Tip the Balance: Improving Exploration of Balanced Crossover Operators by Adaptive Bias

Luca Manzoni[1], Luca Mariot[2], and Eva Tuba[3]

[1] Dipartimento di Matematica e Geoscienze, Università degli Studi di Trieste, Via Valerio 12/1, Trieste, 34127, Italy
lmanzoni@units.it

[2] Cyber Security Research Group, Delft University of Technology, Mekelweg 2, Delft, The Netherlands
l.mariot@tudelft.nl

[3] Faculty of Informatics and Computing, Singidunum University, Danijelova 32, 11000 Belgrade, Serbia
etuba@ieee.org

Abstract. The use of balanced crossover operators in Genetic Algorithms (GA) ensures that the binary strings generated as offsprings have the same Hamming weight of the parents, a constraint which is sought in certain discrete optimization problems. Although this method reduces the size of the search space, the resulting fitness landscape often becomes more difficult for the GA to explore and to discover optimal solutions. This issue has been studied in this paper by applying an adaptive bias strategy to a counter-based crossover operator that introduces unbalancedness in the offspring with a certain probability, which is decreased throughout the evolutionary process. Experiments show that improving the exploration of the search space with this adaptive bias strategy is beneficial for the GA performances in terms of the number of optimal solutions found, even if these benefits are not reflected in the resulting fitness distributions.

Keywords: Genetic algorithms · crossover operators · boolean functions · balancedness · nonlinearity

1 Introduction

When dealing with certain optimization problems coming from the area of combinatorics, cryptography, and coding theory, it is often required that the candidate solutions satisfy a balancedness constraint, or more precisely that the bitstrings representing them have a specified *Hamming weight* (i.e., a fixed number of ones). In the context of *Genetic Algorithms* (GA), however, traditional variation operators such as one-point crossover and flip mutation which are used to explore the search space cannot guarantee that the generated solutions preserve the required Hamming weight [13].

In general, one can envision two main approaches to cope with this issue. The first one is to consider the balancedness constraint as a further property to be

optimized in addition to those already taken into account. In a single-objective optimization setting, this usually amounts to adding a *penalty factor* in the fitness function which punishes deviation from the desired Hamming weight. This approach has the advantage of yielding a GA with deterministic running time. Given enough fitness evaluations, the population always converges to the desired Hamming weight, since the penalty factor is a rather easy optimization objective to meet. However, in this way, the GA explores a search space that is much larger than the admissible set of balanced bitstrings, and this usually results in sub-optimal solutions concerning the other optimization objectives. The second method consists in designing *ad-hoc variation operators*, which ensures that the new solutions generated throughout the optimization process all have the required number of ones. In this way, the search space is greatly reduced, since the GA is constrained to explore only the space of feasible solutions.

As far as our knowledge goes, the first attempt at designing a balanced crossover operator for GA dates back to the work of Millan et al. [13]. There, the authors considered the problem of evolving highly nonlinear balanced Boolean functions for cryptographic applications, and they employed a counter-based crossover operator to ensure that the truth table of the offspring function is composed of an equal number of zeros and ones. Later, Chen et al. [2,3] proposed a combination genetic algorithm for evolving investment portfolios, where the underlying crossover operator preserved the Hamming weight of the vector specifying which assets to invest in. Meinl and Berthold [12] investigated a balanced two-point and a uniform crossover operator for the k-subset selection problem, which has relevance for virtual screening of molecules in drug design. More recently, Mariot and Leporati [9] modified the aforementioned counter-based crossover of Millan et al. to cope with three-valued strings, which were evolved as Walsh spectra of plateaued Boolean functions. The same approach has been extended in [10] to evolve balanced quaternary strings representing pairwise-balanced cellular automata rules, which were then used to build orthogonal Latin squares and in [11] to search for binary orthogonal arrays.

The approach of designing ad-hoc variation operators has been recently studied by the authors in [8], where three different *balanced crossover operators* have been investigated in the context of three optimization problems related to cryptography and combinatorial designs. In particular, the authors assessed that the use of balanced crossover operators gives an advantage over one-point crossover for the considered problems. However, the results also showed that over large problem instances these balanced operators usually produce sub-optimal solutions for the optimization properties other than the balancedness constraint. The reason could lie in the fact that the reduced search space induced by the balanced operators has many isolated local optima where the GA gets stuck.

