

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](https://doi.org/10.1016/j.asoc.2021.107327)

Publication date

2021

Document Version

Accepted author manuscript

Published in

Applied Soft Computing

Citation (APA)

Jakobovic, D., Picek, S., Martins, M. S. R., & Wagner, M. (2021). Toward more efficient heuristic construction of Boolean functions. *Applied Soft Computing*, 107, Article 107327. <https://doi.org/10.1016/j.asoc.2021.107327>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

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 in 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 $(f(0), \dots, f(1))$ 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]:

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

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.

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

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

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

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

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

$$Nl_f = 2^{n-1} - 2^{\frac{n}{2}-1}. \quad (3)$$

87 The nonlinearity of a Boolean function with n variables is bounded above
88 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

114 and hill climbing. With that approach, the authors were able to find some
 115 functions of eight and nine inputs that have a combination of nonlinearity
 116 and autocorrelation values previously not obtained. Millan et al. [23] pro-
 117 posed a new adaptive strategy for the local search algorithm for the gener-
 118 ation of Boolean functions with high nonlinearity. Hernan et al. [24] were
 119 the first to use a multi-objective random bit climber to search for balanced
 120 Boolean functions of size up to eight inputs with high nonlinearity. Picek et
 121 al. [25] experimented with genetic algorithms and genetic programming to
 122 find Boolean functions that possess several cryptographic properties. As far
 123 as we are aware, this is the first application of genetic programming to the
 124 evolution of Boolean functions with cryptographic properties. Mariot and
 125 Leporati [26] used Particle Swarm Optimization to find Boolean functions
 126 with good trade-offs of cryptographic properties for dimensions up to 12.

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

133 4. Analyzing Fitness Landscapes

134 Fitness landscapes describe the relationship between search and fitness
 135 space [28], thus a heuristic strategy can navigate a specific landscape struc-
 136 ture searching for optimal solutions.

137 Several cost models have been used to make specific predictions for com-
 138 binatorial problems, identifying which features of the fitness landscape con-
 139 tribute more to the problem solving complexity during the search. By iden-
 140 tifying these features, some improvements regarding the algorithm perfor-
 141 mance can be designed.

142 The Local Optima Network (LON) [29] is a model designed to under-
 143 stand the local optima structure in combinatorial landscapes, incorporating
 144 network analysis techniques to study fitness landscapes and problem diffi-
 145 culty [30].

146 The fitness landscape in LON models is modeled as a graph where the
 147 local optima represent nodes that can be connected. A local search heuristic
 148 $\mathcal{H}$ maps the solution space S to the set of locally optimal solutions S^* . Given

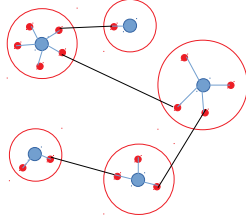


Figure 1: An example of the connectivity in local optima networks.

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

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

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

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

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

176 Recently, some works addressed a LON variant called Compressed Lo-
 177 cal Optima Network. The work proposed in [37] investigated fitness land-

178 scape properties for the Number Partitioning Problem, exploring whether the
 179 global landscape structure of the number partitioning problem changes with
 180 the phase transition. In [38], the authors analyzed the network features to
 181 find differences between the landscape structures for the Permutation Flow-
 182 shop Scheduling Problem (PFSP). The results provided insights into which
 183 features impact the performance of an iterated local search heuristic.

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

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

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

201 5. Creating Networks with Local Search

202 In order to obtain LONs of the Boolean function optimization landscape,
 203 we use a local search procedure that, starting from a given initial solution,
 204 converges to a corresponding local optimum. Along with the initial solution,
 205 all the intermediate solutions leading from the initial solution to the local op-
 206 timum are added as the members of that local optimum’s basin of attraction.
 207 This procedure is repeated for each solution in the set of initial solutions. Af-
 208 ter that, we record all unique local optima and reconstruct the connections
 209 between their basins of attraction.

