Toward more efficient heuristic construction of Boolean functions

Jakobovic, Domagoj; Picek, Stjepan; Martins, Marcella S.R.; Wagner, Markus

DOI
10.1016/j.asoc.2021.107327

Publication date
2021

Document Version
Accepted author manuscript

Published in
Applied Soft Computing

Toward more efficient heuristic construction of Boolean functions

Domagoj Jakobovic[a], Stjepan Picek[b], Marcella S. R. Martins[c], Markus Wagner[d]

[a] *University of Zagreb - Croatia*
[b] *Delft University of Technology - The Netherlands*
[c] *Federal University of Technology - Parana, Brazil*
[d] *University of Adelaide - Australia*

Abstract

Boolean functions have numerous applications in domains as diverse as coding theory, cryptography, and telecommunications. Heuristics play an important role in the construction of Boolean functions with the desired properties for a specific purpose. However, there are only sparse results trying to understand the problem's difficulty. With this work, we aim to address this issue. We conduct a fitness landscape analysis based on Local Optima Networks (LONs) and investigate the influence of different optimization criteria and variation operators. We observe that the naive fitness formulation results in the largest networks of local optima with disconnected components. Also, the combination of variation operators can both increase or decrease the network size. Most importantly, we observe correlations of local optima's fitness, their degrees of interconnection, and the sizes of the respective basins of attraction. This can be exploited to restart algorithms dynamically and influence the degree of perturbation of the current best solution when restarting.

Keywords:
balancedness, nonlinearity, landscape analysis, local optima networks

Email addresses: `domagoj.jakobovic@fer.hr` (Domagoj Jakobovic), `s.picek@tudelft.nl` (Stjepan Picek), `marcella@utfpr.edu.br` (Marcella S. R. Martins), `markus.wagner@adelaide.edu.au` (Markus Wagner)

1. Introduction

Boolean functions are mathematical objects that can be uniquely represented in truth tables and they have applications in diverse domains. Not only do they form a core concept in combinatorial optimization, such as in the satisfiability problem, but they are used to construct Hadamard matrices [1], strongly regular graphs [2], and decision diagrams [3]. In coding theory, every binary unrestricted code of length 2^n can be interpreted as a set of Boolean functions [4, 5]. In sequences, bent sequences constructed using bent Boolean functions have the lowest value of mutual correlations and autocorrelations, and they are used in communication systems with multiple access [6]. In telecommunications, bent Boolean functions are used in CDMA networks [7]. In cryptography, Boolean functions are used in stream and block ciphers as the source of nonlinearity [8, 9], the design of hash functions [10], or for generating pseudorandom numbers [11]. While various domains have different usages of Boolean functions, some shared characteristics remain. For instance, the ratio between the zeros and ones in the Boolean function's truth table is an important characteristic for many fields. Similarly, the nonlinearity property is not only relevant in cryptography, but also coding theory and sequences. Unfortunately, such widespread use of Boolean functions can also represent a problem since there are numerous scenarios (e.g., considering Boolean function size or relevant properties) for Boolean functions, and it is not always readily available how to construct the required Boolean function.

There are several construction methods to construct Boolean functions: algebraic constructions, random search, heuristics, and combinations of those methods [12]. The advantages of heuristics seem to be (1) the ability to generate many different functions, (2) easy adjustment for different criteria, and (3) very good performance if the size of a Boolean function is not too large. On the other hand, the main drawbacks are (1) no guarantee that optimal solutions will be reached, (2) for every new Boolean function size, new optimization needs to be undertaken, and (3) due to the huge search space size, heuristics are limited in the Boolean function size. In practice, in many domains, the size n of a Boolean function is not very large. For instance, in error-correcting codes, the sizes usually do not surpass 10 since they already give codes of size 2^n (i.e., codes of length 1 024). In cryptography, when used as vectorial Boolean functions, they rarely surpass the size 8, and in the stream ciphers, the size was at most 10 until recent algebraic attacks, and now, the size goes up to 20 inputs. At the same time, already for $n > 5$, the exhaustive search is not possible. Note, that, for a Boolean function with n inputs, there are 2^{2^n} possible Boolean functions.

Heuristics is applied to evolve Boolean functions for cryptography [13] and combinatorial designs [14, 15]. What is more, some of the common properties of Boolean functions commonly evolved with heuristics are relevant is the telecommunications [7] and sequences [6] domains. Thus, while heuristics has an important role in the design of Boolean functions, there are only sparse results trying to understand the problem's difficulty or when it can reach optimal solutions. Fitness landscape analysis (FLA) studies the influence of representations on the design of such heuristics, addressing the relative importance of features in explaining the algorithm performance [16].

This article investigates how a range of different design decisions can affect the search for Boolean functions. In particular, we conduct the first FLA for Boolean functions considering several function sizes most occurring in the literature, Boolean function properties, and variation operators in isolation as well as in combination. As far as we know, this is also the first time that combined neighborhood strategies are applied (in parallel) and considered in an FLA context in general.

2. Boolean Functions and Their Properties

Let n be a positive integer, i.e., $n \in \mathbb{N}^+$. The set of all n-tuples of elements in the field $\mathbb{F}_2$ is denoted as $\mathbb{F}_2^n$ where $\mathbb{F}_2$ is the Galois field with two elements. The inner product of two vectors a and b is denoted by $a \cdot b$ and equals $a \cdot b = \bigoplus_{i=0}^{n-1} a_i b_i$. Here, "$\oplus$" represents addition modulo two (bitwise XOR).

An $(n, 1)$-function is any mapping f from $\mathbb{F}_2^n$ to $\mathbb{F}_2$ and such a function is called the Boolean function. A Boolean function f on $\mathbb{F}_2^n$ can be uniquely represented by a truth table (TT), which is a vector $(f(0), ..., f(1))$ that contains the function values of f, ordered lexicographically, i.e., $a \leq b$.

The Walsh-Hadamard transform W_f is a unique representation of a Boolean function that measures the correlation between $f(x)$ and the linear functions $a \cdot x$ [17]:

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

A Boolean function f is *balanced* if it takes the value 1 exactly the same number 2^{n-1} of times as the value 0 when the input ranges over $\mathbb{F}_2^n$.

x_2	x_1	x_0	TT
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table 1: Truth table of a Boolean function with 3 inputs.

The minimum Hamming distance between a Boolean function f and all affine functions (in the same number of variables as f) is called the nonlinearity of f. The nonlinearity Nl_f of a Boolean function f can be expressed in terms of the Walsh-Hadamard coefficients as [17]:

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

In Table 1, we give an example of a Boolean function with 3 inputs. Clearly, this function is balanced as it has the same number of zeros and ones in the truth table representation (TT column).

In Table 2, we give an example of Walsh-Hadamard calculation of the Boolean function from Table 1. Notice that to conform with Eq. (1), instead of TT, we write $f(x)$. Also, while we write a values as integers, they should be considered as binary values. Finally, from column $W_f(a)$, we see that the maximal absolute Walsh-Hadamard spectrum value equals 4, which means that nonlinearity equals 2 as per Eq. (2) ($2^2 - \frac{1}{2} \cdot 4 = 2$).

The maximal value of the Walsh-Hadamard spectrum equals at least $2^{n/2}$, which occurs in the case of bent Boolean functions [1]. Bent functions cannot be balanced, as their Hamming weight equals $2^n - 1 \pm 2^{\frac{n}{2}-1}$. Bent functions exist only for n even. The nonlinearity of bent functions equals [1, 18]:

$$Nl_f = 2^{n-1} - 2^{\frac{n}{2}-1}. \tag{3}$$

The nonlinearity of a Boolean function with n variables is bounded above by $2^{n-1} - 2^{\frac{n}{2}-1}$ (the Covering Radius Bound). Clearly, this bound cannot be

a	$f(x)$	$W_f(a)$
0	1	$(-1)^1 + (-1)^1 + (-1)^0 + (-1)^1 + (-1)^0 + (-1)^0 + (-1)^1 + (-1)^0 = 0$
1	1	$(-1)^1 + (-1)^0 + (-1)^0 + (-1)^0 + (-1)^0 + (-1)^1 + (-1)^1 + (-1)^1 = 0$
2	0	$(-1)^1 + (-1)^1 + (-1)^1 + (-1)^0 + (-1)^0 + (-1)^0 + (-1)^0 + (-1)^1 = 0$
3	1	$(-1)^1 + (-1)^0 + (-1)^1 + (-1)^1 + (-1)^0 + (-1)^1 + (-1)^0 + (-1)^0 = 0$
4	0	$(-1)^1 + (-1)^1 + (-1)^0 + (-1)^1 + (-1)^1 + (-1)^1 + (-1)^0 + (-1)^1 = -4$
5	0	$(-1)^1 + (-1)^0 + (-1)^0 + (-1)^0 + (-1)^1 + (-1)^0 + (-1)^0 + (-1)^0 = 4$
6	1	$(-1)^1 + (-1)^1 + (-1)^1 + (-1)^0 + (-1)^1 + (-1)^1 + (-1)^1 + (-1)^0 = -4$
7	0	$(-1)^1 + (-1)^0 + (-1)^1 + (-1)^1 + (-1)^1 + (-1)^0 + (-1)^1 + (-1)^1 = -4$

Table 2: Calculation of the Walsh-Hadamard spectrum for a Boolean function with 3 inputs.

tight when n is odd, so for Boolean functions with an odd number of inputs, the maximal nonlinearity lies between $2^{n-1} - 2^{\frac{n-1}{2}}$ and $2^{n-1} - 2^{\frac{n}{2}-1}$.

While we consider here only two properties, balancedness and nonlinearity, they play important roles in different domains. For example, finding the covering radius for the Reed-Muller code of order one is equivalent to finding maximally nonlinear Boolean functions [17]. Note that balanced Boolean functions are used in cryptography and coding theory while bent functions are, for instance, used in sequences and mobile networks.

3. Applications of Boolean Functions

In the last few decades, there has been a number of papers considering heuristics and Boolean functions. A more careful study reveals that a large part of those works considers applications in cryptography, and we provide an overview in the following.

To the best of our knowledge, Millan et al. were the first to apply genetic algorithms (GAs) to the evolution of cryptographically suitable Boolean functions [19]. There, the authors experimented with GA to evolve Boolean functions with high nonlinearity. Later, Millan et al. [20] continued to use GA to evolve Boolean functions with high nonlinearity. In conjunction with the GA, they used hill climbing and a resetting step to find Boolean functions with even higher nonlinearity and sizes of up to 12 inputs. Dawson et al. [21] experimented with two-stage optimization to generate Boolean functions. They used a combination of simulated annealing and hill-climbing with a cost function motivated by the Parseval theorem to find functions with high nonlinearity and low autocorrelation. Kavut and Melek [22] developed improved cost functions for a search that combines simulated annealing and hill climbing. With that approach, the authors were able to find some functions of eight and nine inputs that have a combination of nonlinearity and autocorrelation values previously not obtained. Millan et al. [23] proposed a new adaptive strategy for the local search algorithm for the generation of Boolean functions with high nonlinearity. Hernan et al. [24] were the first to use a multi-objective random bit climber to search for balanced Boolean functions of size up to eight inputs with high nonlinearity. Picek et al. [25] experimented with genetic algorithms and genetic programming to find Boolean functions that possess several cryptographic properties. As far as we are aware, this is the first application of genetic programming to the evolution of Boolean functions with cryptographic properties. Mariot and Leporati [26] used Particle Swarm Optimization to find Boolean functions with good trade-offs of cryptographic properties for dimensions up to 12.

There have been several successful approaches where the authors could find bent Boolean functions for different dimensions. Hrbacek and Dvorak experimented with Cartesian genetic programming to evolve bent Boolean functions of size up to 16 inputs [27]. When considering combinatorial designs, Mariot et al. used evolutionary algorithms to design binary orthogonal arrays [14] and orthogonal Latin squares [15].

4. Analyzing Fitness Landscapes

Fitness landscapes describe the relationship between search and fitness space [28], thus a heuristic strategy can navigate a specific landscape structure searching for optimal solutions.

Several cost models have been used to make specific predictions for combinatorial problems, identifying which features of the fitness landscape contribute more to the problem solving complexity during the search. By identifying these features, some improvements regarding the algorithm performance can be designed.

The Local Optima Network (LON) [29] is a model designed to understand the local optima structure in combinatorial landscapes, incorporating network analysis techniques to study fitness landscapes and problem difficulty [30].

The fitness landscape in LON models is modeled as a graph where the local optima represent nodes that can be connected. A local search heuristic $\mathcal{H}$ maps the solution space S to the set of locally optimal solutions S^*. Given

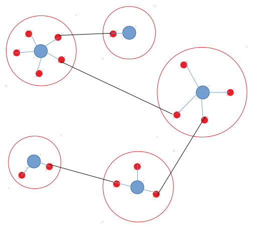

Figure 1: An example of the connectivity in local optima networks.

a fitness function F, a solution i in the solution space S is a local maximum according to a neighborhood operator N if $F(i) \geq F(s), \forall s \in N(i)$.

Each local optima i has an associated set of basin of attraction defined by $B_i = \{s \in S | \mathcal{H}(s) = i\}$. This set contains all the solutions that, after applying a local search starting from each of them, the procedure returns i. The cardinality of B_i is the size of the basin of attraction of i. Given a neighborhood operator, we assume a connection between two local optima if at least one solution in one basin has a neighbor solution in the other basin. This assumption is based on previous basin-edges models, which also do not consider weighted edges [29, 31, 32].