In this paper, we investigate a hybrid approach where we allow a balanced crossover operator to produce *partially unbalanced* candidate solutions with a certain probability. Similarly to the philosophy underlying *simulated annealing*, the rationale is to accept in the early stages of the optimization process slightly worse solutions with respect to the balancedness constraint, to allow the GA to

escape local optima and improve its exploration capabilities, and then to focus only on a region of the admissible solution set by decreasing the probability of introducing unbalancedness in the offspring. In particular, we modify the counter-based crossover operator originally proposed by Millan et al. [13] and later considered in the experimental evaluation of [8] by introducing an *adaptive bias strategy*, where the probability of adding unbalancedness in the offspring is gradually decreased by using a *geometric cooling mechanism* analogous to the one employed in simulated annealing.

We experimentally evaluate this adaptive bias strategy on the optimization problem of *balanced nonlinear Boolean functions*, where the objective is to find a highly nonlinear Boolean function of n variables whose truth table is composed by an equal number of ones and zeros. More precisely, we perform a parameter sweep for the unbalancedness probability and the cooling factor over the problem instance of Boolean functions of $n = 7$ variables, showing that three parameter combinations for the adaptive bias strategy can produce optimal balanced solutions of nonlinearity 56. Although the number of such solutions is still limited concerning the number of experimental runs performed, we remark that nonetheless, our results improve on the original counter-based crossover considered in [8], where the best solutions obtained have nonlinearity 54. Thus, this adaptive bias strategy seems to represent a promising approach to be further investigated in the context of balanced crossover operators.

The rest of this paper is organized as follows. Section 2 recalls the details of the counter-based crossover operator described in [8] and then describes our adaptive bias strategy to modify this crossover. Section 3 presents the experimental evaluation of the adaptive bias strategy. Finally, Section 4 sums up the main contributions of the paper and points out some possible future directions of research on the topic.

2 Partially Unbalanced Crossover

In this section, we first delve into the details of the balanced crossover operators based on counters originally proposed in [13], whose performances with respect to one-point crossover have been later investigated in [8]. We then introduce our adaptive bias strategy by modifying this crossover operator to allow the generation of partially unbalanced offspring with a specified probability.

2.1 Counter-Based Balanced Crossover

In what follows, we consider a bitstring of length $n \in \mathbb{N}$ as a vector of the n-dimensional vector space $\mathbb{F}_2^n$, where $\mathbb{F}_2 = \{0, 1\}$ is the finite field with two elements. Given $x \in \mathbb{F}_2^n$, the *support* of x is the set $supp(x) = \{i : 1 \leq i \leq n,\ x_i \neq 0\}$ which specifies the positions of the elements set to 1 in x. The *Hamming weight* of x is then defined as $w_H(x) = |supp(x)|$, i.e. the number of ones in x. The number of n-bit strings with a specified Hamming weight $1 \leq k \leq n$ is $\binom{n}{k}$, since it is equal to the number of ways in which one can select a subset of k elements from a set

of cardinality n. This one-to-one relationship can be easily seen by identifying a bitstring $x \in \mathbb{F}_2^n$ of Hamming weight k with the characteristic function of the corresponding subset of k elements.

Given two bitstrings $x, y \in \mathbb{F}_2^n$ of Hamming weight k, the aim of a balanced crossover operator is to produce an offspring bitstring $z \in \mathbb{F}_2^n$ having the same weight k as the parents. As mentioned in the previous section this can be accomplished by adopting several encodings for the candidate solutions, three of which have been explored in [8]. Directly working on the bitstring representation is the most straightforward option, and this is the approach adopted in the *counter-based crossover operator* originally proposed by Millan et al. in [13]. The main idea underlying this operator is to build the offspring by copying bit by bit either from the first or the second parent by selecting them at random, and then use two *counters* cnt_0 and cnt_1 to control respectively the multiplicities of 0 and 1. As soon as one of the two counters reaches its prescribed threshold (which is $n - k$ for cnt_0, and k for cnt_1), the remaining positions of the offspring are filled with the complementary value to maintain the balancedness constraint. More precisely, the child chromosome $z \in \mathbb{F}_2^n$ is built from $x, y \in \mathbb{F}_2^n$ of Hamming weight k as follows:

Initialization Set both cnt_0 and cnt_1 to 0

Loop For all positions $i \in \{1, \cdots, n\}$ of the child $z \in \mathbb{F}_2^n$, do one the following:

- If $cnt_0 = n - k$, set z_i to 1
- If $cnt_1 = k$, set z_i to 0
- If both $cnt_0 < n-k$ and $cnt_1 < k$, randomly copy with uniform probability either x_i or y_i in z_i, and update the relevant counter

Return z

From the high-level description above, it is easy to see that the Hamming weight of the produced offspring is always k, since if $n - k$ zeros are reached then z is filled only with 1, while if k ones are copied then z is completed only with zeros. Figure 1 reports an example of this counter-based crossover operator applied to two bitstrings $x, y \in \mathbb{F}_2^8$ of length 8 and Hamming weight $k = 4$. The cells colored in green in the offspring z correspond to the positions copied from the first parent x, while those in green are taken from y. Finally, the rightmost two cells in blue are those that are forced to be set to 0, since the counter cnt_1 reaches the threshold $k = 4$ at position $i = 6$.

Looking at the way this crossover operator works, one might wonder whether it introduces a positional bias in the offspring bitstrings. In fact, by going from left to right the last bits are more likely to be set deterministically, once one of the two counters reaches its threshold. This issue has been investigated in [8], where the authors also considered a *shuffling version* of this crossover operator where the offspring positions are randomly permuted before starting to copy from the parents. Interestingly enough, the results in that work showed not only that the shuffled version of the counter-based crossover operator is not beneficial, but in certain optimization problems (i.e. the search of binary orthogonal arrays addressed in [11]) worsens the GA performances. For this reason, in the rest of

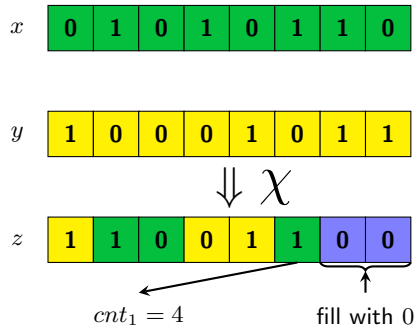


Fig. 1: Example counter-based crossover applied over two 8-bit strings with Hamming weight $w_H = 4$.

this work, we only consider the basic "left-to-right" version described above of the counter-based crossover operator.

2.2 Adaptive Bias Strategy

We now introduce the adaptive bias strategy which we used to modify the counter-based crossover described in the previous section, to allow a certain amount of unbalancedness in the generated offspring. The motivation behind this idea is that by allowing the GA to explore a slightly larger space of candidate solutions it could be easier to escape local optima than just by searching in the set of admissible solutions satisfying the required balancedness constraint.

The adaptive bias strategy comes into play in the counter-based crossover once one of the two counters for the control of the Hamming weight reaches its respective threshold. Suppose that the counter-based crossover has been applied over two bitstrings $x, y \in \mathbb{F}_2^n$ and that the threshold of k has been achieved for the counter of ones cnt_1 at a certain position i. For the $n - i$ remaining positions of the offspring, instead of directly copying 0, the crossover *continues* to copy 1 with a certain *unbalancedness probability* $p \in [0, 1]$. Thus, at each position $i < j \leq n$ a random number $r \in [0, 1]$ is drawn with uniform probability, and if r is less than p then 1 is copied in the j-th position of the offspring, and the process is repeated for the next position. On the other hand, in the case where $r \geq p$ the value 0 is copied in z_j, and *all remaining positions* are set deterministically to 0. In this way, the probability of obtaining an offspring bitstring composed of $k + n - i$ ones (which would result in a high deviation from the desired Hamming weight k) is p^{n-i}. Symmetrically, the same process is applied if the threshold of $n - k$ zeros is first reached by cnt_0 at position i, by continuing to copy with probability p the value 0 in the positions $i < j \leq n$ of the offspring z, and filling the remaining ones with value 1 once the sampled random number r is greater than or equal to p. The probability of generating an offspring with $2n - k - i$ zeros is therefore again p^{n-i}.