210 The local search is described in Algorithm 1; it can be used with an
 211 arbitrary representation and an arbitrary neighborhood relationship, where
 212 $\mathcal{N}(\cdot)$ represents the neighborhood of the given solution. In the local search,
 213 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
```

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

220 5.1. Neighborhood Operators

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

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

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

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

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

257 As we always consider the entire neighborhood before selecting the best,
258 the order of the neighborhood-operators in these combinations does not mat-
259 ter, unless – as previously highlighted – the fitness of several solutions is
260 identical. In that case, the algorithm will keep the first solution with the
261 best fitness value it encounters, which favors first neighborhoods in the com-
262 bination.

263 5.2. Initialization Strategies

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

271 5.3. Fitness Functions for Optimization of Boolean Functions

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

$$fitness_1 : Nl_f. \quad (4)$$

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

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

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

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

285 The *Indicator* property is the normalized number of occurrences of the max-
 286 imal nonlinearity value in the whole spectrum (denoted $\#max_values$). Nat-
 287 urally, the smaller the number of such maximal values, the easier it is for the
 288 algorithm to reach the next nonlinearity value: $Indicator = 2^n - \frac{\#max_values}{2^n}$.

289 6. Results and Discussion

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

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

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

302 As it is possible to perform an exhaustive search for problem size $n =$
 303 4 – because the total number of solutions is 2^{16} – we build the LONs by
 304 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

¹While 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*

²We 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.

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

Function	Operator	n_v	n_e	z	C	C_r	b	l	π	S
<i>fitness₁</i>	<i>flip</i>	12774	275388	43.1170	0.2672	0.0034	3.4656	—	0	9
	<i>insert</i>	12904	435761	67.5389	0.2771	0.0052	3.5939	—	0	7
	<i>swap</i>	12774	275388	43.1170	0.2672	0.0034	3.4656	—	0	9
	<i>flipinsert</i>	1182	150095	253.9679	0.4945	0.2151	54.7394	1.82	1	1
	<i>swapflip</i>	1176	103700	176.3605	0.4093	0.1500	55.0136	1.92	1	1
	<i>swapflipinsert</i>	1176	174167	296.2024	0.5072	0.2521	55.0136	1.77	1	1
	<i>swapinsert</i>	9806	533767	108.8654	0.2794	0.0111	4.5030	—	0	7
<i>fitness₂</i>	<i>flip</i>	1776	14249	16.0462	0.3082	0.0089	2.0878	—	0	3
	<i>insert</i>	1712	20581	24.0432	0.2956	0.0135	2.1379	3.73	1	1
	<i>swap</i>	1776	14249	16.0462	0.3082	0.0092	2.0878	—	0	3
	<i>flipinsert</i>	10922	471252	86.2941	0.1973	0.0079	6.0004	3.23	1	1