Figure 1 shows a simplified LON for visualization purposes, illustrating the basin of attraction (red circles), their local optima (big blue dots), the solutions that converge to the local optima when applying the local search (small red dots), and the edges between the local optima (black lines) that exist due to neighborhood. Note that sophisticated heuristics might result in many more interconnections between the basin of attraction than what we have presented in the example.

In early works, local optima networks were exhaustively extracted on representative NK landscape instances [33, 29]. Additionally, some works investigated the correlation between LON features and the performance of search heuristics [34, 35, 16].

Permutation-based problems have also been subject to LON analyzes [36]. Besides, [30] extended the LON modeling to neutral fitness landscapes. Neutral networks are connected networks of solutions of equal fitness, with possibly jumps between them. The authors study two neutral versions of the NK landscape model, tuning the amount of neutrality. The results confirmed that the study of neutrality could improve the heuristic search.

Recently, some works addressed a LON variant called Compressed Local Optima Network. The work proposed in [37] investigated fitness landscape properties for the Number Partitioning Problem, exploring whether the global landscape structure of the number partitioning problem changes with the phase transition. In [38], the authors analyzed the network features to find differences between the landscape structures for the Permutation Flowshop Scheduling Problem (PFSP). The results provided insights into which features impact the performance of an iterated local search heuristic.

The authors in [31] investigated two hill-climbing local search procedures for building their corresponding LONs. The LONs were analyzed to understand the difficulty of Travelling Thief Problem (TTP) instances. Among others, they found that certain operators can result in LONs with disconnected components and that at times potentially exploitable correlations of node degree, basin size, and fitness exist.

Using a similar methodology, the first landscape analysis in the greater field of security investigated cryptographic S-Boxes [32]. For the chosen fitness functions and two neighborhood operators (considered in isolation), it was observed that the number of local optima is substantial, and a conjecture has been made that links S-Boxes of odd dimensions to their problem difficulty.

Here, we use fitness landscape analysis to study the effects that algorithmic design decisions have on optimizing Boolean functions' two important properties. We consider three fitness functions, two initialization strategies, and three neighborhood operators – the latter in isolation and combination, resulting in seven different neighborhoods.

5. Creating Networks with Local Search

In order to obtain LONs of the Boolean function optimization landscape, we use a local search procedure that, starting from a given initial solution, converges to a corresponding local optimum. Along with the initial solution, all the intermediate solutions leading from the initial solution to the local optimum are added as the members of that local optimum's basin of attraction. This procedure is repeated for each solution in the set of initial solutions. After that, we record all unique local optima and reconstruct the connections between their basins of attraction.

The local search is described in Algorithm 1; it can be used with an arbitrary representation and an arbitrary neighborhood relationship, where $\mathcal{N}(.)$ represents the neighborhood of the given solution. In the local search, a new solution is accepted only if at least one solution with a better fitness

Algorithm 1 A greedy local search heuristic

```
1: $s \leftarrow$ initial solution
2: while there is an improvement do
3:     $s^* = s$
4:     for each $s^{**}$ in $\mathcal{N}(s)$ do
5:         if $F(s^{**}) > F(s^*)$ then
6:             $s^* \leftarrow s^{**}$
7:         end if
8:     end for
9:     $s = s^*$
10: end while
```

value is found within the entire neighborhood. Note that the algorithm is deterministic; if there are multiple solutions with the same fitness value, the algorithm will retain the first one that it encounters, while the ordering of the solutions in the neighborhood depends on the actual neighborhood relation. If no better solution is found in the initial solution neighborhood, the algorithm will not record the initial solution as a local optimum.

5.1. Neighborhood Operators

This study considers the truth table representation of Boolean functions, which is encoded as a bitstring. We opted to use the bitstring encoding, even though the related works usually report graph/tree encoding as the best performing one, due to two reasons. First, the properties we consider in our fitness functions are directly connected with the truth table representation. Consequently, exploring neighborhoods in the bitstring encoding gives a direct insight into the difficulty of the problem. Contrary, having a small change in an encoding like the tree encoding, which represents a Boolean function in the form of an expression, can cause a large change in the truth table, thus making the algorithmic design decisions more difficult. Second, we explore Boolean functions up to dimension 7 (i.e., when the search space is 2^{128}) and related works show that for such sizes, the bitstring encoding achieves the same performance [13].

With the bitstring encoding, we use three neighborhood variants within Algorithm 1:

1. The first (denoted "swap") uses the swap operation (also known as "toggle") to generate the neighborhood; the swap operation takes two different positions in the bitstring and exchanges them.

2. The second variant (denoted “flip”) flips the selected bit in the bitstring.
3. The third variant (denoted “insert”) uses the *insertion* operator; this operator takes a value out of the bitstring at a random position i and inserts it at another random position j, thus pushing the values between i and j by one spot to the right.

Also, to investigate complementary capabilities, we consider the following four combined neighborhoods: (1) swap/flip, (2) swap/insert, (3) flip/insert, and (4) swap/flip/insert. Whenever we consider any of these, e.g., swap/flip, we first construct the neighborhood for each operator in isolation, then merge them in the defined order (e.g., all the swap neighbors first, then all the insert neighbors), and then consider this sorted sequence as the combined neighborhood that is created by considering both operators at the same time. Some authors also considered combined strategies by proposing algorithms that use local search methods based on combined neighborhood operators [39]. However, they apply local search strategies sequentially, differently from our investigation: here, in each algorithm step we merge the solutions obtained simultaneously from all operators separately.

As we always consider the entire neighborhood before selecting the best, the order of the neighborhood-operators in these combinations does not matter, unless – as previously highlighted – the fitness of several solutions is identical. In that case, the algorithm will keep the first solution with the best fitness value it encounters, which favors first neighborhoods in the combination.

5.2. Initialization Strategies

In preliminary experiments, we observed that randomly sampled initial solutions for the subsequent hill-climbs result in very few edges in the final LONs. To give us a greater chance of observing connections in the LONs, and also for a more systematic approach, we consider “lexicographic” sampling (abbreviated: “lex”). This also starts with a random sample, but all subsequent samples continue in lexicographic order from the first sample, based on the binary representation.

5.3. Fitness Functions for Optimization of Boolean Functions

The first fitness function uses the nonlinearity value where the goal is to maximize it:

$$fitness_1 : Nl_f. \tag{4}$$

In the second fitness function, we aim to search for balanced, highly nonlinear functions. We use a two-stage fitness in which a fitness bonus equal to the nonlinearity is awarded only to a perfectly balanced function; otherwise, the fitness is only described by the balancedness penalty. The balancedness penalty BAL is defined as the difference up to the balancedness (i.e., the number of bits that need to be changed to reach balancedness) This difference is included in the fitness function with a negative sign to act as a penalty in maximization scenarios. The delta function $\delta_{BAL,0}$ takes the value one when $BAL = 0$ and is zero otherwise.

$$fitness_2 : -BAL + \delta_{BAL,0} \cdot Nl_f. \tag{5}$$

Finally, the third fitness function extends the second one to consider the whole Walsh-Hadamard spectrum and not only its extreme value:

$$fitness_3 : -BAL + \delta_{BAL,0} \cdot (Nl_f + Indicator). \tag{6}$$

The $Indicator$ property is the normalized number of occurrences of the maximal nonlinearity value in the whole spectrum (denoted $\#max\_values$). Naturally, the smaller the number of such maximal values, the easier it is for the algorithm to reach the next nonlinearity value: $Indicator = 2^n - \frac{\#max\_values}{2^n}$.

6. Results and Discussion

This section analyzes the local optima networks obtained using the local search heuristic to reveal insights about the search space structure. Furthermore, we study the basins of attraction and their relationship with some LON properties looking for additional search difficulty information.

In our experiments, we explore the following parameters of the search space, which represent various design decisions that need to be made when setting up a heuristic search for Boolean functions:

- Boolean functions of size $4 \leq n \leq 7$;
- three fitness functions (Equations (4), (5), and (6));
- two initialization strategies;
- seven neighborhood types (based on swap, flip, and insert);
- number of samples (unique initial solutions).

As it is possible to perform an exhaustive search for problem size $n = 4$ – because the total number of solutions is 2^{16} – we build the LONs by enumerating the search space. In other words, the Algorithm 1 is executed for every possible initial solution (every Boolean function in $n = 4$ variables). In larger sizes, we conduct a sampling process using a fixed sample size, which is the number of unique initial solutions: for each solution, we run Algorithm 1 until no further improvements are possible.[1]

Note that, because both our fitness functions (nonlinearity, balancedness, and the Walsh-Hadamard spectrum) and our neighborhood enumeration here require fully defined functions (so that we can enumerate the complete neighborhood), small structural changes would be necessary to transfer our approach to partially defined Boolean functions that do not define all 2^n possible solutions.

6.1. Topological Properties of Local Optima Networks

In Tables 3 and 4, we show graph properties that are often used for LON analyses [29]. In particular, we extract the following metrics. n_v and n_e represent the number of vertices (or nodes) and the number of edges of the generated LON, respectively. As in many other studies, we do not consider weights in the edges. z is the average degree. C is the average clustering coefficient. C_r is the average clustering coefficient of corresponding random graphs (i.e., random graphs with the same number of vertices and mean degree). b is the average basin size. l is the average shortest path length between any two local optima. π is the connectivity, which indicates if the LON is a connected graph. Finally, S is the number of connected components (sub-graphs).

In Table 3, we report on the exhaustive search for Boolean functions with size $n = 4$. We compare the three fitness functions $fitness_1$, $fitness_2$, and $fitness_3$ using the seven neighborhood operators. We find that the number of vertices (n_v) for $flip$, $swap$ is the same for the three functions, and this behavior also occurs with $swapflip$ and $swapflipinsert$, which indicates that the local optima and the distinct starting points are the same for these operators. However, except for $flip$ and $swap$ on the three functions, the actual LONs are quite different, with the number of edges (n_e) ranging between about 14 000 and 650 000.

The average degrees (z) are higher for combined neighborhoods than for the isolated operators. A higher number of edges (n_e) can also be noted for

[1]While it is possible to calculate the fitness values of all 2^{32} solutions in case of $n = 5$, it is not possible to conduct this many hill-climbs, including all neighborhood calculations, which are needed to create the networks.

the combined neighborhoods on $fitness_2$ and $fitness_3$. Besides, the $fitness_1$ function results in greater average degree than $fitness_2$ and $fitness_3$. Interestingly, the LON consists of only one component for almost all instances for $fitness_2$ and $fitness_3$ (with the exception of *flip* and *swap* for $fitness_2$ function). In combination with the observed high mean degree and small minimum distances between nodes, this can mean that a Tabu Search [40, 41] with restarts or a Memetic Algorithm [42] with built-in local searches, or even an approach with explicit niching might be able to perform well and explore the entire network.

Table 4 shows the results using 10 000 lexicographic-ordered samples (i.e., for $n \geq 5$), where a "sample" refers to a sampled starting point and a subsequent deterministic hill-climb. We typically find several hundreds of local optima. This indicates that there is a very large number of local optima in the landscape.[2]

Next, with the clustering coefficient (C) of a node i, we measure how close its neighbors are to being a clique, and it characterizes the extent to which nodes adjacent to node i are connected to each other. This determines, together with l, whether a graph is a small-world network (in which nodes are highly clustered yet the path length between them is small). We can observe in both tables that the LONs show a significantly higher degree of local clustering than their corresponding random graphs (C_r). This means that the local optima are connected in two ways: dense local clusters and sparse interconnections, which can be difficult to find and exploit for all operators. Besides this, all connected LONs in both tables have a small minimal path length l on average, i.e., any pair of local optima can be connected by traversing only a few other local optima.

Additionally, for $n = 4$, we briefly investigate the extent to which sampled landscapes are representative of the entire problem. To do so, we sample the landscapes, extract the graph properties, and calculate the correlations. The resulting graph properties can be seen in Table .6 in the Appendix. Table 5 reports the Spearman correlation coefficient between the sampled landscapes and the completely enumerated landscapes. When the correlation is higher than 0.4, then we highlight it in light blue. As one might expect, *random*

[2]We do not report on the results of the LONs based on random initial solutions (and the subsequent hill-climbs), as these consisted of hundreds or even thousands of disconnected components.

initialization can be used to roughly estimate of the number of components (S). Generally, for the *lex* initialization, the 10 000 samples results in higher correlations than for the 1 000 case; in some of the later experiments, we still consider 1 000 samples due to the size of the neighborhood. In detail, *lex* shows a high correlation coefficient (with the complete enumeration) for the degree, and it ranges between 0.4 and 0.5 for both clustering coefficient (C) and number of components (S).