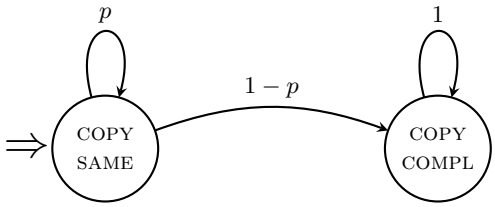


Fig. 2: Finite state machine representation of the adaptive bias strategy.

Figure 2 depicts the finite state diagram representing the technique described above. In particular, the initial state is COPY-SAME, in which the same value associated to the counter that reached its threshold is still copied, thereby introducing unbalancedness in the offspring. The control stays in state COPY-SAME with probability p, while it changes to COPY-COMPL with probability $1 - p$. After reaching COPY-COMPL, the crossover continues to copy the complementary bit value with probability 1.

Algorithm 1 reports the pseudocode of our modified version of the counter-based crossover described in [8], which includes the adaptive bias strategy described above. The input to this algorithm are the two parent bitstrings $x, y \in \mathbb{F}_2^n$, their length n and Hamming weight k, and the unbalancedness probability p. The first 10 lines of Algorithm 1 implements the basic counter-based crossover before one of the counters reaches its threshold. Thus, each position i of z is determined by randomly copying either the value x_i or y_i with uniform probability, which is performed at line 3 by the subroutine RANDOM-SELECT(). Then, the relevant counter is increased depending on the copied value at lines 4–8. The **if-else** block at lines 11–15 determines which is the bit value val whose counter reached the corresponding threshold. Then, in the **while** loop at lines 17–30 the offspring z is completed with the adaptive bias strategy. In particular, the decision whether the same bit value val or its complement $val \oplus 1$ must be copied in position i is controlled by the flag *same*, which is set to false as soon as the random number r sampled at line 19 is greater than or equal to the unbalancedness probability p. Once all positions have been filled, the algorithm finally returns the offspring z.

The use of the modified crossover operator described in Algorithm 1 allows generating bitstrings where the bias of the Hamming weight is related to the unbalancedness probability p. This parameter p must be in turn controlled to drive the evolutionary process towards solutions whose number of ones increasingly approaches the target weight. Intuitively, the goal is to favor *exploration* of the search space of slightly unbalanced bitstrings in the early first stages of the optimization process, and then to focus on the *exploitation* of a subset of the space of bitstrings with the desired Hamming weight. To this end, we adopted a *discount mechanism* of the unbalancedness mechanism inspired by the *geometric cooling schedule* of simulated annealing [4]. In particular, after a certain number m of fitness evaluations, the unbalancedness probability is updated as follows:

$$p \leftarrow \alpha \cdot p \ , \tag{1}$$

Algorithm 1 COUNTER-CROSS-UNBAL(x, y, n, k, p)

```
    cnt_0 := 0; cnt_1 := 0; z := 0^n; i = 0;
    while cnt_0 < n − k AND cnt_1 < k do
      z_i := RANDOM-SELECT(x_i, y_i)
      if (z_i = 1) then
 5:     cnt_1 := cnt_1 + 1
      else
        cnt_0 := cnt_0 + 1
      end if
      i := i + 1
10: end while
    if cnt_0 = n − k then
      val := 0
    else
      val := 1
15: end if
    same := TRUE
    while i < n do
      if same = TRUE then
        r := RANDOM(0, 1)
20:     if r < p then
          z_i := val
        else
          z_i := val ⊕ 1
          same := FALSE
25:     end if
      else
        z_i := val ⊕ 1
      end if
      i := i + 1
30: end while
    return z
```

where $\alpha \in (0, 1)$ is a *cooling factor* analogous to that used in simulated annealing. In this way, the unbalancedness probability decreases exponentially, thus making the generation of unbalanced solutions more and more unlikely as the GA optimization process proceeds. In particular, if $p_0 \in (0, 1)$ is the initial unbalancedness probability set at the beginning of the GA, the resulting probability after $t \in \mathbb{N}$ updates is $p(t) = \alpha^t \cdot p_0$.