	<i>swapflip</i>	10920	401688	73.5692	0.2265	0.0067	6.0015	3.43	1	1
	<i>swapflipinsert</i>	10920	648497	118.7723	0.1955	0.0109	6.0015	2.94	1	1
	<i>swapinsert</i>	1484	29029	39.1226	0.2638	0.0260	2.3140	2.81	1	1
<i>fitness₃</i>	<i>flip</i>	2292	24305	21.2086	0.2648	0.0089	2.2094	3.79	1	1
	<i>insert</i>	2166	30553	28.2114	0.2604	0.0129	2.2872	3.53	1	1
	<i>swap</i>	2292	24305	21.2086	0.2648	0.0093	2.2094	3.79	1	1
	<i>flipinsert</i>	10082	472850	93.8008	0.2053	0.0093	6.5003	3.08	1	1
	<i>swapflip</i>	10080	398788	79.1246	0.2296	0.0079	6.5016	3.24	1	1
	<i>swapflipinsert</i>	10080	642179	127.4165	0.2012	0.0127	6.5016	2.83	1	1
	<i>swapinsert</i>	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_+	b	l	π	S
5	lex	$fitness_2$	<i>flip</i>	730	6272	17.1836	0.3022	0.0221	13.6356	3.0883	1	1
			<i>insert</i>	260	4377	33.6692	0.4909	0.1296	3.8577	2.0222	1	1
			<i>swap</i>	145	1960	27.0345	0.5687	0.1888	6.1172	1.9519	1	1
			<i>flipinsert</i>	382	9569	50.0995	0.5323	0.1324	27.0969	1.9449	1	1
			<i>swapflip</i>	193	2948	30.5492	0.5772	0.1610	52.6528	1.9664	1	1
			<i>swapflipinsert</i>	280	7000	50.0000	0.5756	0.1784	36.6464	1.8604	1	1
		$fitness_3$	<i>swapinsert</i>	240	5373	44.7750	0.5505	0.1862	4.1458	1.8633	1	1
			<i>flip</i>	813	8055	19.8155	0.3190	0.0238	12.3456	2.9684	1	1
			<i>insert</i>	251	4355	34.7012	0.4918	0.1413	4.4542	1.9452	1	1
			<i>swap</i>	183	3238	35.3880	0.5245	0.1972	5.3497	1.8835	1	1
			<i>flipinsert</i>	414	11849	57.2415	0.5437	0.1395	25.8527	1.8909	1	1
			<i>swapflip</i>	277	6554	47.3213	0.5370	0.1714	37.6570	1.8891	1	1
6	lex	$fitness_2$	<i>swapflipinsert</i>	352	12190	69.2614	0.5584	0.1978	30.1477	1.8096	1	1
			<i>swapinsert</i>	255	6363	49.9059	0.5292	0.1951	4.3451	1.8267	1	1
			<i>flip</i>	363	3025	16.6667	0.3863	0.0450	27.5840	3	1	1
			<i>insert</i>	236	3536	29.9661	0.4187	0.1277	2.3136	2.1369	1	1
			<i>swap</i>	62	487	15.7097	0.4982	0.2505	5.9677	1.9334	1	1
			<i>flipinsert</i>	314	6550	41.7197	0.4724	0.1327	32.7229	2.0490	1	1
		$fitness_3$	<i>swapflip</i>	128	1698	26.5313	0.5045	0.2089	78.8047	1.9382	1	1
			<i>swapflipinsert</i>	221	4404	39.8552	0.5305	0.1815	46.0905	1.9461	1	1
			<i>swapinsert</i>	156	2608	33.4359	0.5128	0.2163	3.0128	1.8864	1	1
			<i>flip</i>	595	4822	16.2084	0.3481	0.0280	17.2185	3.1043	1	1
			<i>insert</i>	163	2283	28.0123	0.5252	0.1816	4.4479	1.9076	1	1
7	lex	$fitness_2$	<i>swap</i>	113	1396	24.7080	0.5608	0.2219	5.9735	1.8254	1	1
			<i>flipinsert</i>	422	9640	45.6872	0.4684	0.1076	26.3436	2.0016	1	1
			<i>swapflip</i>	249	4420	35.5020	0.4949	0.1437	43.8153	1.9196	1	1
			<i>swapflipinsert</i>	348	9459	54.3621	0.5046	0.1572	32.0920	1.8754	1	1
			<i>swapinsert</i>	177	3389	38.2938	0.5402	0.2185	4.7345	1.8073	1	1
		$fitness_3$	<i>flip</i>	512	4223	16.4961	0.3550	0.0329	19.3516	—	0	2
			<i>insert</i>	513	7742	30.1832	0.3696	0.0592	2.3294	—	0	2
			<i>swap</i>	30	154	10.2667	0.5846	0.3326	21.9333	2	1	1
			<i>flip</i>	639	5212	16.3130	0.3201	0.0250	15.7042	—	0	2
			<i>insert</i>	465	7152	30.7613	0.3926	0.0659	6.2839	3	1	1
			<i>swap</i>	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, 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.

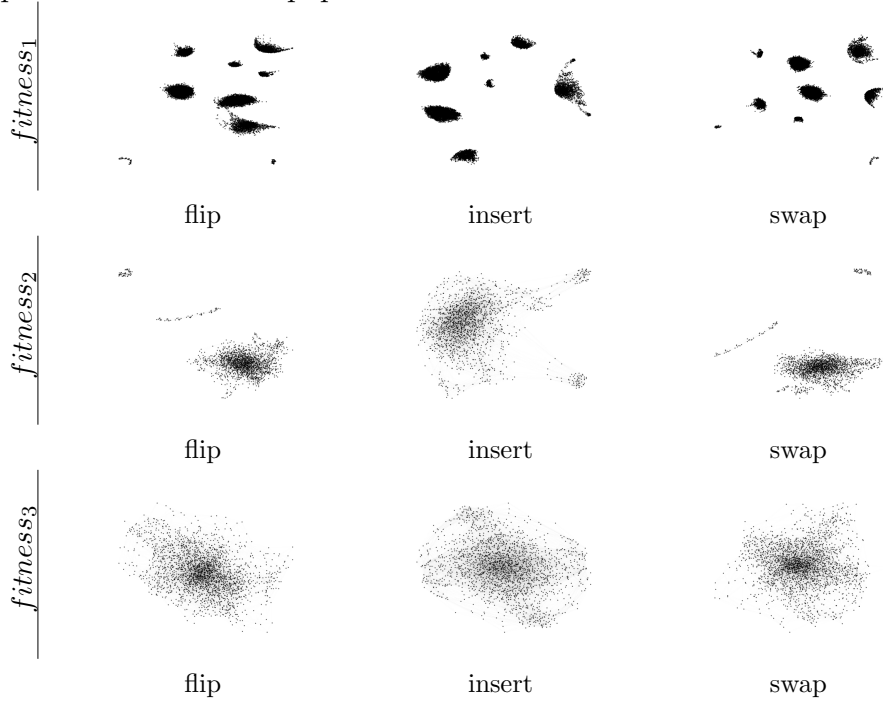