Function	**Operator**	n_v	n_e	z	C	C_r	b	l	π	S
$fitness_1$	$flip$	12774	275388	43.1170	0.2672	0.0034	3.4656	–	0	9
	$insert$	12904	435761	67.5389	0.2771	0.0052	3.5939	–	0	7
	$swap$	12774	275388	43.1170	0.2672	0.0034	3.4656	–	0	9
	$flipinsert$	1182	150095	253.9679	0.4945	0.2151	54.7394	1.82	1	1
	$swapflip$	1176	103700	176.3605	0.4093	0.1500	55.0136	1.92	1	1
	$swapflipinsert$	1176	174167	296.2024	0.5072	0.2521	55.0136	1.77	1	1
	$swapinsert$	9806	533767	108.8654	0.2794	0.0111	4.5030	–	0	7
$fitness_2$	$flip$	1776	14249	16.0462	0.3082	0.0089	2.0878	–	0	3
	$insert$	1712	20581	24.0432	0.2956	0.0135	2.1379	3.73	1	1
	$swap$	1776	14249	16.0462	0.3082	0.0092	2.0878	–	0	3
	$flipinsert$	10922	471252	86.2941	0.1973	0.0079	6.0004	3.23	1	1
	$swapflip$	10920	401688	73.5692	0.2265	0.0067	6.0015	3.43	1	1
	$swapflipinsert$	10920	648497	118.7723	0.1955	0.0109	6.0015	2.94	1	1
	$swapinsert$	1484	29029	39.1226	0.2638	0.0260	2.3140	2.81	1	1
$fitness_3$	$flip$	2292	24305	21.2086	0.2648	0.0089	2.2094	3.79	1	1
	$insert$	2166	30553	28.2114	0.2604	0.0129	2.2872	3.53	1	1
	$swap$	2292	24305	21.2086	0.2648	0.0093	2.2094	3.79	1	1
	$flipinsert$	10082	472850	93.8008	0.2053	0.0093	6.5003	3.08	1	1
	$swapflip$	10080	398788	79.1246	0.2296	0.0079	6.5016	3.24	1	1
	$swapflipinsert$	10080	642179	127.4165	0.2012	0.0127	6.5016	2.83	1	1
	$swapinsert$	1866	47681	51.1050	0.2429	0.0271	2.4952	2.73	1	1

Table 3: General LON and basins' statistics for Boolean functions size 4. In each row, we show the LON that is the result of conducting a hill-climb from each of the possible 2^{16} unique solutions in the search space. A dash is shown when l cannot be computed as multiple disconnected components exist.

Size	**Initialization**	**Function**	**Operator**	n_v	n_e	z	C	C_r	b	l	π	S
5	lex	$fitness_2$	$flip$	730	6272	17.1836	0.3022	0.0221	13.6356	3.0883	1	1
			$insert$	260	4377	33.6692	0.4909	0.1296	3.8577	2.0222	1	1
			$swap$	145	1960	27.0345	0.5687	0.1888	6.1172	1.9519	1	1
			$flipinsert$	382	9569	50.0995	0.5323	0.1324	27.0969	1.9449	1	1
			$swapflip$	193	2948	30.5492	0.5772	0.1610	52.6528	1.9664	1	1
			$swapflipinsert$	280	7000	50.0000	0.5756	0.1784	36.6464	1.8604	1	1
			$swapinsert$	240	5373	44.7750	0.5505	0.1862	4.1458	1.8633	1	1
		$fitness_3$	$flip$	813	8055	19.8155	0.3190	0.0238	12.3456	2.9684	1	1
			$insert$	251	4355	34.7012	0.4918	0.1413	4.4542	1.9452	1	1
			$swap$	183	3238	35.3880	0.5245	0.1972	5.3497	1.8835	1	1
			$flipinsert$	414	11849	57.2415	0.5437	0.1395	25.8527	1.8909	1	1
			$swapflip$	277	6554	47.3213	0.5370	0.1714	37.6570	1.8891	1	1
			$swapflipinsert$	352	12190	69.2614	0.5584	0.1978	30.1477	1.8096	1	1
			$swapinsert$	255	6363	49.9059	0.5292	0.1951	4.3451	1.8267	1	1
6	lex	$fitness_2$	$flip$	363	3025	16.6667	0.3863	0.0450	27.5840	3	1	1
			$insert$	236	3536	29.9661	0.4187	0.1277	2.3136	2.1369	1	1
			$swap$	62	487	15.7097	0.4982	0.2505	5.9677	1.9334	1	1
			$flipinsert$	314	6550	41.7197	0.4724	0.1327	32.7229	2.0490	1	1
			$swapflip$	128	1698	26.5313	0.5045	0.2089	78.8047	1.9382	1	1
			$swapflipinsert$	221	4404	39.8552	0.5305	0.1815	46.0905	1.9461	1	1
			$swapinsert$	156	2608	33.4359	0.5128	0.2163	3.0128	1.8864	1	1
		$fitness_3$	$flip$	595	4822	16.2084	0.3481	0.0280	17.2185	3.1043	1	1
			$insert$	163	2283	28.0123	0.5252	0.1816	4.4479	1.9076	1	1
			$swap$	113	1396	24.7080	0.5608	0.2219	5.9735	1.8254	1	1
			$flipinsert$	422	9640	45.6872	0.4684	0.1076	26.3436	2.0016	1	1
			$swapflip$	249	4420	35.5020	0.4949	0.1437	43.8153	1.9196	1	1
			$swapflipinsert$	348	9459	54.3621	0.5046	0.1572	32.0920	1.8754	1	1
			$swapinsert$	177	3389	38.2938	0.5402	0.2185	4.7345	1.8073	1	1
7	lex	$fitness_2$	$flip$	512	4223	16.4961	0.3550	0.0329	19.3516	–	0	2
			$insert$	513	7742	30.1832	0.3696	0.0592	2.3294	–	0	2
			$swap$	30	154	10.2667	0.5846	0.3326	21.9333	2	1	1
		$fitness_3$	$flip$	639	5212	16.3130	0.3201	0.0250	15.7042	–	0	2
			$insert$	465	7152	30.7613	0.3926	0.0659	6.2839	3	1	1
			$swap$	75	610	16.2667	0.6362	0.2105	14.6267	2	1	1

Table 4: General LON and basins' statistics for 10 000 samples, with the lex initialization. We omit the random initialization, as the LONs always consisted of hundred to thousands of disconnected components. We also omit $fitness_1$ here as it has also resulted in disconnected components. For $n = 7$ the results for the combined neighborhoods are missing as they were too large to be computed on our systems. A dash is shown when l cannot be computed as multiple disconnected components exist.

Samples	Initialization	n_v	n_e	z	C	C_r	b	l	π	S
$1,000$	lex	-0.0623	0.1891	0.8638	0.1670	0.0675	0.2181	-0.2018	0.2582	0.3173
	random	-0.0172	0.2468	0.1955			-0.2335			0.2709
$10,000$	lex	-0.1807	0.2073	0.9210	0.5187	0.2338	0.4150	0.1632	0.3920	0.3738
	random	-0.0046	0.3290	0.3451	-0.3966		-0.2441			0.4353

Table 5: Spearman correlation coefficient between exhausted and sampled landscapes for $n = 4$ for both *lex* and *random* initialization with 1 000 and 10 000 samples. Highlighted values present correlation higher than 0.4.

Figures 2 to 7 present the obtained networks for Boolean functions with size $n = 4$ to $n = 7$. We can see in Figures 2, 4, and 6 that swap and insert LONs, for example, present local dense connected components for $fitness_1$ function, while, in Figures 3, 5, and 7 for flipinsert, swapflipinsert, swapflip, and swapinsert, for examples, LONs are connected graphs for almost all fitness functions (except in swapinsert for $fitness_1$). The LONs for $n = 5$, $n = 6$, and $n = 7$ with *lex* initialization using 10 000 samples are connected graphs for almost all fitness functions and operators. For this reason, we suppressed them in this paper.

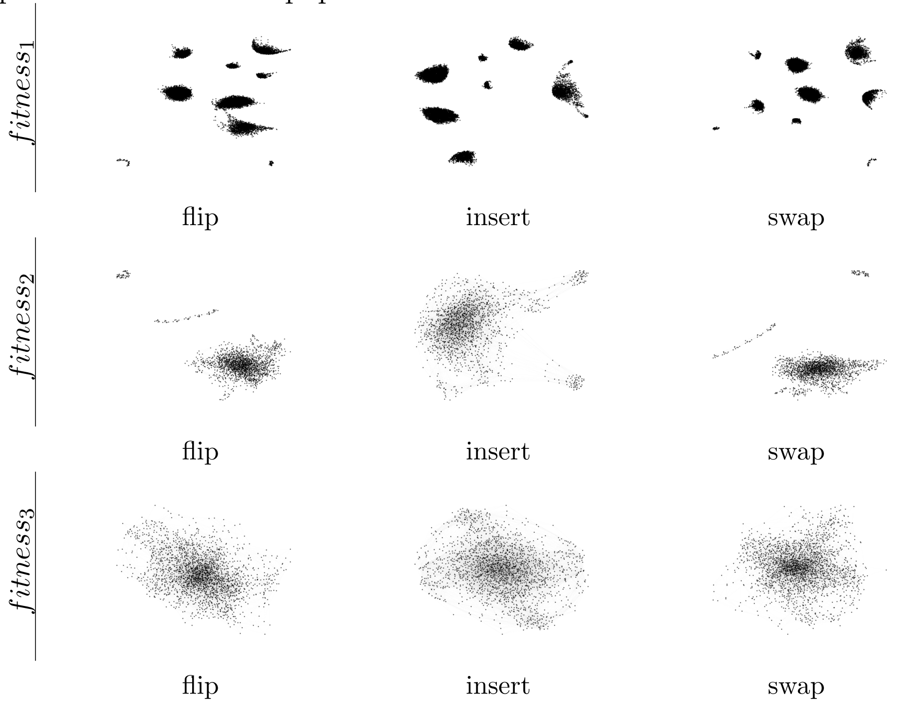

Figure 2: LON graphs on exhaustive $n = 4$ with the three fitness functions for operators flip, insert, swap.

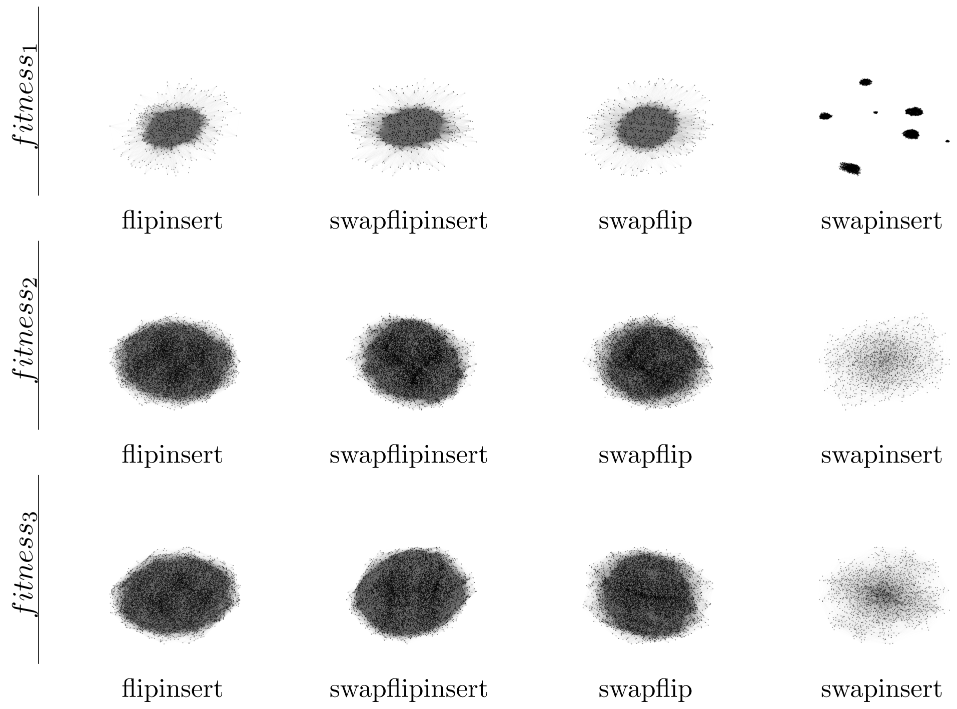

Figure 3: LON graphs on exhaustive $n = 4$ with the three fitness functions for combination using operators flipinsert, swapflipinsert, swapflip, and swapinsert.

6.2. Distribution of Degree

To characterize the networks visually, we provide three types of plots for our search space: (1) the cumulative degree distribution; (2) the correlation between the degree of local optima and their corresponding basin sizes; and (3) the correlation between the fitness of local optima and their corresponding basin sizes.

For the first one, the cumulative degree distribution function represents the probability $P(k)$ that a random node has a degree larger than k.

Let us start with $n = 4$. In the left columns of Figure 8 (for neighborhoods in isolation) and of Figure 9 (for neighborhoods in combination), we can see that the degree distributions hardly decay for small degrees for all the three single operators type and fitness functions, while their dropping rate is very high for high degrees, presenting short tails to the right. This behavior shows that there are few nodes with a large number of neighbors. However, most parts of the local optima have a small number of connections. A benefit of these few nodes with high connectivity is that these efficiently connect the entire landscape: a search at a random node has more chances to move to

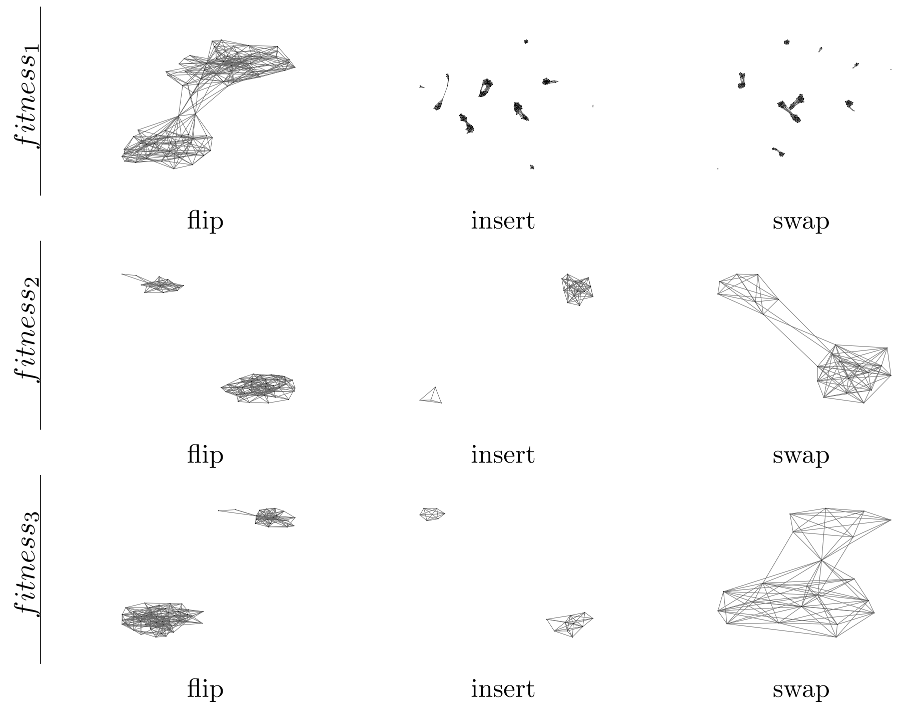

Figure 4: LON graphs for *lex* initialization on $n = 5$ with fitness functions $fitness_1$, $fitness_2$ and $fitness_3$ using 1 000 samples for all three operators: flip, insert, swap.

one of these high degree nodes, and then to another node, which can be an efficient way to search the entire network.