Further, to foster the selection of solutions with a Hamming weight close to the required one, we also adopted a *penalty factor* at the fitness function level. Formally, given a bitstring $x \in \mathbb{F}_2^n$, this penalty factor is simply defined as the absolute value of the difference between the Hamming weight of x and the target weight k, i.e. $pen(x) = |w_H(x) - k|$. In case of maximization problems, $pen(x)$ is subtracted from the fitness value of x, while for minimization problems it is added. In our experiments, we investigated two versions of the penalty factor: in the first one, the full value of the penalty factor is taken into account throughout

the whole optimization process, while in the second one it is weighted with the complementary value of the unbalancedness probability p, that is

$$w_{pen}(x) = (1 - p) \cdot pen(x) \ . \tag{2}$$

The motivation underlying the use of this *adaptive penalty factor* $w_{pen}(x)$ as defined in Equation 2 is to discount the Hamming weight of the solutions produced in the early stages of the evolutionary process when computing their fitness values, thereby favoring exploration of the search space. Subsequently, as the unbalancedness probability p decreases, w_{pen} approaches the full value of the penalty factor, thus shifting the focus towards the selection of solutions with the desired Hamming weight.

3 Experiments

In this section, we present the experimental evaluation that we performed on our adaptive bias strategy applied to the counter-based crossover operator. We first briefly introduce the optimization problem considered in the experiments, namely maximizing the nonlinearity of balanced Boolean functions. We then describe the experimental settings and parameters adopted for our tests, and finally, we present the obtained results.

3.1 Balanced Nonlinear Boolean Functions

As a test problem, we considered the optimization of nonlinearity in balanced *Boolean functions*, which comes from the area of cryptography and coding theory, and that has been tackled in several works with evolutionary algorithms (see e.g. [13,9,15]). In particular, this is one of the three problems considered in [8] to investigate the performances of balanced crossover operators in GA, including the counter-based crossover described in Section 2.1: thus, we used this problem to compare the results with those reported in [8].

We now briefly recall the basic notions about Boolean functions necessary to define our optimization problem of interest. For further details on this subject, the reader is referred to [1].

A *Boolean function* of $n \in \mathbb{N}$ variables is a mapping of the form $f : \mathbb{F}_2^n \to \mathbb{F}_2$. The *truth table* Ω_f of f is the 2^n-bit string which represents for each input vector $x \in \mathbb{F}_2^n$ the value of $f(x)$ in lexicographic order. For cryptographic applications, it is desirable that f is *balanced*, i.e. that its truth table is composed by an equal number of zeros and ones. In other words, the Hamming weight of Ω_f must be 2^{n-1}. Another important cryptographic property is the *nonlinearity* of f, which is the distance of f from the set of all affine functions. This can be computed via the *Walsh transform* of f, which is defined for all $a \in \mathbb{F}_2^n$ as:

$$W_f(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus a \cdot x} \ , \tag{3}$$

where $a \cdot x$ denotes the scalar product of a and x, and $\oplus$ is the XOR operation. Then, the nonlinearity of f is:

$$Nl(f) = 2^{n-1} - \frac{1}{2} \cdot \max_{a \in \mathbb{F}_2^n} \{|W_f(a)|\} \ . \tag{4}$$

As a cryptographic criterion, the nonlinearity of f should be as high as possible. Thus, the optimization problem requires finding a Boolean function of n variables having highest possible nonlinearity. Given $f : \mathbb{F}_2^n \to \mathbb{F}_2$, we defined the following two fitness functions to be maximized, depending on whether the penalty factor is weighted or not:

$$fit_1(f) = Nl(f) - pen(\Omega_f) = Nl(f) - |2^{n-1} - w_H(\Omega_f)| \ , \tag{5}$$

$$fit_2(f) = Nl(f) - wpen(\Omega_f) = Nl(f) - (1-p) \cdot |2^{n-1} - w_H(\Omega_f)| \ . \tag{6}$$

In particular, the candidate solutions evolved by GA are encoded by their truth tables, which, as recalled above, are binary strings of length 2^n.