Figure 2: LON graphs on exhaustive $n = 4$ with the three fitness functions for operators flip, insert, swap.

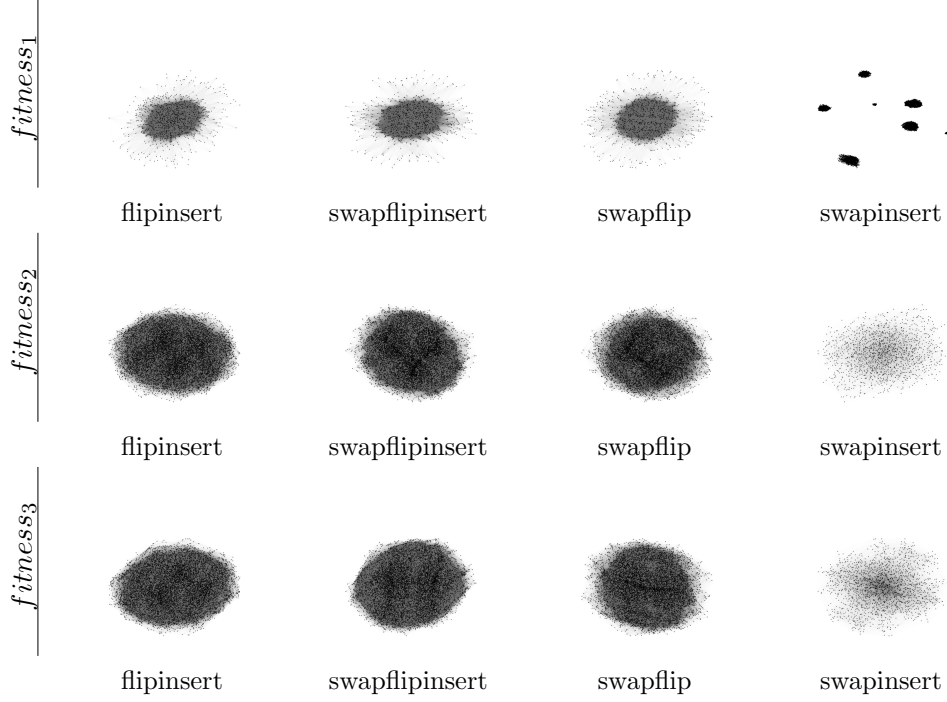


Figure 3: LON graphs on exhaustive $n = 4$ with the three fitness functions for combination using operators flipinsert, swapflipinsert, swapflip, and swapinsert.

388 6.2. Distribution of Degree

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

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

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

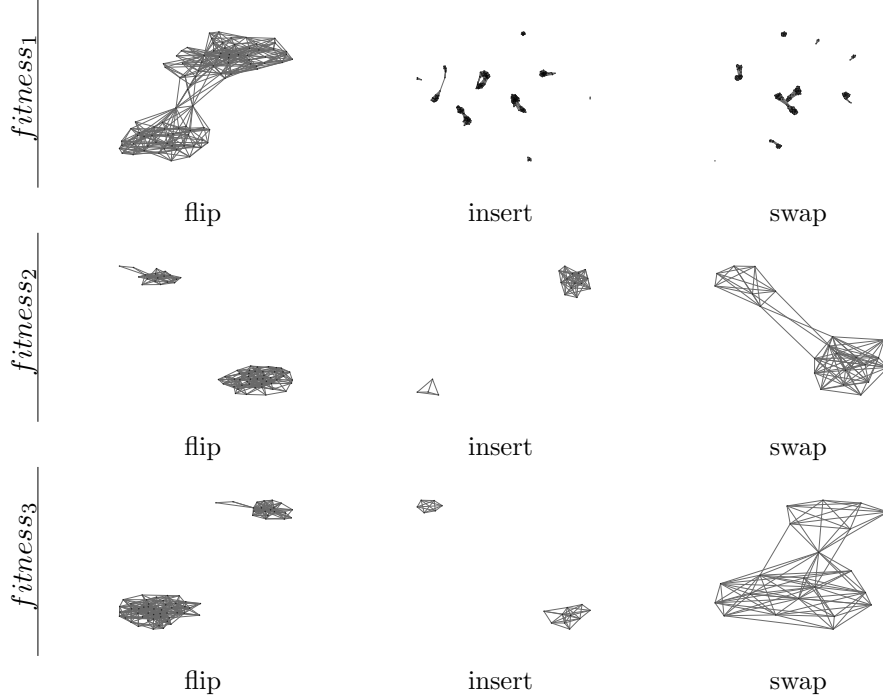


Figure 4: LON graphs for *lex* initialization on $n = 5$ with fitness functions $fitness_1$, $fitness_2$ and $fitness_3$ using 1 000 samples for all three operators: flip, insert, swap.

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

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

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

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

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

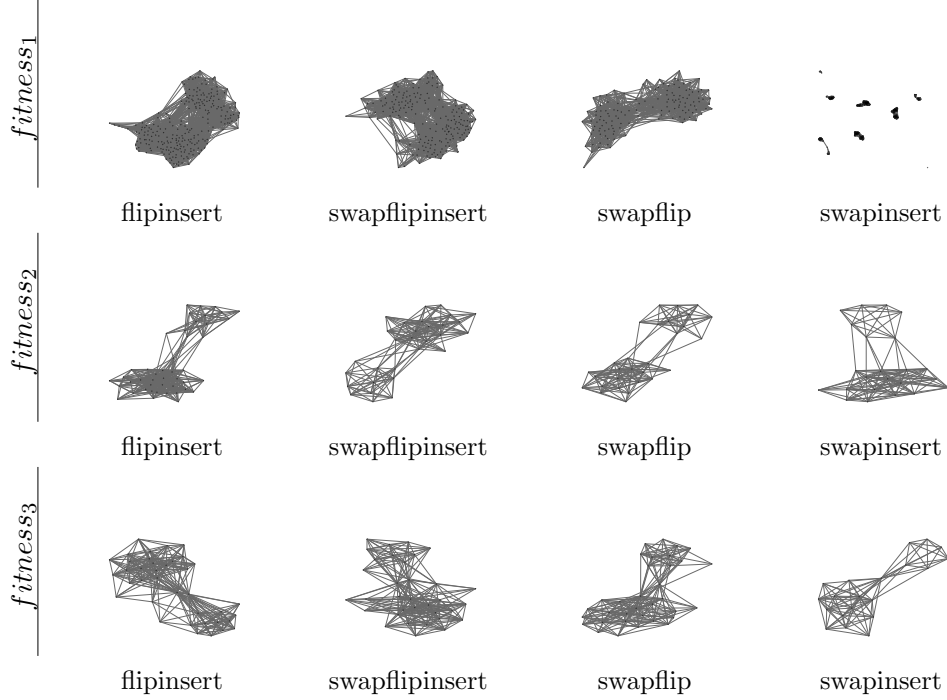


Figure 5: LON graphs for *lex* initialization on $n = 5$ with fitness functions $fitness_1$, $fitness_2$ and $fitness_3$ using 1000 samples for combination using operators flipinsert, swapflipinsert, swapflip, and swapinsert.

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

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

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

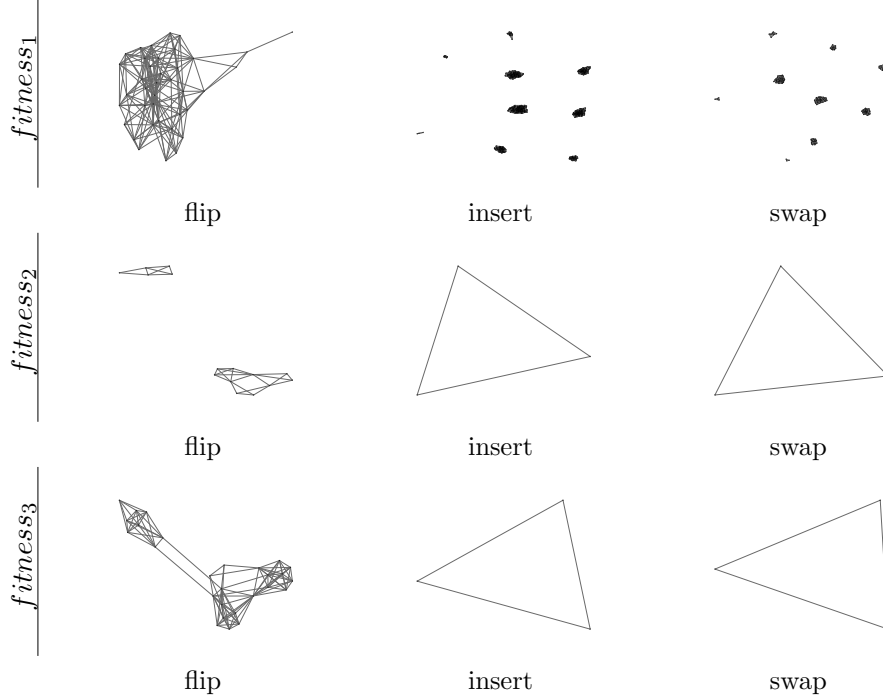


Figure 6: LON graphs for *lex* initialization on $n = 6$ with fitness functions $fitness_1$, $fitness_2$ and $fitness_3$ using 1 000 samples for all three operators: flip, insert, swap.

437 explore next.

438 6.3. Basin Size Correlation

439 Our “matrices of plots” present the basin correlation in the middle and
 440 right columns, i.e., Figure 8 and Figure 9 show this for $n = 4$, and Fig-
 441 ures 10, 11, and 12 show these for the larger values of n . A particular focus
 442 of the last three mentioned figures is the difference of 1 000 samples to 10 000
 443 samples.

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

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

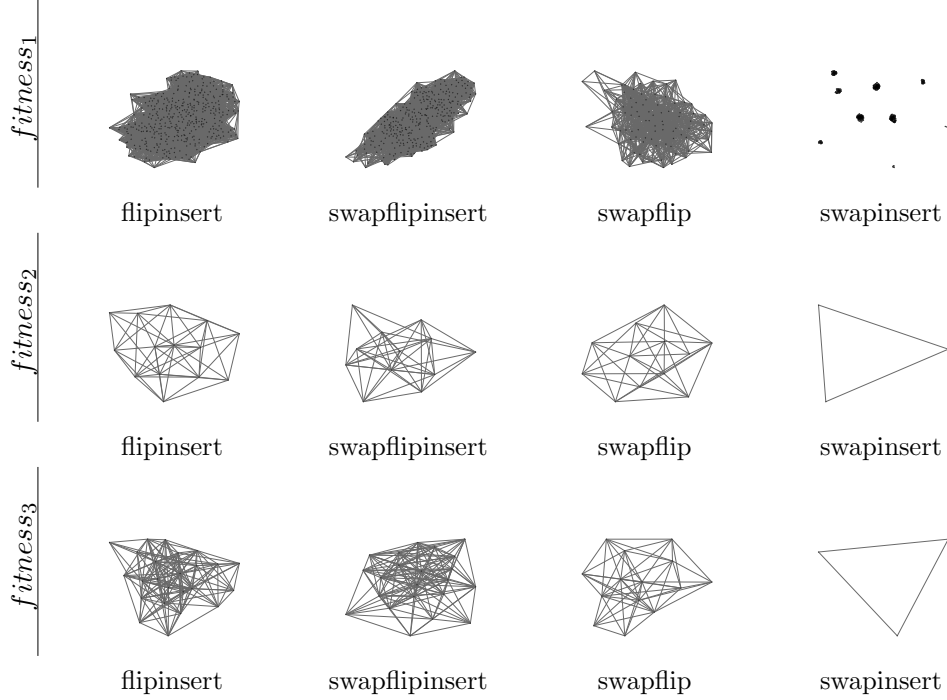


Figure 7: LON graphs for *lex* initialization on $n = 6$ with fitness functions $fitness_1$, $fitness_2$ and $fitness_3$ using 1000 samples for combination using operators flipinsert, swapflipinsert, swapflip, and swapinsert.

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

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

471 however, the correlation is a lot weaker and hence might be difficult to exploit.

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

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

493 These experimental results can summarize some insights regarding the
494 search improvements. The topological properties for *fitness*₂ and *fitness*₃
495 are LON of only one component for almost all instances, presenting high
496 mean degree and small minimum distances between nodes. Besides, the lo-
497 cal optima are connected as dense local clusters and sparse interconnections,
498 which can be difficult to exploit. Some heuristics such as Tabu Search with
499 restarts or a Memetic Algorithm with built-in local searches, or even an