Local search strategies on networks have been investigated according to the degree distribution [43], particularly because some real-world network present properties in the topological structure that can be described by a power-law, or a scale-free degree distribution $P(k) = k^{-\alpha}$, where $\alpha \in [2, 3]$ is a scaling parameter.

Aiming to study the cumulative degree distribution more strictly, we use the Kolmogorov-Smirnov test to investigate the adequacy of power-law [44] and exponential models [45][3]. The test is performed on all distributions shown with a significance level of 0.1. When the $p - value > 0.1$, the test fails to reject power-law and exponential as plausible distribution models.

Considering the distributions reported in Figure 8 for $n = 4$, none of them fits power-law nor exponential models. For Figures .13, .14, .15, and .16 (see Appendix) with 1 000 samples, the $n = 6$ instances using lexicographic

[3]originally proposed by [29] to describe the degree distributions for NK models

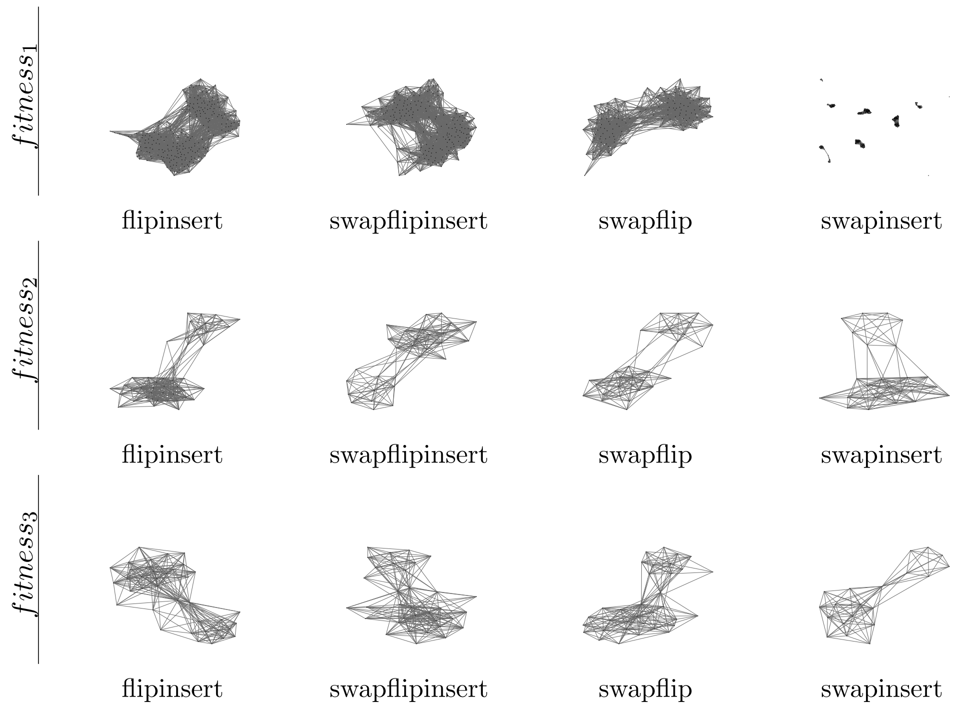

Figure 5: LON graphs for lex initialization on $n = 5$ with fitness functions $fitness_1$, $fitness_2$ and $fitness_3$ using 1 000 samples for combination using operators flipinsert, swapflipinsert, swapflip, and swapinsert.

sampling (lex) and $fitness_2$ function type for flip operator, and $fitness_1$ function type for swap operator, fit a power-law. For Figures .17, .18, .19, and .20 (see Appendix) with 10 000 samples the $n = 5$ instances for $fitness_2$ function type using lexicographic sampling (lex) for flip operator fit a power-law, as well as for the same instance considering the $fitness_3$ function type. The remaining instances do not fit a power-law nor an exponential model.

The degree distribution contributes to search a power-law graph more rapidly, assuming that the number of edges per node varies from node to node, i.e., its edges do not allow us uniformly sample the graph, but they preferentially lead to high degree nodes [31]. This means that a landscape with few nodes and a high degree enables that a search at a given node chosen at random presents more chances to move to one of these high degree nodes instead to another node, which can efficiently search the entire network.

To summarize our analyzes of degree distributions, as most instances cannot be represented with the straightforward interpretation from power-law models, another way to analyze the difficulty of the search space for the heuristics is to consider the size of the basins of attraction – which we will

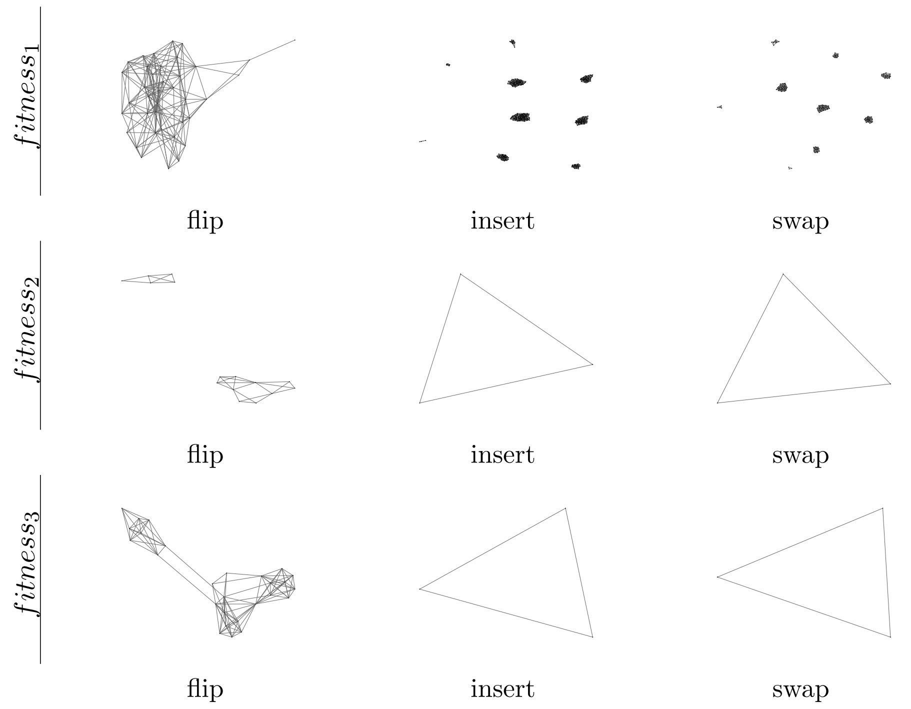

Figure 6: LON graphs for *lex* initialization on $n = 6$ with fitness functions $fitness_1$, $fitness_2$ and $fitness_3$ using 1 000 samples for all three operators: flip, insert, swap.

explore next.

6.3. Basin Size Correlation

Our "matrices of plots" present the basin correlation in the middle and right columns, i.e., Figure 8 and Figure 9 show this for $n = 4$, and Figures 10, 11, and 12 show these for the larger values of n. A particular focus of the last three mentioned figures is the difference of 1 000 samples to 10 000 samples.

Let us again start with $n = 4$ in Figures 8 and 9. Firstly, flips and swaps seem to result in almost perfectly identical LONs. This is interesting, as the flips generate neighbors with the same Hamming distance to the original, and swaps generate neighbors with the Hamming distances 0 or 2.

Secondly, the swapinsert (green) neighborhood typically results in very different LONs, as we have already seen in the earlier tables: the local optima are significantly less interconnected. This might result from using two operators that result in neighbors with the Hamming distance 0 or 2 in combination with the lexicographic sampling. Also, it appears that the use of the flip operator results in significantly greater basins of attraction – however, as

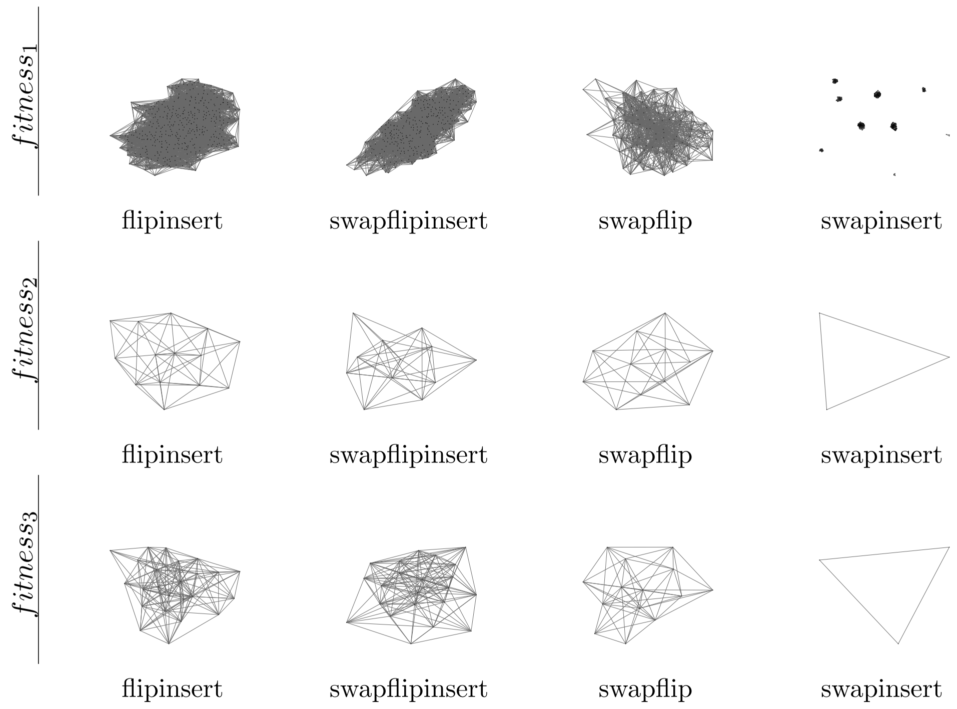

Figure 7: LON graphs for lex initialization on $n = 6$ with fitness functions $fitness_1$, $fitness_2$ and $fitness_3$ using 1 000 samples for combination using operators flipinsert, swapflipinsert, swapflip, and swapinsert.

we are using the lexicographic sampling of the initial solutions, this comes to no big surprise.

Thirdly, we can observe that the three different fitness functions resulted in quite different landscapes, and in particular, $fitness_1$ is quite different from the other two. We can see some correlations for $fitness_1$ of basin size with degree and fitness. At first sight, it is not clear how this information can be used in a heuristic. However, if techniques like self-adaptation and restarts are used in combination with $fitness_1$, then the progress achieved over time can be used in online control to indicate the expected achievable solution quality. Moreover, it should be possible to estimate this characteristic in a search heuristic with restarts to influence the amount of perturbation that is performed on the current best solution (i.e., to provide the first solution for the next hill-climb): if a solution resulted after a local search presents poor fitness, then a not-too-small perturbation should be applied to determine the initial point for the next run, aiming to increase chances to escape the small, bad basin of attraction. Note that the opposite does not hold, meaning that a large perturbation does not guarantee success. For $fitness_2$ and $fitness_3$, however, the correlation is a lot weaker and hence might be difficult to exploit.

Lastly, let us highlight a few interesting aspects of the landscapes when n is larger, i.e., in Figures 10, 11, and 12. For example, we can observe (except in the case of $n = 6$) that the distance from the black distributions to the blue ones (factor 10 increase in samples) is roughly the same across all experiments -— in particular, this applies (roughly) to both dimensions in all three figures. If the increase would be limited to a shift in the y-axis, then this would mean that the 10-fold increase in samples does not uncover different structures (as expressed in different degree distributions) in the landscape. However, the increase along the x-axis means that the rate of uncovering new structures is relatively stable. We believe that the number of samples has yet to be further increased as the degree distributions do not show signs of convergence yet. $n = 6$ with swaps or inserts shows significantly different behavior, and it might be the case that substructures have not been discovered during a local search that resulted in interconnections between the local optima. This warrants additional future research.

Besides this, we can generally observe good correlations of degrees and basin sizes when inserts or swaps are used for $n = 5$ and $n = 6$. We can also observe that for $n = 7$ only inserts seem to provide a decent correlation of degrees and basin sizes that might be exploitable, as mentioned above. Also, as before, the fitness of local optima seems to only carry some possibly exploitable information in the case of flips.