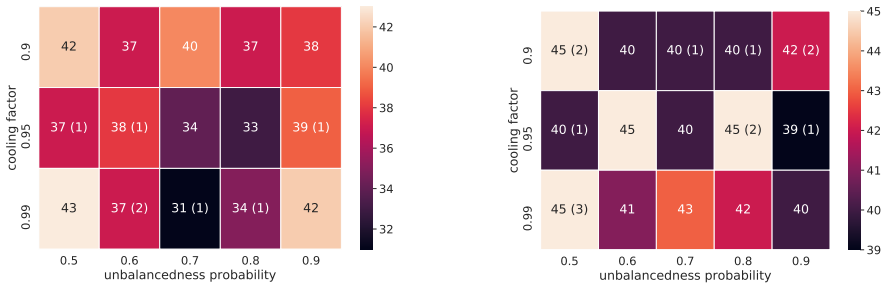
3.2 Experimental Settings

For the sake of comparison, we adopted an experimental setting closely matching the one used in [8] in our experiments. As a test problem instance, we considered the space of Boolean functions of $n = 7$ variables, since as noted in [8] for $n = 6$ variables all balanced crossover operators converge quite easily on optimal balanced solutions of nonlinearity 26. On the other hand, for $n = 7$ variables only the map-of-ones crossover was able to produce one optimal balanced solution of nonlinearity 56 over 50 experimental runs, while all the other ones reached a maximum nonlinearity of 54.

We employed a steady-state GA with a tournament selection operator, where at each iteration $t = 3$ random individuals from the population are sampled. The best two individuals in the tournament are then selected for crossover and mutation, while the third one is replaced by the offspring produced by the two parents. For the crossover, we compared the basic counter-based operator and its modified version with our adaptive bias strategy. Concerning mutation, we used the swap-based mutation operator of [8] with the same mutation probability $p_m = 0.7$. The population is composed of 50 individuals, and we stop the GA after $fit = 1\,000\,000$ fitness evaluations. In particular, for our adaptive bias strategy, we updated the unbalancedness probability every 2 000 fitness evaluations, thus a total of 500 updates took place during an evolutionary run. Finally, each experiment is repeated for $R = 50$ independent runs.

3.3 Results

As far as we know, there are no studies in the literature suggesting what is the best amount of unbalance that can improve the performance of optimization algorithms for the search of highly nonlinear balanced Boolean functions. For this reason, we



(a) Full penalty factor (b) Weighted penalty factor

Fig. 3: Heatmaps for the parameter sweep over $p \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ and $\alpha \in \{0.9, 0.95, 0.99\}$ on the space of Boolean functions of $n = 7$ variables.

performed a parameter sweep over the two main values characterizing our adaptive bias strategy, namely the unbalancedness probability p and the cooling factor α. In particular, for p we considered the values in the set $\{0.5, 0.6, 0.7, 0.8, 0.9\}$, while for α we adopted the values in $\{0.9, 0.95, 0.99\}$. In particular, for the latter parameter, we chose to focus on values close to 1 in order to have a slow decrease in the unbalancedness probability, similarly to the works that used simulated annealing to search for cryptographic Boolean functions [4,9].

Figures 3a and 3b report the heatmaps of our parameter sweep experiments respectively for fitness function fit_1 (where the full penalty factor $pen(\cdot)$ is considered) and fit_2 (where the dynamically weighted penalty $wpen(\cdot)$ is used instead). For each parameter combination (p, α), the numerical entry in the corresponding heatmap reports the number of best balanced solutions found by the GA over the 50 experimental runs which reached a nonlinearity of at least 54. The second number in parentheses, if present, indicates the number of solutions reaching nonlinearity 56 (which is the maximum possible for balanced Boolean functions of 7 variables). In general, it can be observed that using the dynamically weighted penalty factor *wpen* improves the performances of the adaptive bias strategy, since all parameter combinations except one reach at least the 80% of experimental runs where the best solution found has nonlinearity greater than or equal to 54. The same applies also for the number of parameter combinations finding optimal solutions of nonlinearity 56. For this reason, we considered only the results with the dynamically weighted penalty factor for our subsequent comparison.

To compare our results, we selected the best parameter combinations for the adaptive bias strategy with the weighted penalty factor achieving a 90% success rate. Here the success rate is defined as the number of runs where the best solutions have nonlinearity 54, and where *at least* one run produced an optimal solution of nonlinearity 56. This selection resulted in the three parameter combinations $(p = 0.5, \alpha = 0.9)$, $(p = 0.5, \alpha = 0.99)$ and $(p = 0.8, \alpha = 0.95)$.