500 approach with explicit niching might be able to explore the entire network
501 searching for promising solutions. According to the basin distribution there
502 are few nodes with a large number of neighbors connecting the entire land-
503 scape: a search at a random node has more chances to move to one of these
504 high degree nodes, and then to another node, which can be an efficient way
505 to search the entire network. Moreover flip operator seems to provide more
506 information using basin size and fitness of local optima correlation than the
507 others operators.

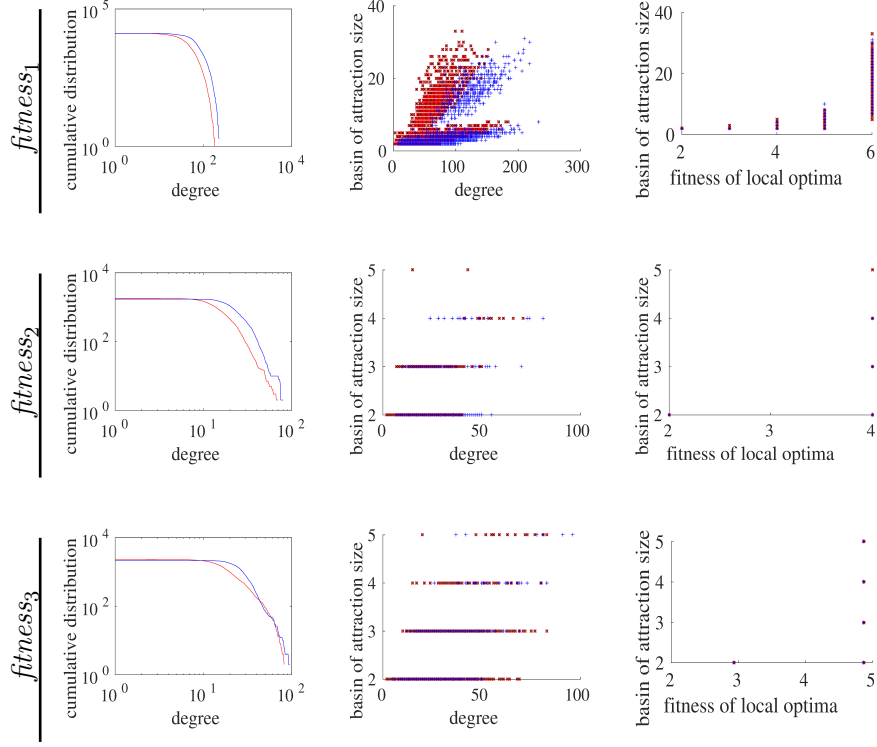


Figure 8: Statistical measures on exhaustive $n = 4$ with the three fitness functions for operators flip (black), insert (blue), swap (red): Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation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

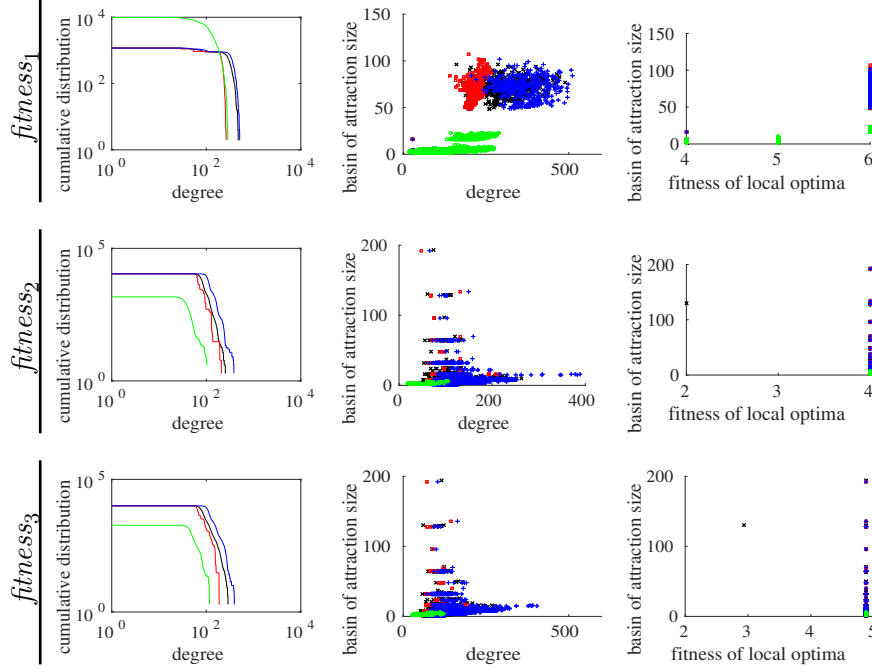


Figure 9: Statistical measures on exhaustive $n = 4$ with the three fitness functions for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green): Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima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

519 using a local search heuristic to investigate the search space structure. Fur-
520 thermore, we studied the degree distribution and the correlation between the
521 basins of attraction with some LON properties looking for additional infor-
522 mation about the search difficulty, considering scenarios (e.g., combinations
523 of neighborhoods) not investigated before.

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

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

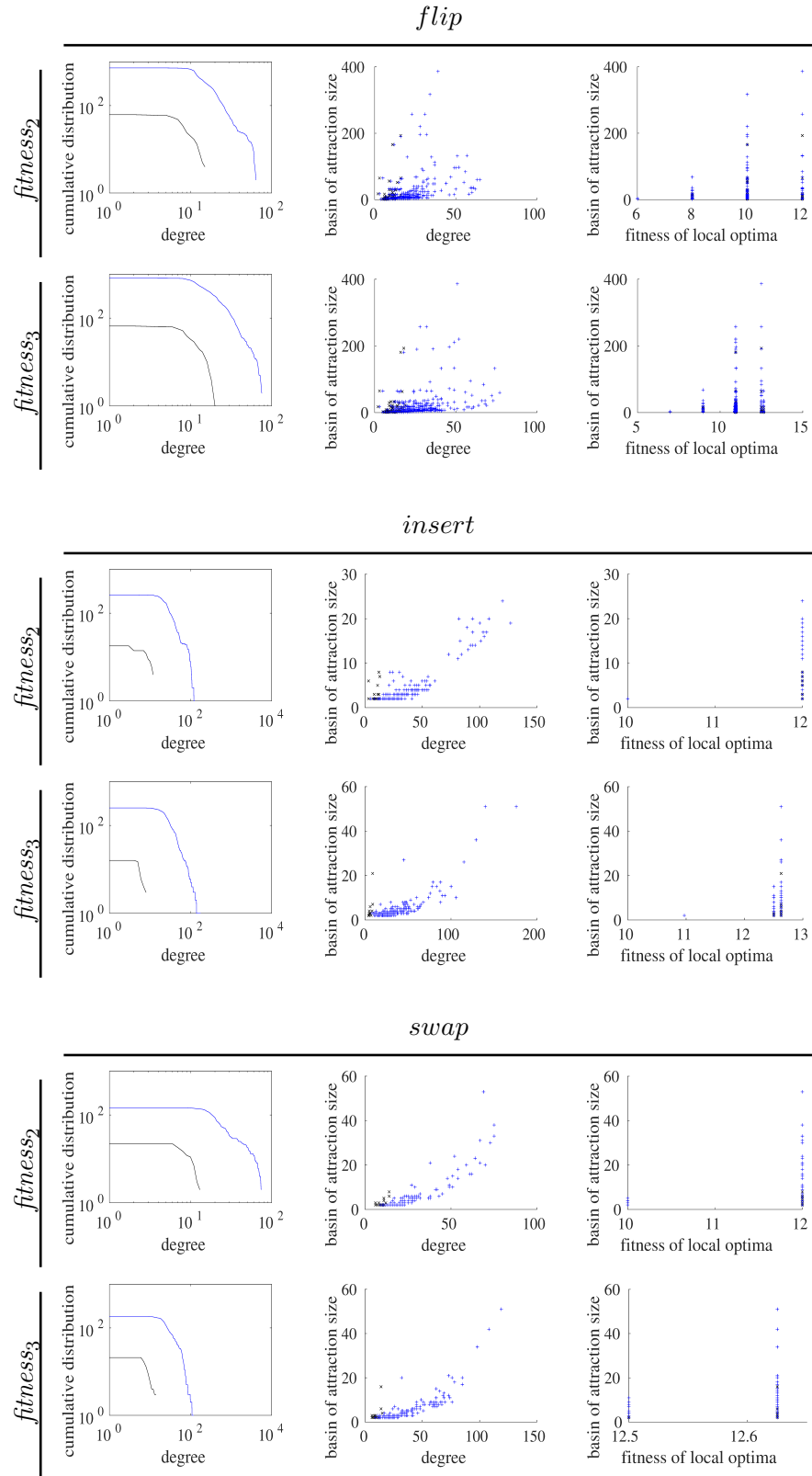


Figure 10: Statistical measures for *lex* initialization on $n = 5$ with fitness functions *fitness₂* and *fitness₃* using 1 000 (black) and 10 000 (blue) samples for all three operators: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding basin sizes (middle), and Correlation between the fitness of local optima and their corresponding basin sizes (right).

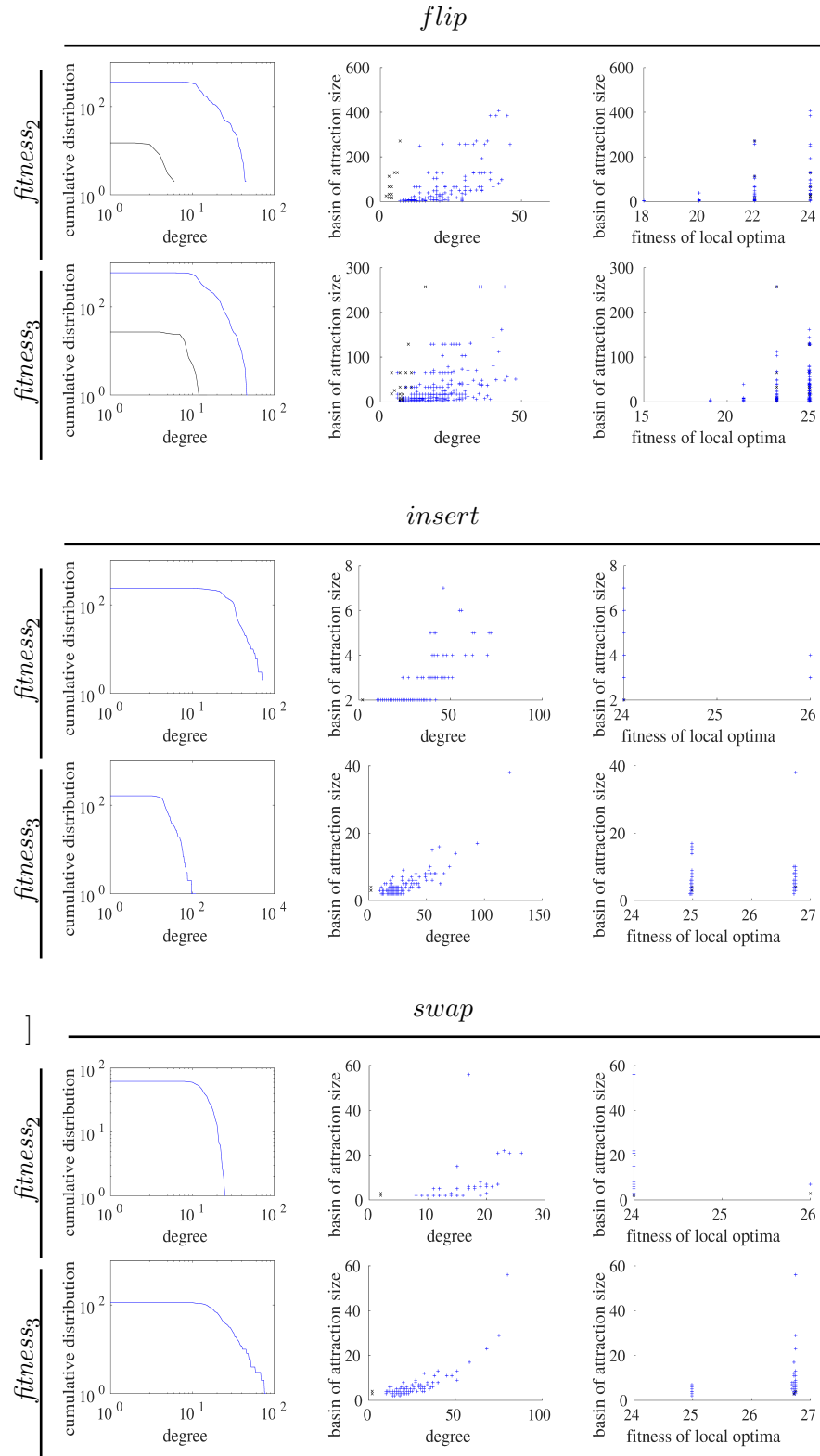


Figure 11: Statistical measures for *lex* initialization on $n = 6$ with fitness functions *fitness₂* and *fitness₃* using 1000 (black) and 10000 (blue) samples for all three operators: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their 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.

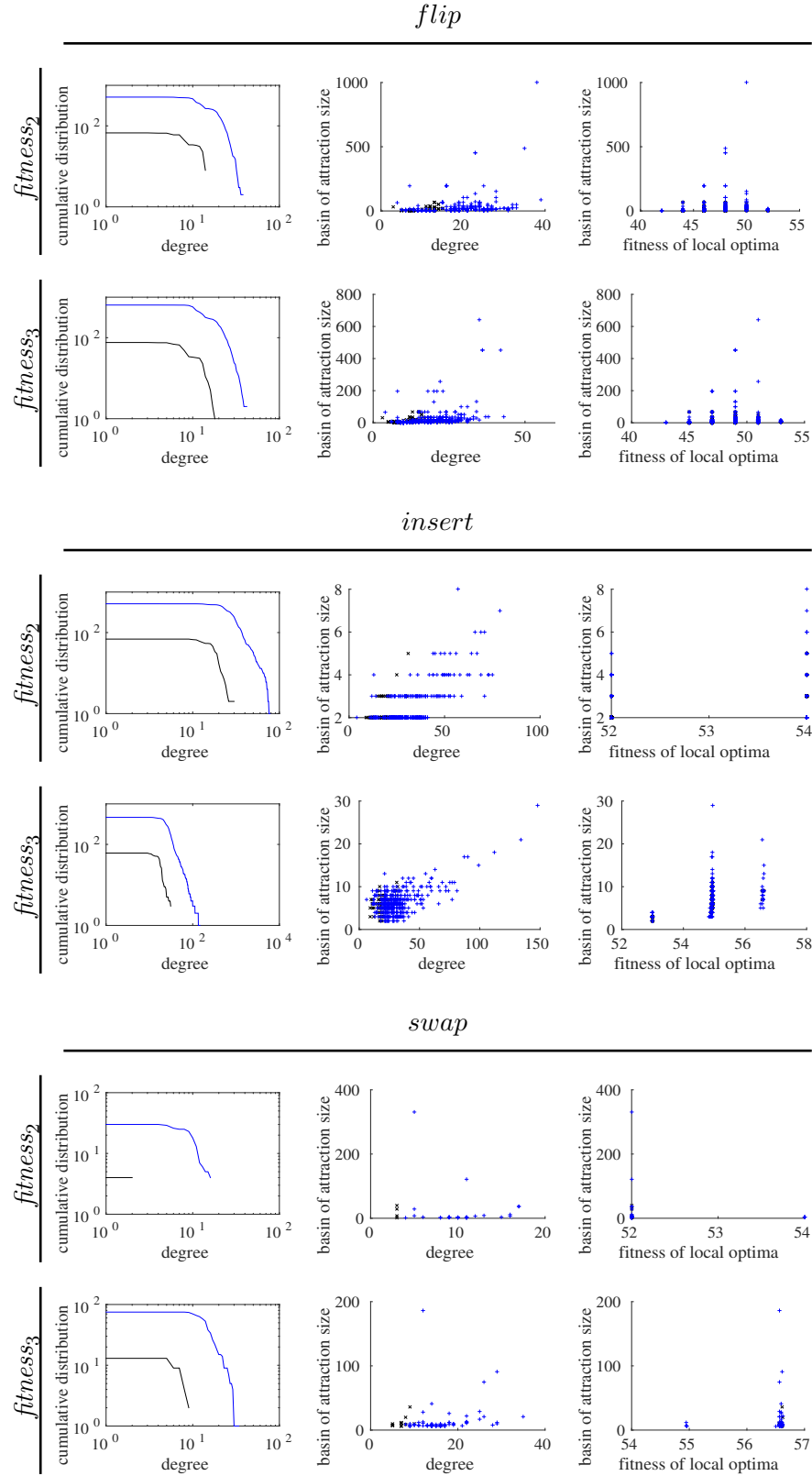


Figure 12: Statistical measures for *lex* initialization on $n = 7$ with fitness functions $fitness_2$ and $fitness_3$ using 1 000 (black) and 10 000 (blue) samples for all three operators: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their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

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

534 Acknowledgments

535 Our work was supported by the Australian Research Council projects
536 DE160100850, DP200102364, and DP210102670. Parts of our work have been
537 inspired by COST Action CA15140 supported by COST (European Cooper-
538 ation in Science and Technology).

539 References

- 540 [1] O. Rothaus, On “bent” functions, *Journal of Combinatorial Theory,*
541 *Series A* 20 (3) (1976) 300 – 305.