These experimental results can summarize some insights regarding the search improvements. The topological properties for $fitness_2$ and $fitness_3$ are LON of only one component for almost all instances, presenting high mean degree and small minimum distances between nodes. Besides, the local optima are connected as dense local clusters and sparse interconnections, which can be difficult to exploit. Some heuristics such as Tabu Search with restarts or a Memetic Algorithm with built-in local searches, or even an approach with explicit niching might be able to explore the entire network searching for promising solutions. According to the basin distribution there are few nodes with a large number of neighbors connecting the entire landscape: a search at a random node has more chances to move to one of these high degree nodes, and then to another node, which can be an efficient way to search the entire network. Moreover flip operator seems to provide more information using basin size and fitness of local optima correlation than the others operators.

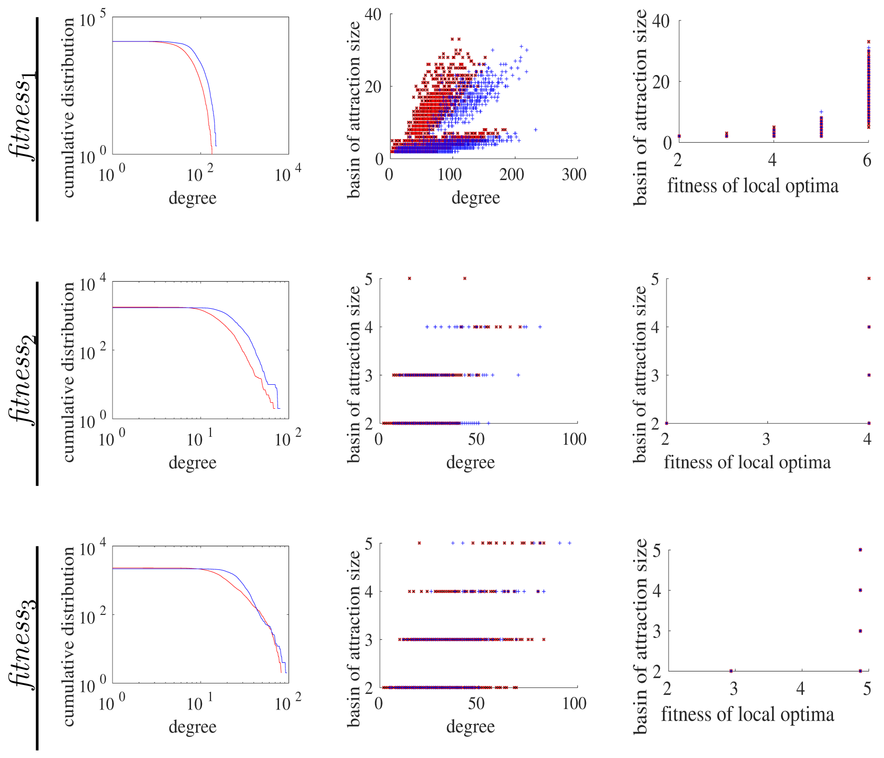

Figure 8: Statistical measures on exhaustive $n = 4$ with the three fitness functions for operators flip (black), insert (blue), swap (red): Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and their corresponding basin sizes (right).

7. Conclusions

Boolean functions are interesting mathematical objects that are widely used in many domains. Heuristics play an important role in their construction. However, little is known about the actual problem difficulty and the effect of various design decisions. This paper conducted a fitness landscape analysis (FLA) to study the effect of various decisions on the optimization of cryptographic properties. We investigated Boolean functions considering a different number of function sizes, three fitness functions, seven neighborhood operators used in isolation as well as in combination, and two initialization strategies.

We presented and analyzed the local optima networks (LONs) obtained

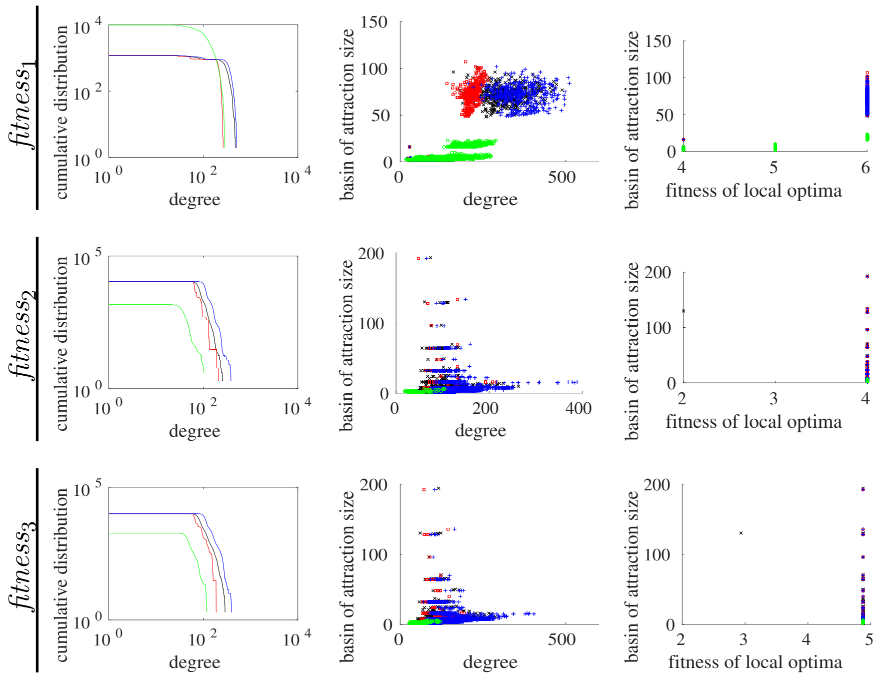

Figure 9: Statistical measures on exhaustive $n = 4$ with the three fitness functions for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green): Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and their corresponding basin sizes (right).

using a local search heuristic to investigate the search space structure. Furthermore, we studied the degree distribution and the correlation between the basins of attraction with some LON properties looking for additional information about the search difficulty, considering scenarios (e.g., combinations of neighborhoods) not investigated before.

We have observed (1) that the naive fitness function results in LONs with disconnected components, (2) which can typically be avoided by moving to other fitness functions. However, (3) we then appear to lose a correlation of basin size and LON degrees. (4) For our largest $n = 7$ (i.e., when the search space is 2^{128}), inserts appear to provide the largest possible exploitable correlation of basin sizes, local optima degrees, and local optima fitness.

In this paper, we concentrated on Boolean functions with 4 to 7 variables. In future work, we plan to extend our analysis up to 10 variables (i.e., up

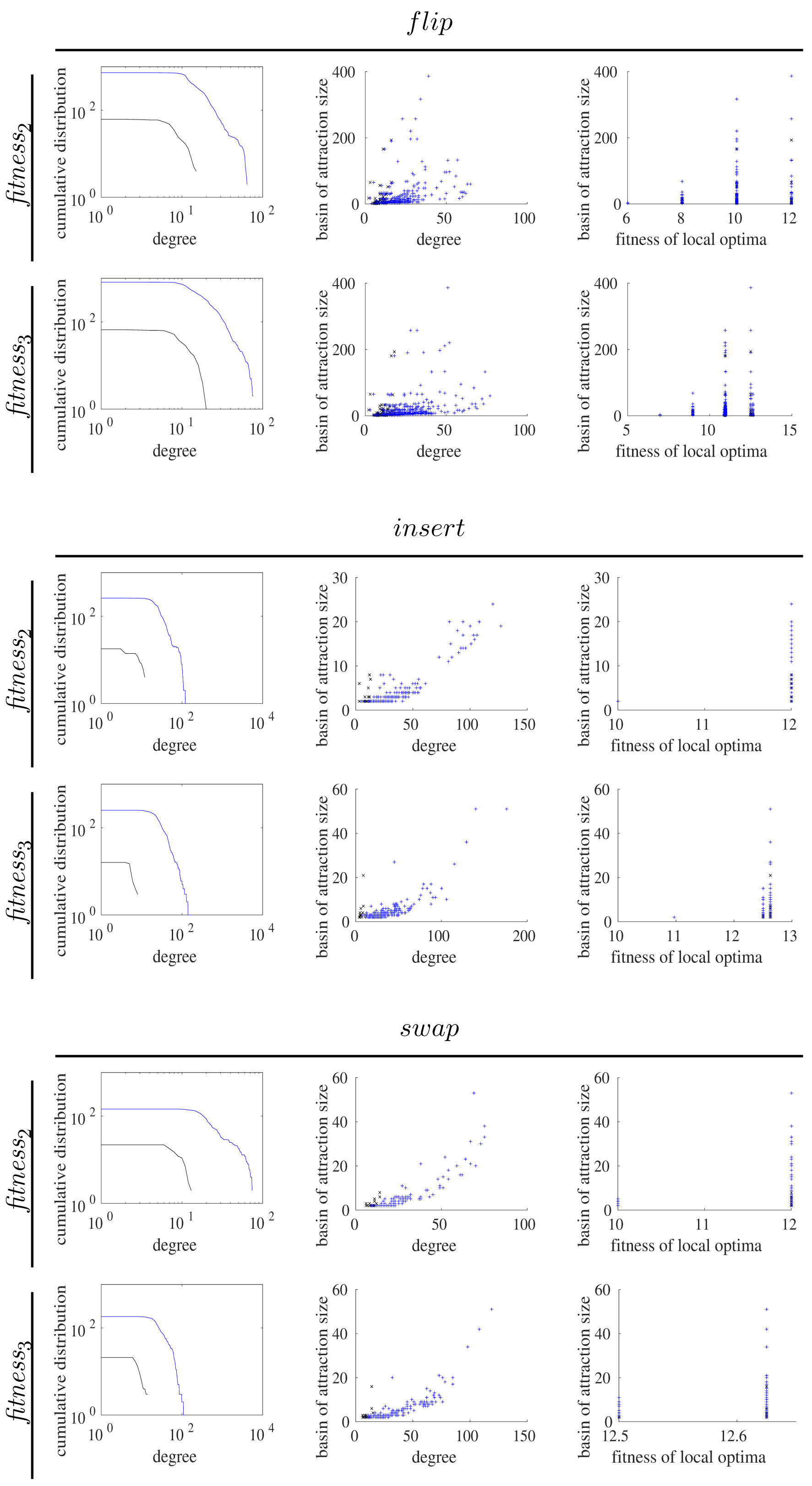

Figure 10: Statistical measures for *lex* initialization on $n = 5$ with fitness functions $fitness_2$ and $fitness_3$ using 1 000 (black) and 10 000 (blue) samples for all three operators: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and their corresponding basin sizes (right).

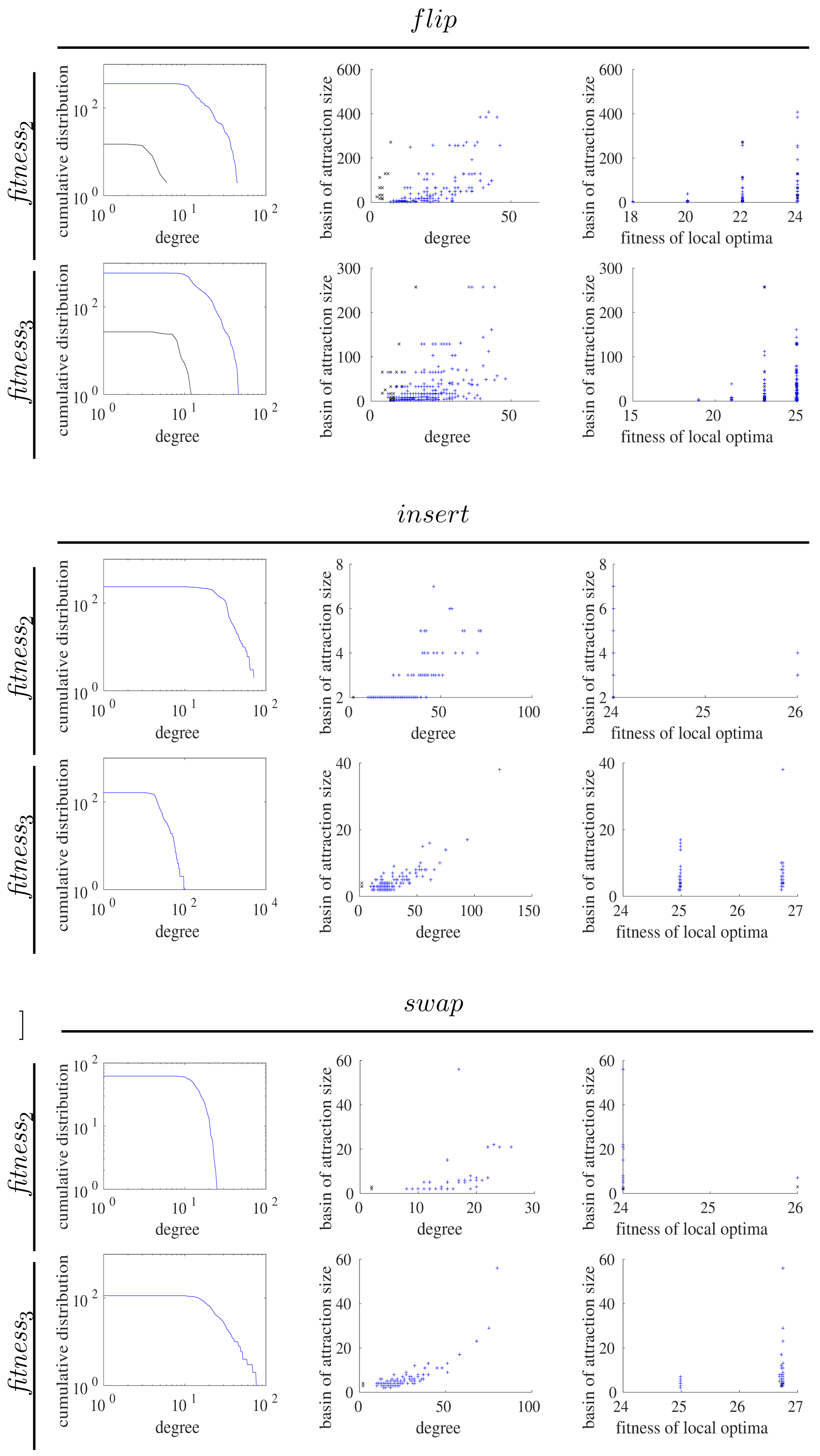

Figure 11: Statistical measures for *lex* initialization on $n = 6$ with fitness functions $fitness_2$ and $fitness_3$ using 1 000 (black) and 10 000 (blue) samples for all three operators: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and their corresponding basin sizes (right). Note that, in a few cases in the left column, no black line can be drawn as not enough points exist for a log-log plot.