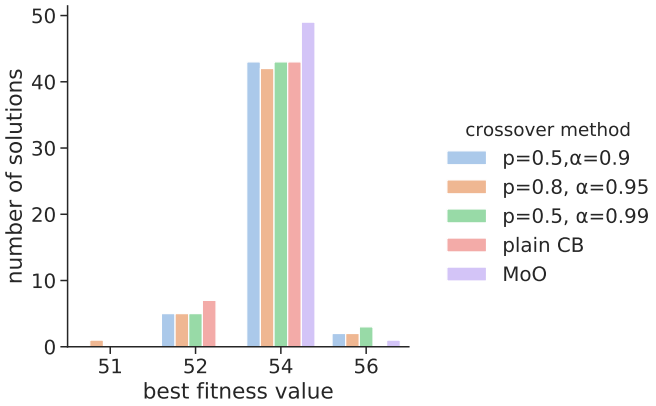


Fig. 4: Distribution of the fitness values for the five compared methods.

We then compared their results with those achieved by the basic counter-based crossover and the map-of-ones crossover analyzed in [8]. In particular, according to the results of [8] the map-of-ones operator scored the best performance over the problem instance of $n = 7$ variables. Notice that we tested these two crossover operators using the same experimental settings showed in Section 3.2, which are the same as those adopted in [8] except for the larger number of fitness evaluations considered (1 000 000 instead of 500 000). Figure 4 depicts the distribution of the best fitness values achieved by the three selected parameter combinations of our adaptive bias strategy and by the basic counter-based and map-of-ones crossover (respectively denoted as plain CB and MoO). It can be remarked from the plot that the three best combinations of our adaptive bias strategy can produce more optimal solutions of nonlinearity 56 than both the basic counter-based crossover (which produced none) and the map-of-ones crossover (which produced only one out of 50 experimental runs). In particular, the combination $(p = 0.5, \alpha = 0.99)$ produced 3 optimal solutions. We performed a statistical analysis of the results with the Mann-Whitney U-test [5], by pairwise comparing the distributions of the best fitness obtained by the five tested crossover operators. The alternative hypothesis is that the results of two crossovers tested are different. The tests show that there is no statistical difference ($\alpha = 0.01$) between any of the proposed methods and either the MoO and the plain CB. However, there is a statistical difference between MoO and plain CB. This shows that, in some sense, the partially unbalanced crossovers are "between" MoO and plain CB with respect to the solutions found.

4 Conclusions

In this paper, we proposed an adaptive bias strategy to perturb the behavior of a counter-based balanced crossover operator, to introduce a certain amount of unbalancedness in the offspring. The motivation behind this strategy is to improve the GA exploration of the search space in the first stages of the optimization process, to avoid early convergence to local optima. This is accomplished in the crossover operator by continuing to copy the binary allele that already reached the prescribed threshold with a specific unbalancedness probability p, while with probability $1 - p$ the chromosome is completed with the complementary value. The unbalancedness probability is then exponentially decreased by multiplying it by a cooling factor of α. We tested this approach on the problem of maximizing the nonlinearity of balanced Boolean functions and compared our results with the balanced crossovers analyzed in [8]. Specifically, we focused on the space of Boolean functions of $n = 7$ variables, and we performed a parameter sweep for tuning both the unbalancedness probability p and the cooling factor α. The results showed that using a dynamically weighted penalty factor in the fitness function where the weight is the complement of the unbalancedness probability allows generating more solutions with nonlinearity greater than or equal to 54. Next, we compared the three best parameter combinations emerging from our parameter sweep with those achieved by the plain counter-based crossover and the map-of-one crossover defined in [8]. The comparison showed that the three parameters combinations are able to produce slightly better results concerning the number of optimal balanced solutions with nonlinearity 56.

Looking at the distribution plot of Figure 4, a natural question is whether our adaptive bias strategy gives a significant advantage over the plain counter-based and the map-of-ones crossover, especially since the statistical tests performed are inconclusive. However, our best parameter combination can produce 3 optimal balanced functions of nonlinearity 56 out of 50 experimental runs, while the map-of-ones generated only one of them. We remark moreover that the search of highly nonlinear balanced Boolean functions is known to be a difficult combinatorial optimization problem for GA [4,9,15,14]. In particular, while it is relatively easy to converge over an optimal balanced solution for $n = 6$ variables, a steep increase in the difficulty of the problem can be observed on the $n = 7$ problem instance [7,8]. Therefore, even a slight improvement in the number of optimal solutions produced over this problem instance is of interest, and our adaptive bias strategy seems to represent an interesting candidate for achieving this goal.