- 542 [2] A. Bernasconi, B. Codenotti, J. M. Vanderkam, A characterization of
543 bent functions in terms of strongly regular graphs, *IEEE Transactions*
544 *on Computers* 50 (9) (2001) 984–985.
- 545 [3] X. Huang, K. Fang, L. Fang, Q. Chen, Z.-R. Lai, L. Wei, Bi-
546 kronecker functional decision diagrams: A novel canonical representa-
547 tion of boolean functions, in: *AAAI Conference on Artificial Intelligence,*
548 *Vol. 33,* 2019, pp. 2867–2875.
- 549 [4] S. Kavut, S. Maitra, M. D. Yucel, Search for boolean functions with
550 excellent profiles in the rotation symmetric class, *IEEE Transactions on*
551 *Information Theory* 53 (5) (2007) 1743–1751.
- 552 [5] A. Kerdock, A class of low-rate nonlinear binary codes, *Information and*
553 *Control* 20 (2) (1972) 182 – 187.
- 554 [6] J. Olsen, R. Scholtz, L. Welch, Bent-function sequences, *IEEE Trans.*
555 *on Information Theory* 28 (6) (1982) 858–864.
- 556 [7] K. Paterson, On Codes With Low Peak-to-Average Power Ratio for
557 Multicode CDMA, *IEEE Transactions on Information Theory* 50 (2004)
558 550 – 559.
- 559 [8] M. Hell, T. Johansson, A. Maximov, W. Meier, A stream cipher pro-
560 posal: Grain-128, in: *IEEE Int. Symposium on Information Theory,* 2006,
561 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.

- 594 [18] J. Dillon, A Survey of Bent Functions*, Tech. rep., Reprinted from the
595 NSA Technical Journal. Special Issue, unclassified (1972).
- 596 [19] W. Millan, A. Clark, E. Dawson, An Effective Genetic Algorithm for
597 Finding Highly Nonlinear Boolean Functions, in: First Int. Conference
598 on Information and Communication Security, ICICS '97, Springer, 1997,
599 pp. 149–158.
- 600 [20] W. Millan, A. Clark, E. Dawson, Heuristic design of cryptographically
601 strong balanced Boolean functions, in: Advances in Cryptology - EU-
602 ROCRYPT '98, 1998, pp. 489–499.
- 603 [21] J. A. Clark, J. L. Jacob, Two-Stage Optimisation in the Design of
604 Boolean Functions, in: Information Security and Privacy, Vol. 1841 of
605 LNCS, Springer, 2000, pp. 242–254.
- 606 [22] S. Kavut, M. D. Yücel, Improved Cost Function in the Design of Boolean
607 Functions Satisfying Multiple Criteria, in: Progress in Cryptology - IN-
608 DOCRYPT 2003, Vol. 2904 of LNCS, Springer, 2003, pp. 121–134.
- 609 [23] W. Millan, J. Fuller, E. Dawson, New concepts in evolutionary search
610 for Boolean functions in cryptology, Computational Intelligence 20 (3)
611 (2004) 463–474.
- 612 [24] H. Aguirre, H. Okazaki, Y. Fuwa, An Evolutionary Multiobjective Ap-
613 proach to Design Highly Non-linear Boolean Functions, in: Genetic and
614 Evolutionary Computation Conference (GECCO), 2007, pp. 749–756.
- 615 [25] S. Picek, D. Jakobovic, M. Golub, Evolving Cryptographically Sound
616 Boolean Functions, in: Genetic and Evolutionary Computation Confer-
617 ence (GECCO), GECCO '13 Companion, ACM, 2013, pp. 191–192.
- 618 [26] L. Mariot, A. Leporati, Heuristic Search by Particle Swarm Optimization
619 of Boolean Functions for Cryptographic Applications, in: Genetic and
620 Evolutionary Computation Conference, GECCO, Companion Material
621 Proceedings, 2015, pp. 1425–1426.
- 622 [27] R. Hrbacek, V. Dvorak, Bent Function Synthesis by Means of Cartesian
623 Genetic Programming, in: Parallel Problem Solving from Nature - PPSN