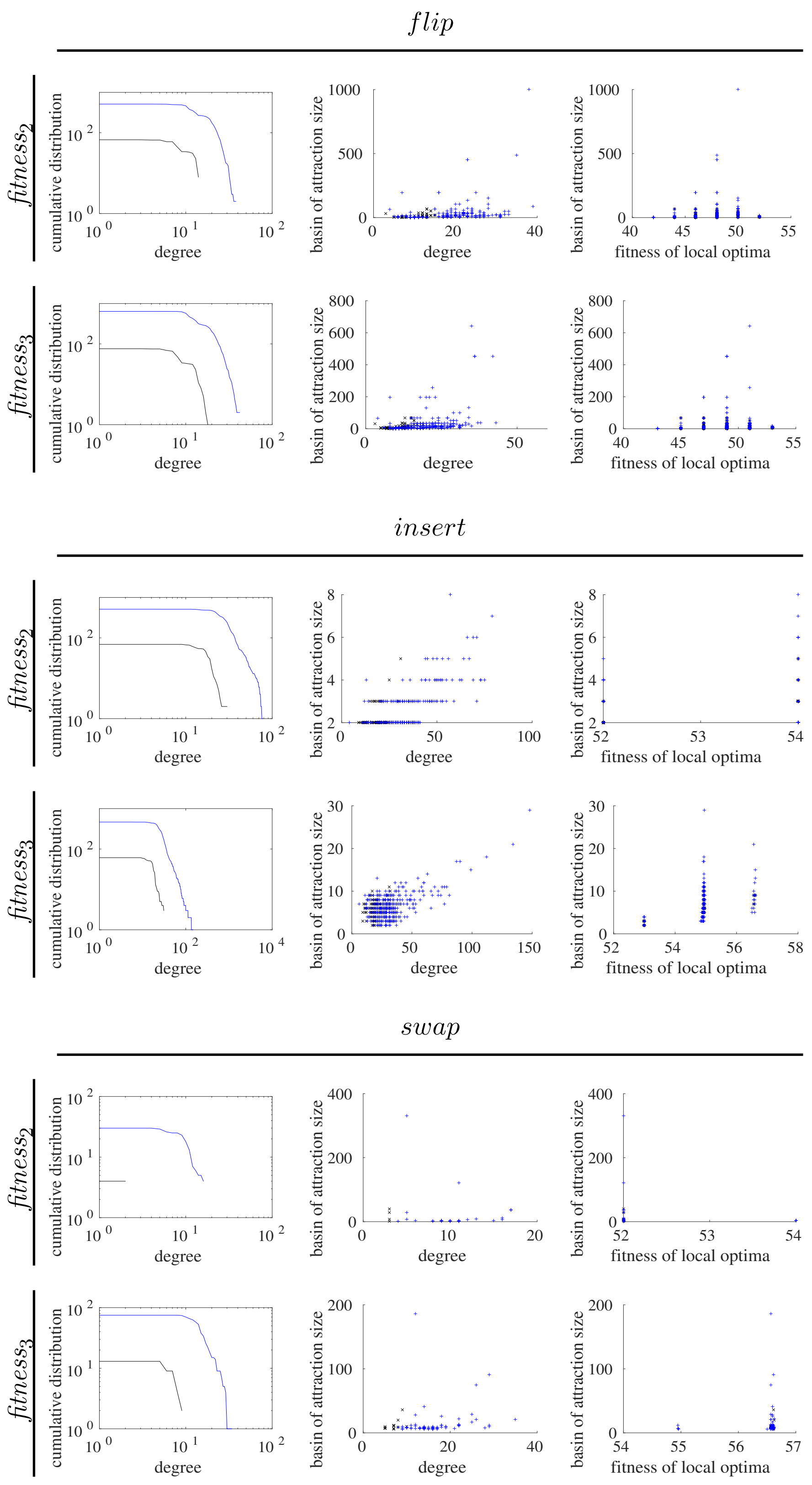

Figure 12: Statistical measures for *lex* initialization on $n = 7$ with fitness functions $fitness_2$ and $fitness_3$ using 1 000 (black) and 10 000 (blue) samples for all three operators: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and their corresponding basin sizes (right).

to 1 024 bits), which will require new approaches for evaluating complete neighborhoods.

Acknowledgments

Our work was supported by the Australian Research Council projects DE160100850, DP200102364, and DP210102670. Parts of our work have been inspired by COST Action CA15140 supported by COST (European Cooperation in Science and Technology).

References

[1] O. Rothaus, On "bent" functions, Journal of Combinatorial Theory, Series A 20 (3) (1976) 300 – 305.

[2] A. Bernasconi, B. Codenotti, J. M. Vanderkam, A characterization of bent functions in terms of strongly regular graphs, IEEE Transactions on Computers 50 (9) (2001) 984–985.

[3] X. Huang, K. Fang, L. Fang, Q. Chen, Z.-R. Lai, L. Wei, Bi-kronecker functional decision diagrams: A novel canonical representation of boolean functions, in: AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 2867–2875.

[4] S. Kavut, S. Maitra, M. D. Yucel, Search for boolean functions with excellent profiles in the rotation symmetric class, IEEE Transactions on Information Theory 53 (5) (2007) 1743–1751.

[5] A. Kerdock, A class of low-rate nonlinear binary codes, Information and Control 20 (2) (1972) 182 – 187.

[6] J. Olsen, R. Scholtz, L. Welch, Bent-function sequences, IEEE Trans. on Information Theory 28 (6) (1982) 858–864.

[7] K. Paterson, On Codes With Low Peak-to-Average Power Ratio for Multicode CDMA, IEEE Transactions on Information Theory 50 (2004) 550 – 559.

[8] M. Hell, T. Johansson, A. Maximov, W. Meier, A stream cipher proposal: Grain-128, in: IEEE Int. Symposium on Information Theory, 2006, pp. 1614–1618.

[9] C. M. Adams, Constructing symmetric ciphers using the cast design procedure, Designs, Codes and Cryptography 12 (3) (1997) 283–316.

[10] Y. Zheng, J. Pieprzyk, J. Seberry, HAVAL — a one-way hashing algorithm with variable length of output (extended abstract), in: Advances in Cryptology — AUSCRYPT '92: Workshop on the Theory and Application of Cryptographic Techniques, Springer, 1993, pp. 81–104.

[11] S. Picek, D. Sisejkovic, V. Rozic, B. Yang, D. Jakobovic, N. Mentens, Evolving cryptographic pseudorandom number generators, in: Parallel Problem Solving from Nature (PPSN), Springer, 2016, pp. 613–622.

[12] S. Picek, D. Jakobovic, J. F. Miller, E. Marchiori, L. Batina, Evolutionary Methods for the Construction of Cryptographic Boolean Functions, in: 18th European Conference on Genetic Programming (EuroGP), 2015, pp. 192–204.

[13] S. Picek, C. Carlet, S. Guilley, J. F. Miller, D. Jakobovic, Evolutionary algorithms for boolean functions in diverse domains of cryptography, Evolutionary Computation 24 (4) (2016) 667–694. doi:10.1162/EVCO_a_00190.
URL `https://doi.org/10.1162/EVCO_a_00190`

[14] L. Mariot, S. Picek, D. Jakobovic, A. Leporati, Evolutionary search of binary orthogonal arrays, in: Parallel Problem Solving from Nature – PPSN XV, Springer, 2018, pp. 121–133.

[15] L. Mariot, S. Picek, D. Jakobovic, A. Leporati, Evolutionary algorithms for the design of orthogonal latin squares based on cellular automata, in: Genetic and Evolutionary Computation Conference (GECCO), ACM, 2017, pp. 306–313.

[16] G. Ochoa, S. Verel, F. Daolio, M. Tomassini, Local optima networks: A new model of combinatorial fitness landscapes, in: Recent Advances in the Theory and Application of Fitness Landscapes, Springer, 2014, pp. 233–262.

[17] C. Carlet, Boolean functions for cryptography and error correcting codes, Boolean models and methods in mathematics, computer science, and engineering 2 (2010) 257–397.

[18] J. Dillon, A Survey of Bent Functions*, Tech. rep., Reprinted from the NSA Technical Journal. Special Issue, unclassified (1972).

[19] W. Millan, A. Clark, E. Dawson, An Effective Genetic Algorithm for Finding Highly Nonlinear Boolean Functions, in: First Int. Conference on Information and Communication Security, ICICS '97, Springer, 1997, pp. 149–158.

[20] W. Millan, A. Clark, E. Dawson, Heuristic design of cryptographically strong balanced Boolean functions, in: Advances in Cryptology - EUROCRYPT '98, 1998, pp. 489–499.

[21] J. A. Clark, J. L. Jacob, Two-Stage Optimisation in the Design of Boolean Functions, in: Information Security and Privacy, Vol. 1841 of LNCS, Springer, 2000, pp. 242–254.

[22] S. Kavut, M. D. Yücel, Improved Cost Function in the Design of Boolean Functions Satisfying Multiple Criteria, in: Progress in Cryptology - INDOCRYPT 2003, Vol. 2904 of LNCS, Springer, 2003, pp. 121–134.

[23] W. Millan, J. Fuller, E. Dawson, New concepts in evolutionary search for Boolean functions in cryptology, Computational Intelligence 20 (3) (2004) 463–474.

[24] H. Aguirre, H. Okazaki, Y. Fuwa, An Evolutionary Multiobjective Approach to Design Highly Non-linear Boolean Functions, in: Genetic and Evolutionary Computation Conference (GECCO), 2007, pp. 749–756.

[25] S. Picek, D. Jakobovic, M. Golub, Evolving Cryptographically Sound Boolean Functions, in: Genetic and Evolutionary Computation Conference (GECCO), GECCO '13 Companion, ACM, 2013, pp. 191–192.

[26] L. Mariot, A. Leporati, Heuristic Search by Particle Swarm Optimization of Boolean Functions for Cryptographic Applications, in: Genetic and Evolutionary Computation Conference, GECCO, Companion Material Proceedings, 2015, pp. 1425–1426.

[27] R. Hrbacek, V. Dvorak, Bent Function Synthesis by Means of Cartesian Genetic Programming, in: Parallel Problem Solving from Nature - PPSN XIII, Vol. 8672 of LNCS, Springer, 2014, pp. 414–423.

[28] S. Kauffman, S. Levin, Towards a general theory of adaptive walks on rugged landscapes, Journal of Theoretical Biology 128 (1) (1987) 11–45.

[29] G. Ochoa, M. Tomassini, S. Vérel, C. Darabos, A study of NK landscapes' basins and local optima networks, in: Genetic and Evolutionary Computation Conference (GECCO), ACM, 2008, pp. 555–562.

[30] S. Vérel, F. Daolio, G. Ochoa, M. Tomassini, Local optima networks with escape edges., in: Artificial Evolution, Springer, 2011, pp. 49–60.

[31] M. E. Yafrani, M. S. R. Martins, M. E. Krari, M. Wagner, M. R. B. S. Delgado, B. Ahiod, R. Lüders, A fitness landscape analysis of the travelling thief problem, in: Genetic and Evolutionary Computation Conference (GECCO), ACM, 2018, pp. 277–284.

[32] D. Jakobovic, S. Picek, M. S. Martins, M. Wagner, A characterisation of s-box fitness landscapes in cryptography, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2019, pp. 285–293.

[33] M. Tomassini, S. Verel, G. Ochoa, Complex-network analysis of combinatorial spaces: The NK landscape case, Physical Review E 78 (6) (2008) 066114.

[34] F. Chicano, F. Daolio, G. Ochoa, S. Vérel, M. Tomassini, E. Alba, Local optima networks, landscape autocorrelation and heuristic search performance, Parallel Problem Solving from Nature (PPSN) (2012) 337–347.

[35] F. Daolio, S. Verel, G. Ochoa, M. Tomassini, Local optima networks and the performance of iterated local search, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2012, pp. 369–376.

[36] F. Daolio, S. Verel, G. Ochoa, M. Tomassini, Local optima networks of the permutation flow-shop problem, in: International Conference on Artificial Evolution (Evolution Artificielle), Springer, 2013, pp. 41–52.

[37] G. Ochoa, N. Veerapen, F. Daolio, M. Tomassini, Understanding phase transitions with local optima networks: number partitioning as a case study, in: European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2017, pp. 233–248.

