pp. 105–107. IEEE, 2009.

[21] H. M. Salkin and C. A. De Kluyver. "The knapsack problem: a survey". *Naval Research Logistics Quarterly*, vol. 22, no. 1, pp. 127–144, 1975.

[22] D. Pisinger. "The quadratic knapsack problem—a survey". *Discrete applied mathematics*, vol. 155, no. 5, pp. 623–648, 2007.

[23] A. L. Olsen. "Penalty functions and the knapsack problem". In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pp. 554–558. IEEE, 1994.

[24] R. Tinós and S. Yang. "A framework for inducing artificial changes in optimization problems". *Information Sciences*, vol. 485, pp. 486–504, 2019.

[25] K. Deb and D. E. Goldberg. "Sufficient conditions for deceptive and easy binary functions". *Annals of mathematics and Artificial Intelligence*, vol. 10, no. 4, pp. 385–408, 1994.

[26] R. Tinós and S. Yang. "A self-organizing random immigrants genetic algorithm for dynamic optimization problems". *Genetic Programming and Evolvable Machines*, vol. 8, no. 3, pp. 255–286, 2007.

[27] S. A. Kauffman. *The origins of order: Self-organization and selection in evolution.* Oxford University Press, USA, 1993.

[28] M. Pelikan, K. Sastry, D. E. Goldberg, M. V. Butz and M. Hauschild. "Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap". In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pp. 851–858. ACM, 2009.