624 XIII, Vol. 8672 of LNCS, Springer, 2014, pp. 414–423.

- 625 [28] S. Kauffman, S. Levin, Towards a general theory of adaptive walks on
626 rugged landscapes, *Journal of Theoretical Biology* 128 (1) (1987) 11–45.
- 627 [29] G. Ochoa, M. Tomassini, S. Vérel, C. Darabos, A study of NK land-
628 scapes’ basins and local optima networks, in: *Genetic and Evolutionary*
629 *Computation Conference (GECCO)*, ACM, 2008, pp. 555–562.
- 630 [30] S. Vérel, F. Daolio, G. Ochoa, M. Tomassini, Local optima networks
631 with escape edges., in: *Artificial Evolution*, Springer, 2011, pp. 49–60.
- 632 [31] M. E. Yafrani, M. S. R. Martins, M. E. Krari, M. Wagner, M. R. B. S.
633 Delgado, B. Ahiod, R. Lüders, A fitness landscape analysis of the trav-
634 elling thief problem, in: *Genetic and Evolutionary Computation Confer-*
635 *ence (GECCO)*, ACM, 2018, pp. 277–284.
- 636 [32] D. Jakobovic, S. Picek, M. S. Martins, M. Wagner, A characterisation of
637 s-box fitness landscapes in cryptography, in: *Proceedings of the Genetic*
638 *and Evolutionary Computation Conference*, 2019, pp. 285–293.
- 639 [33] M. Tomassini, S. Verel, G. Ochoa, Complex-network analysis of com-
640 binatorial spaces: The NK landscape case, *Physical Review E* 78 (6)
641 (2008) 066114.
- 642 [34] F. Chicano, F. Daolio, G. Ochoa, S. Vérel, M. Tomassini, E. Alba, Local
643 optima networks, landscape autocorrelation and heuristic search perfor-
644 mance, *Parallel Problem Solving from Nature (PPSN)* (2012) 337–347.
- 645 [35] F. Daolio, S. Verel, G. Ochoa, M. Tomassini, Local optima networks and
646 the performance of iterated local search, in: *Genetic and Evolutionary*
647 *Computation Conference, GECCO*, ACM, 2012, pp. 369–376.
- 648 [36] F. Daolio, S. Verel, G. Ochoa, M. Tomassini, Local optima networks
649 of the permutation flow-shop problem, in: *International Conference on*
650 *Artificial Evolution (Evolution Artificielle)*, Springer, 2013, pp. 41–52.
- 651 [37] G. Ochoa, N. Veerapen, F. Daolio, M. Tomassini, Understanding phase
652 transitions with local optima networks: number partitioning as a case
653 study, in: *European Conference on Evolutionary Computation in Com-*
654 *binatorial 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.

- 684 2. $n = 5$: Figure .13 shows the difference between the two initialization
685 strategies, which we had not visualized before. Figure .14 shows the
686 complex neighborhoods for 1 000 samples.
- 687 3. $n = 6$: Figure .15 shows the individual neighborhoods for 1 000 samples.
688 Figure .16 shows the complex neighborhoods for 1 000 samples.
- 689 4. Figures .17-.20 show, for 10 000 samples, the results for the neighbor-
690 hoods 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	$fitness_1$	<i>flip</i>	121	520	8.5950	0.3526	0.0858	9.0000	—	0	2
			<i>insert</i>	438	3363	15.3562	0.5590	0.0381	2.8607	—	0	13
			<i>swap</i>	306	1948	12.7320	0.6407	0.0424	4.7516	—	0	14
			<i>flipinsert</i>	182	2494	27.4066	0.5753	0.1508	6.6374	2.0764	1	1
			<i>swapflip</i>	306	3514	22.9673	0.4806	0.0763	4.7516	3.0680	1	1
			<i>swapflipinsert</i>	363	5279	29.0854	0.4869	0.0807	4.3140	2.9272	1	1
		$fitness_2$	<i>swapinsert</i>	363	3325	18.3196	0.6810	0.0492	4.3140	—	0	11