[38] L. Hernando, F. Daolio, N. Veerapen, G. Ochoa, Local optima networks of the permutation flowshop scheduling problem: Makespan vs. total flow time, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1964–1971.

[39] F. Chicano, D. Whitley, G. Ochoa, R. Tinós, Optimizing one million variable NK landscapes by hybridizing deterministic recombination and local search, in: Genetic and Evolutionary Computation Conference (GECCO), ACM, 2017, pp. 753–760.

[40] F. Glover, Tabu search - Part I, ORSA Journal on Computing 1 (3) (1989) 190–206.

[41] F. Glover, Tabu search - Part II, ORSA Journal on Computing 2 (1) (1990) 4–32.

[42] P. Moscato, New ideas in optimization, McGraw-Hill Ltd., UK, 1999, Ch. Memetic Algorithms: A Short Introduction, pp. 219–234.

[43] L. A. Adamic, R. M. Lukose, B. A. Huberman, Local search in unstructured networks, Handbook of Graphs and Networks: from the Genome to the Internet (2006).

[44] A. Clauset, C. R. Shalizi, M. E. Newman, Power-law distributions in empirical data, SIAM Review 51 (4) (2009) 661–703.

[45] W. Deng, W. Li, X. Cai, Q. A. Wang, The exponential degree distribution in complex networks: Non-equilibrium network theory, numerical simulation and empirical data, Physica A: Statistical Mechanics and its Applications 390 (8) (2011) 1481–1485.

Appendix with Supplemental Material

We make use of the appendix in order to provide additional visualizations of the results. In particular:

1. $n = 4$: Table .6 shows the metrics when considering the sample process for $n = 4$ for both lex and random initialization with 1 000 and 10 000 samples.

2. $n = 5$: Figure .13 shows the difference between the two initialization strategies, which we had not visualized before. Figure .14 shows the complex neighborhoods for 1 000 samples.

3. $n = 6$: Figure .15 shows the individual neighborhoods for 1 000 samples. Figure .16 shows the complex neighborhoods for 1 000 samples.

4. Figures .17-.20 show, for 10 000 samples, the results for the neighborhoods in isolation and in combination for $fitness_2$ and $fitness_3$.

Samples	**Initialization**	**Function**	**Operator**	n_v	n_e	z	C	C_r	b	l	π	S
1 000	lex	$fitness_1$	$flip$	121	520	8.5950	0.3526	0.0858	9.0000	–	0	2
			$insert$	438	3363	15.3562	0.5590	0.0381	2.8607	–	0	13
			$swap$	306	1948	12.7320	0.6407	0.0424	4.7516	–	0	14
			$flipinsert$	182	2494	27.4066	0.5753	0.1508	6.6374	2.0764	1	1
			$swapflip$	306	3514	22.9673	0.4806	0.0763	4.7516	3.0680	1	1
			$swapflipinsert$	363	5279	29.0854	0.4869	0.0807	4.3140	2.9272	1	1
			$swapinsert$	363	3325	18.3196	0.6810	0.0492	4.3140	–	0	11
		$fitness_2$	$flip$	185	849	9.1784	0.3165	0.0494	5.2973	–	0	2
			$insert$	626	5660	18.0831	0.4115	0.0297	2.0272	–	0	11
			$swap$	379	2236	11.7995	0.5123	0.0292	4.3140	–	0	20
			$flipinsert$	185	2132	23.0486	0.4563	0.1244	5.4216	–	0	2
			$swapflip$	379	3416	18.0264	0.4235	0.0458	4.3140	–	0	2
			$swapflipinsert$	382	4208	22.0314	0.4484	0.0578	4.2749	–	0	2
			$swapinsert$	382	3005	15.7330	0.5823	0.0392	4.2749	–	0	19
		$fitness_3$	$flip$	173	802	9.2717	0.3259	0.0588	5.5954	–	0	2
			$insert$	630	5630	17.8730	0.4157	0.0301	2.0397	–	0	9
			$swap$	387	2334	12.0620	0.5106	0.0333	4.2765	–	0	19
			$flipinsert$	173	2054	23.7457	0.4820	0.1400	5.7977	–	0	2
			$swapflip$	387	3650	18.8630	0.4224	0.0459	4.2765	–	0	2
			$swapflipinsert$	390	4470	22.9231	0.4484	0.0600	4.2385	–	0	2
			$swapinsert$	390	3131	16.0564	0.5924	0.0409	4.2385	–	0	19
1 000	random	$fitness_1$	$flip$	972	1	0.0021	0.0000	0.0000	3.1173	–	0	971
			$insert$	835	0	0.0000	0.0000	0.0000	2.1449	–	0	835
			$swap$	972	1	0.0021	0.0000	0.0000	2.6173	–	0	971
			$flipinsert$	987	0	0.0000	0.0000	0.0000	2.5502	–	0	987
			$swapflip$	972	1	0.0021	0.0000	0.0000	2.6173	–	0	971
			$swapflipinsert$	987	3	0.0061	0.0000	0.0000	2.5380	–	0	984
			$swapinsert$	987	2	0.0041	0.0000	0.0000	2.5380	–	0	985
		$fitness_2$	$flip$	803	0	0.0000	0.0000	0.0000	2.9178	–	0	803
			$insert$	692	0	0.0000	0.0000	0.0000	2.2934	–	0	692
			$swap$	832	6	0.0144	0.0000	0.0000	2.9075	–	0	826
			$flipinsert$	832	0	0.0000	0.0000	0.0000	2.9014	–	0	832
			$swapflip$	832	8	0.0192	0.0000	0.0000	2.9123	–	0	824
			$swapflipinsert$	832	10	0.0240	0.0000	0.0000	2.9038	–	0	822
			$swapinsert$	832	8	0.0192	0.0000	0.0000	2.8990	–	0	824
		$fitness_3$	$flip$	803	0	0.0000	0.0000	0.0000	2.9178	–	0	803
			$insert$	704	0	0.0000	0.0000	0.0000	2.2884	–	0	704
			$swap$	844	6	0.0142	0.0000	0.0000	2.8945	–	0	838
			$flipinsert$	844	0	0.0000	0.0000	0.0000	2.8886	–	0	844
			$swapflip$	844	8	0.0190	0.0000	0.0000	2.8993	–	0	836
			$swapflipinsert$	844	12	0.0284	0.0000	0.0000	2.8910	–	0	832
			$swapinsert$	844	10	0.0237	0.0000	0.0000	2.8863	–	0	834
10 000	lex	$fitness_1$	$flip$	732	6286	17.1749	0.3105	0.0233	14.2514	2.9103	1	1
			$insert$	3574	71540	40.0336	0.3816	0.0113	3.3898	–	0	11
			$swap$	1785	32130	36.0000	0.4864	0.0199	7.3647	–	0	14
			$flipinsert$	938	37871	80.7484	0.4886	0.0855	11.9872	2.1088	1	1
			$swapflip$	1785	52112	58.3888	0.3814	0.0326	7.3647	3.1423	1	1
			$swapflipinsert$	2035	88221	86.7037	0.3988	0.0429	6.8590	3.0030	1	1
			$swapinsert$	2035	62664	61.5862	0.5420	0.0302	6.8590	–	0	14
		$fitness_2$	$flip$	1835	13696	14.9275	0.2121	0.0085	5.3232	–	0	2
			$insert$	5289	99516	37.6313	0.2836	0.0072	2.1624	–	0	8
			$swap$	3797	56380	29.6971	0.3364	0.0078	4.3303	–	0	14
			$flipinsert$	1812	48743	53.8002	0.2743	0.0299	5.5425	2.9435	1	1
			$swapflip$	3796	75460	39.7576	0.2920	0.0105	4.3314	4.2448	1	1
			$swapflipinsert$	3840	107445	55.9609	0.2797	0.0147	4.2745	3.8454	1	1
			$swapinsert$	3841	87823	45.7292	0.3417	0.0118	4.2734	–	0	14
		$fitness_3$	$flip$	1707	13253	15.5278	0.2230	0.0082	5.6473	–	0	2
			$insert$	5281	99007	37.4956	0.2841	0.0070	2.1888	–	0	9
			$swap$	3777	58117	30.7742	0.3377	0.0079	4.3818	–	0	14
			$flipinsert$	1683	47481	56.4242	0.2861	0.0336	5.9673	2.8311	1	1
			$swapflip$	3776	78943	41.8130	0.2926	0.0110	4.3829	4.1114	1	1
			$swapflipinsert$	3819	111198	58.2341	0.2819	0.0153	4.3263	3.7668	1	1
			$swapinsert$	3820	89878	47.0565	0.3470	0.0123	4.3251	–	0	14
10 000	random	$fitness_1$	$flip$	9698	11	0.0023	0.0000	0.0000	3.1448	–	0	9687
			$insert$	8335	29	0.0070	0.0000	0.0000	2.1445	–	0	8306
			$swap$	9698	93	0.0192	0.0003	0.0000	2.6172	–	0	9606
			$flipinsert$	9831	88	0.0179	0.0000	0.0000	2.5442	–	0	9743
			$swapflip$	9697	130	0.0268	0.0005	0.0000	2.6176	–	0	9569
			$swapflipinsert$	9828	194	0.0395	0.0002	0.0000	2.5308	–	0	9635
			$swapinsert$	9829	153	0.0311	0.0000	0.0000	2.5305	–	0	9676
		$fitness_2$	$flip$	8033	5	0.0012	0.0000	0.0000	2.9512	–	0	8028
			$insert$	6848	28	0.0082	0.0000	0.0000	2.3112	–	0	6820
			$swap$	8332	534	0.1282	0.0061	0.0000	2.9399	–	0	7881
			$flipinsert$	8353	43	0.0103	0.0000	0.0000	2.9251	–	0	8310
			$swapflip$	8326	744	0.1787	0.0084	0.0000	2.9455	–	0	7774
			$swapflipinsert$	8325	1108	0.2662	0.0115	0.0000	2.9348	–	0	7559
			$swapinsert$	8331	881	0.2115	0.0108	0.0000	2.9292	–	0	7648
		$fitness_3$	$flip$	8033	4	0.0010	0.0000	0.0000	2.9512	–	0	8029
			$insert$	6963	29	0.0083	0.0000	0.0000	2.3078	–	0	6934
			$swap$	8445	505	0.1196	0.0065	0.0000	2.9275	–	0	8010
			$flipinsert$	8468	40	0.0094	0.0000	0.0000	2.9127	–	0	8428
			$swapflip$	8438	690	0.1635	0.0082	0.0000	2.9333	–	0	7899
			$swapflipinsert$	8437	1058	0.2508	0.0102	0.0000	2.9230	–	0	7679
			$swapinsert$	8444	853	0.2020	0.0097	0.0000	2.9172	–	0	7769

Table .6: General LON and basins' statistics for $n = 4$ with 1 000 and 10 000 samples, considering lex and random initialization. A dash is shown when l cannot be computed as multiple disconnected components exist.

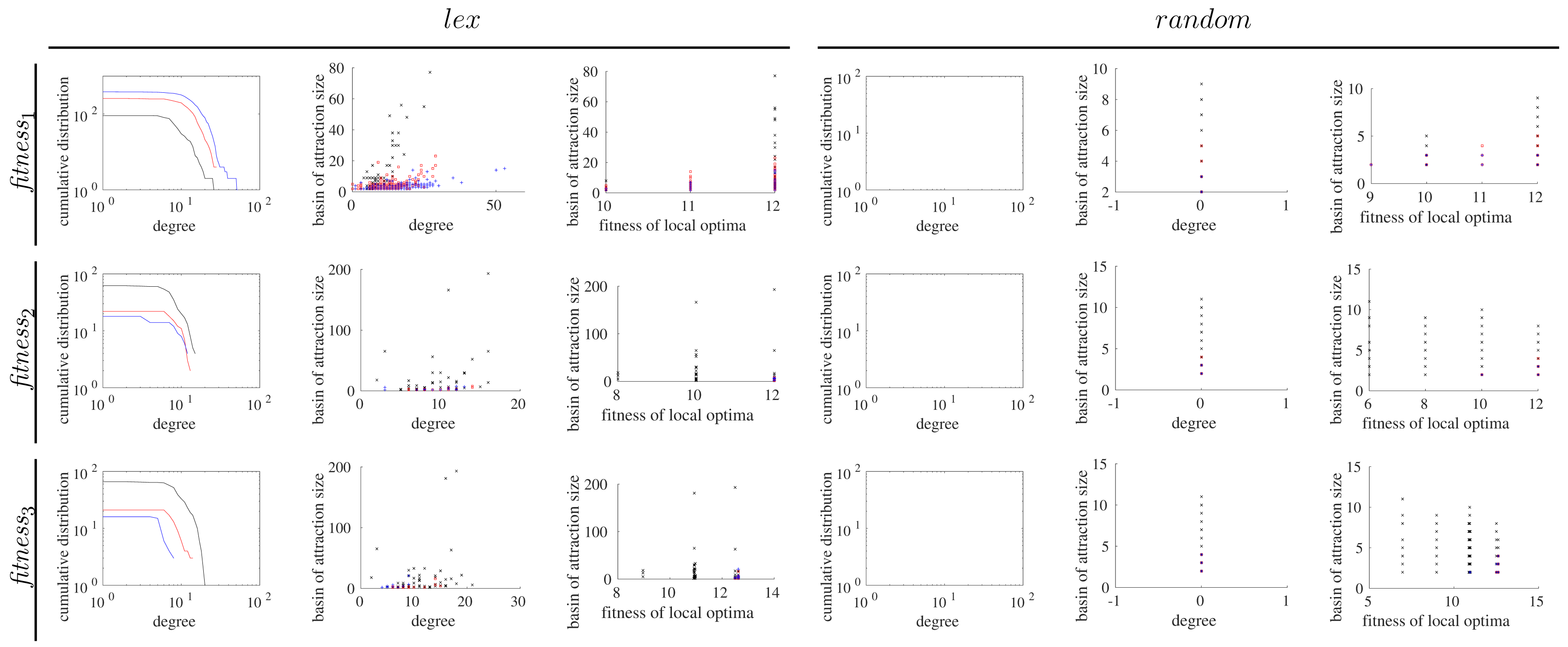

Figure .13: Statistical measures for *lex* (left) and *random* (right) initialization on $n = 5$ with the three fitness functions for operators flip (black), insert (blue), swap (red) using 1 000 samples (within each block of three): Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and corresponding basin sizes (right). The forth column is blank, as nodes are of degree 0 ($z = 0$) (see fifth column).