Clearly, further research in this direction is required to boost even more the performance of our adaptive bias strategy for the counter-based crossover operator. In addition to increasing the population size, a first idea would be to perform some additional tuning of the unbalancedness probability and the cooling factor around the three combinations that yielded the best results in our parameter sweep, to assess how the GA performance changes by applying some small perturbations. It would also be interesting to investigate the fitness landscape over the space of Boolean functions of $n = 7$ variables, to understand what amount of unbalancedness would be optimal for our adaptive bias strategy.

One possibility in this respect would be to employ *Local Optima Network* analysis, which the authors of [6] performed for the case of vectorial Boolean functions. Finally, a third direction for future research is to investigate the performance of our adaptive bias strategy over other optimization problems that require balanced solutions, such as the bent functions problem or the orthogonal arrays problem considered in [8] or the evolution of pairwise-balanced cellular automata rules to build orthogonal Latin squares studied in [10].

References

1. Carlet, C., Crama, Y., Hammer, P.L.: Boolean functions for cryptography and error-correcting codes. In: Crama, Y., Hammer, P.L. (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering, pp. 257–397. Cambridge University Press (2010)
2. Chen, J., Hou, J.: A combination genetic algorithm with applications on portfolio optimization. In: Advances in Applied Artificial Intelligence, 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2006, Annecy, France, June 27-30, 2006, Proceedings. pp. 197–206 (2006)
3. Chen, J., Hou, J., Wu, S., Chang-Chien, Y.: Constructing investment strategy portfolios by combination genetic algorithms. Expert Syst. Appl. **36**(2), 3824–3828 (2009)
4. Clark, J.A., Jacob, J.L., Maitra, S., Stanica, P.: Almost boolean functions: The design of boolean functions by spectral inversion. Comput. Intell. **20**(3), 450–462 (2004)
5. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. J. Heuristics **15**(6), 617–644 (2009)
6. Jakobovic, D., Picek, S., Martins, M.S.R., Wagner, M.: A characterisation of s-box fitness landscapes in cryptography. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019. pp. 285–293 (2019)
7. Manzoni, L., Mariot, L., Tuba, E.: Does constraining the search space of GA always help?: the case of balanced crossover operators. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019. pp. 151–152 (2019)
8. Manzoni, L., Mariot, L., Tuba, E.: Balanced crossover operators in genetic algorithms. Swarm and Evolutionary Computation **54**, 100646 (2020)
9. Mariot, L., Leporati, A.: A genetic algorithm for evolving plateaued cryptographic boolean functions. In: Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings. pp. 33–45 (2015)
10. Mariot, L., Picek, S., Jakobovic, D., Leporati, A.: Evolutionary algorithms for the design of orthogonal latin squares based on cellular automata. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017. pp. 306–313 (2017)
11. Mariot, L., Picek, S., Jakobovic, D., Leporati, A.: Evolutionary search of binary orthogonal arrays. In: Parallel Problem Solving from Nature - PPSN XV - 15th

International Conference, Coimbra, Portugal, September 8-12, 2018, Proceedings, Part I. pp. 121–133 (2018)
12. Meinl, T., Berthold, M.R.: Crossover operators for multiobjective k-subset selection. In: Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009. pp. 1809–1810 (2009)
13. Millan, W., Clark, A.J., Dawson, E.: Heuristic design of cryptographically strong balanced boolean functions. In: Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding. pp. 489–499 (1998)
14. Picek, S., Carlet, C., Guilley, S., Miller, J.F., Jakobovic, D.: Evolutionary algorithms for boolean functions in diverse domains of cryptography. Evol. Comput. **24**(4), 667–694 (2016)
15. Picek, S., Jakobovic, D., Miller, J.F., Batina, L., Cupic, M.: Cryptographic boolean functions: One output, many design criteria. Appl. Soft Comput. **40**, 635–653 (2016)