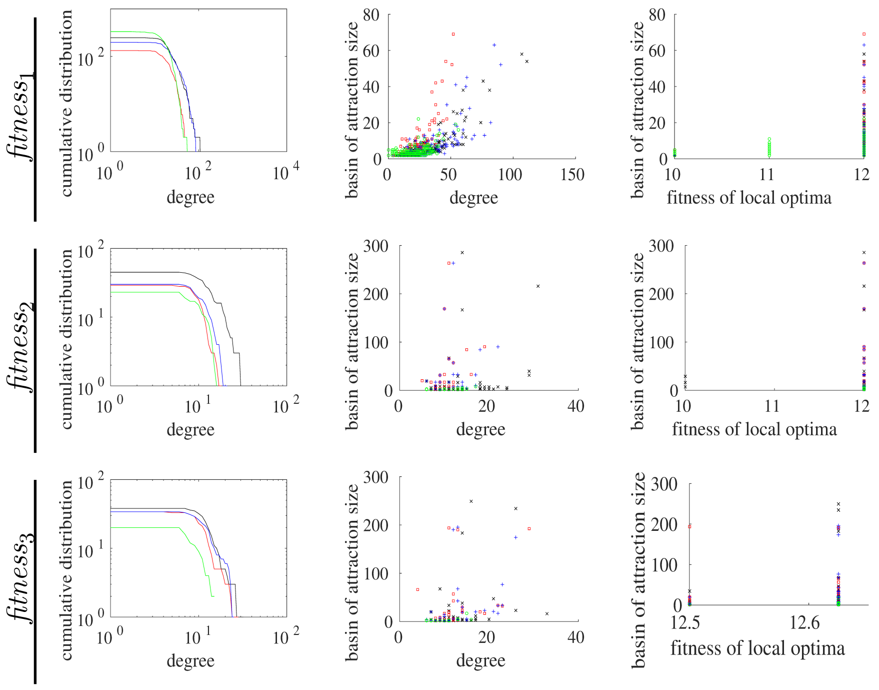

Figure .14: Statistical measures for *lex* initialization on $n = 5$ with the three fitness functions for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green) using 1 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and corresponding basin sizes (right).

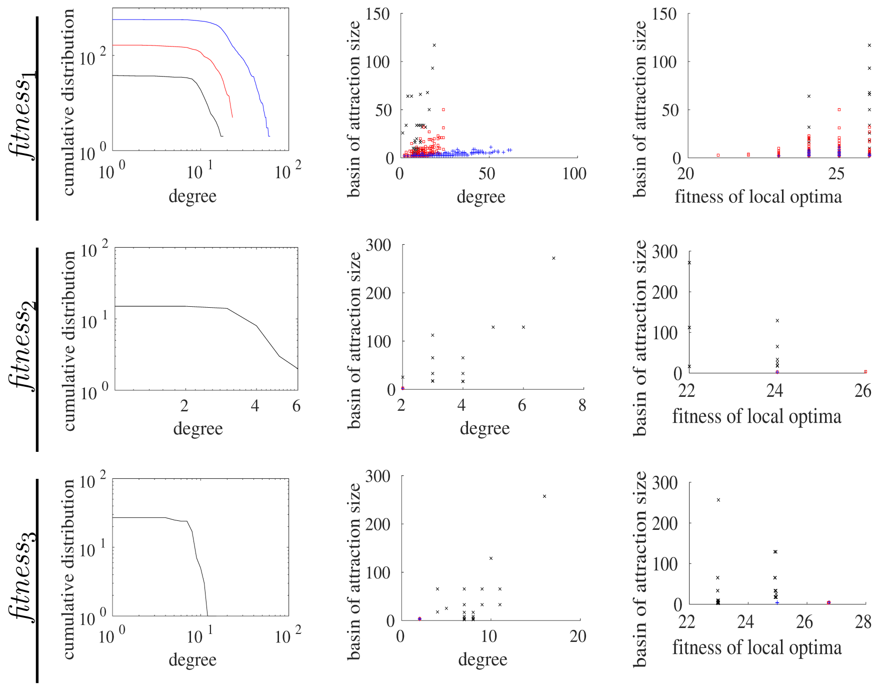

Figure .15: Statistical measures for *lex* initialization on $n = 6$ with the three fitness functions for operators flip (black), insert (blue), swap (red) using 1 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and corresponding basin sizes (right).

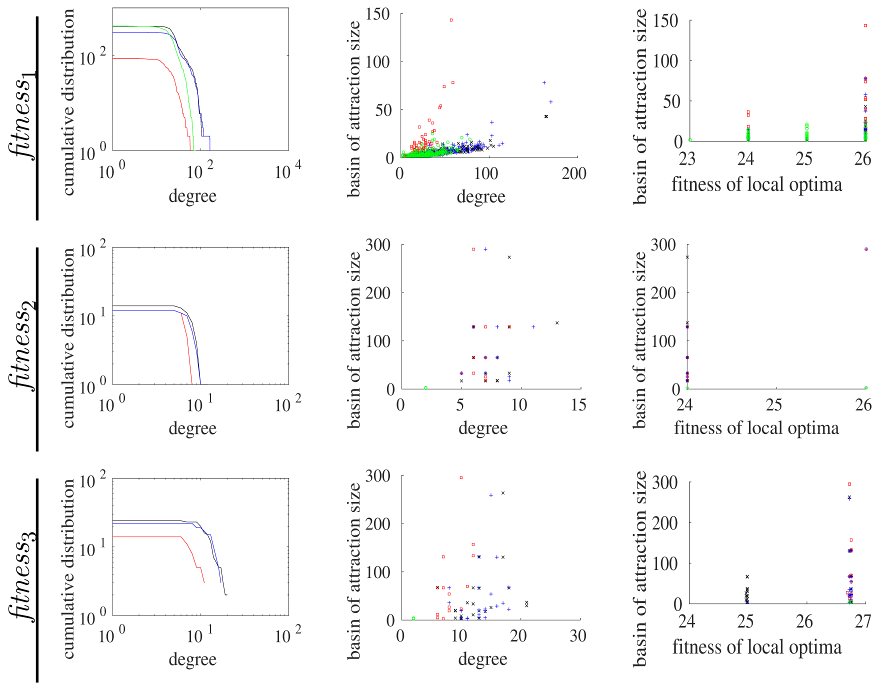

Figure .16: Statistical measures for *lex* initialization on $n = 6$ with fitness functions $fitness_2$ and $fitness_3$ for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green) using 1 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and corresponding basin sizes (right).

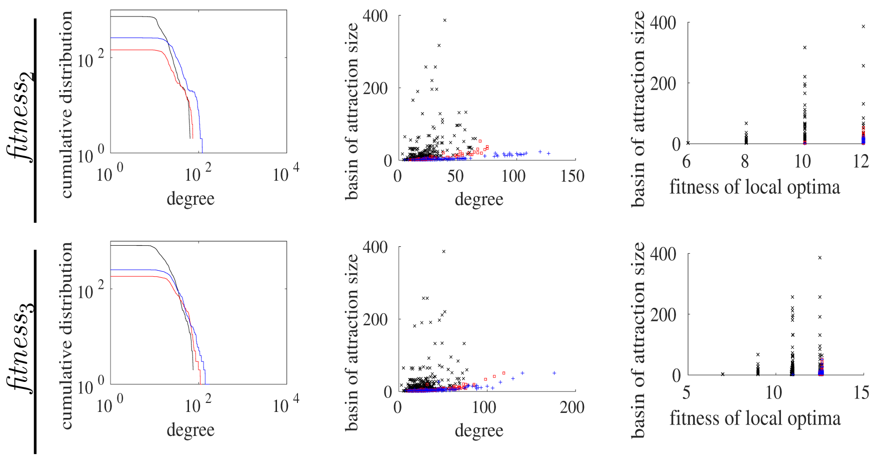

Figure .17: Statistical measures for *lex* initialization on $n = 5$ with fitness functions $fitness_2$ and $fitness_3$ for operators flip (black), insert (blue), swap (red) using 10 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and corresponding basin sizes (right).

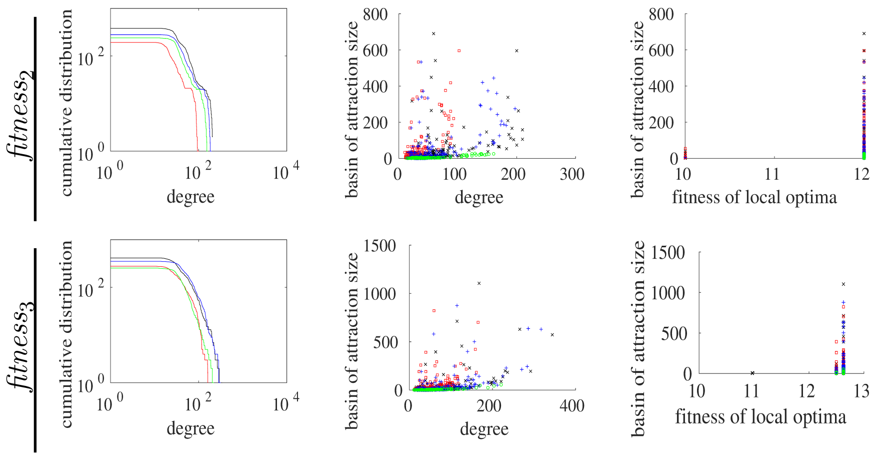

Figure .18: Statistical measures for *lex* initialization on $n = 5$ with fitness functions $fitness_2$ and $fitness_3$ for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green) using 10 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and corresponding basin sizes (right).

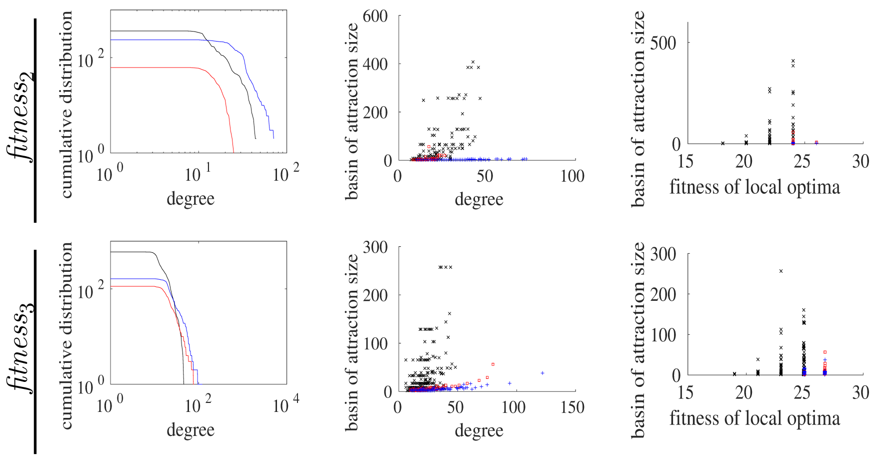

Figure .19: Statistical measures for *lex* initialization on $n = 6$ with fitness functions $fitness_2$ and $fitness_3$ for operators flip (black), insert (blue), swap (red) using 10 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and corresponding basin sizes (right).

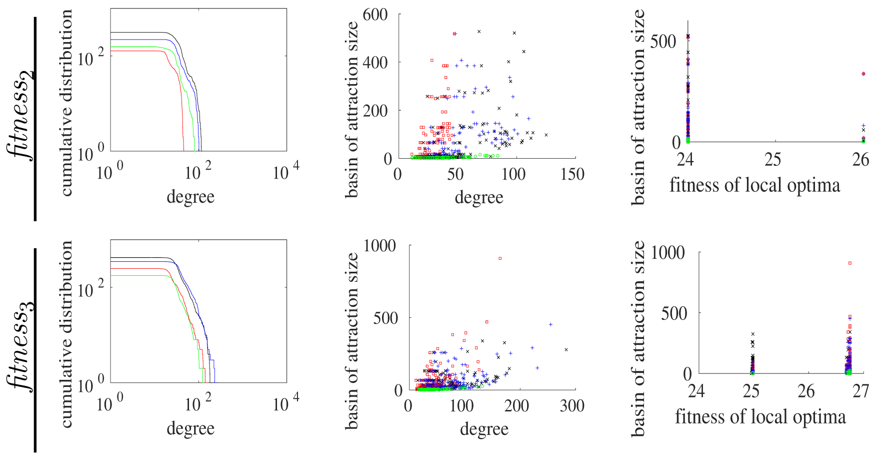

Figure .20: Statistical measures for *lex* initialization on $n = 6$ with fitness functions $fitness_2$ and $fitness_3$ for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green) using 10 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and corresponding basin sizes (right).