			<i>flip</i>	185	849	9.1784	0.3165	0.0494	5.2973	—	0	2
			<i>insert</i>	626	5660	18.0831	0.4115	0.0297	2.0272	—	0	11
			<i>swap</i>	379	2236	11.7995	0.5123	0.0292	4.3140	—	0	20
			<i>flipinsert</i>	185	2132	23.0486	0.4563	0.1244	5.4216	—	0	2
			<i>swapflip</i>	379	3416	18.0264	0.4235	0.0458	4.3140	—	0	2
		$fitness_3$	<i>swapflipinsert</i>	382	4208	22.0314	0.4484	0.0578	4.2749	—	0	2
			<i>swapinsert</i>	382	3005	15.7330	0.5823	0.0392	4.2749	—	0	19
			<i>flip</i>	173	802	9.2717	0.3259	0.0588	5.5954	—	0	2
			<i>insert</i>	630	5630	17.8730	0.4157	0.0301	2.0397	—	0	9
			<i>swap</i>	387	2334	12.0620	0.5106	0.0333	4.2765	—	0	19
			<i>flipinsert</i>	173	2054	23.7457	0.4820	0.1400	5.7977	—	0	2
1 000	random	$fitness_1$	<i>swapflip</i>	387	3650	18.8630	0.4224	0.0459	4.2765	—	0	2
			<i>swapflipinsert</i>	390	4470	22.9231	0.4484	0.0600	4.2385	—	0	2
			<i>swapinsert</i>	390	3131	16.0564	0.5924	0.0409	4.2385	—	0	19
			<i>flip</i>	972	1	0.0021	0.0000	0.0000	3.1173	—	0	971
			<i>insert</i>	835	0	0.0000	0.0000	0.0000	2.1449	—	0	835
			<i>swap</i>	972	1	0.0021	0.0000	0.0000	2.6173	—	0	971
		$fitness_2$	<i>flipinsert</i>	987	0	0.0000	0.0000	0.0000	2.5502	—	0	987
			<i>swapflip</i>	972	1	0.0021	0.0000	0.0000	2.6173	—	0	971
			<i>swapflipinsert</i>	987	3	0.0061	0.0000	0.0000	2.5380	—	0	984
			<i>swapinsert</i>	987	2	0.0041	0.0000	0.0000	2.5380	—	0	985
			<i>flip</i>	803	0	0.0000	0.0000	0.0000	2.9178	—	0	803
			<i>insert</i>	692	0	0.0000	0.0000	0.0000	2.2934	—	0	692
		$fitness_3$	<i>swap</i>	832	6	0.0144	0.0000	0.0000	2.9075	—	0	826
			<i>flipinsert</i>	832	0	0.0000	0.0000	0.0000	2.9014	—	0	832
			<i>swapflip</i>	832	8	0.0192	0.0000	0.0000	2.9123	—	0	824
			<i>swapflipinsert</i>	832	10	0.0240	0.0000	0.0000	2.9038	—	0	822
			<i>swapinsert</i>	832	8	0.0192	0.0000	0.0000	2.8990	—	0	824
			<i>flip</i>	803	0	0.0000	0.0000	0.0000	2.9178	—	0	803
10 000	lex	$fitness_1$	<i>insert</i>	704	0	0.0000	0.0000	0.0000	2.2884	—	0	704
			<i>swap</i>	844	6	0.0142	0.0000	0.0000	2.8945	—	0	838
			<i>flipinsert</i>	844	0	0.0000	0.0000	0.0000	2.8886	—	0	844
			<i>swapflip</i>	844	8	0.0190	0.0000	0.0000	2.8993	—	0	836
			<i>swapflipinsert</i>	844	12	0.0284	0.0000	0.0000	2.8910	—	0	832
			<i>swapinsert</i>	844	10	0.0237	0.0000	0.0000	2.8863	—	0	834
		$fitness_2$	<i>flip</i>	732	6286	17.1749	0.3105	0.0233	14.2514	2.9103	1	1
			<i>insert</i>	3574	71540	40.0336	0.3816	0.0113	3.3898	—	0	11
			<i>swap</i>	1785	32130	36.0000	0.4864	0.0199	7.3647	—	0	14
			<i>flipinsert</i>	938	37871	80.7484	0.4886	0.0855	11.9872	2.1088	1	1
			<i>swapflip</i>	1785	52112	58.3888	0.3814	0.0326	7.3647	3.1423	1	1
			<i>swapflipinsert</i>	2035	88221	86.7037	0.3988	0.0429	6.8590	3.0030	1	1
		$fitness_3$	<i>swapinsert</i>	2035	62664	61.5862	0.5420	0.0302	6.8590	—	0	14
			<i>flip</i>	1835	13696	14.9275	0.2121	0.0085	5.3232	—	0	2
			<i>insert</i>	5289	99516	37.6313	0.2836	0.0072	2.1624	—	0	8
			<i>swap</i>	3797	56380	29.6971	0.3364	0.0078	4.3303	—	0	14
			<i>flipinsert</i>	1812	48743	53.8002	0.2743	0.0299	5.5425	2.9435	1	1
			<i>swapflip</i>	3796	75460	39.7576	0.2920	0.0105	4.3314	4.2448	1	1
10 000	random	$fitness_1$	<i>swapflipinsert</i>	3840	107445	55.9609	0.2797	0.0147	4.2745	3.8454	1	1
			<i>swapinsert</i>	3841	87823	45.7292	0.3417	0.0118	4.2734	—	0	14
			<i>flip</i>	1707	13253	15.5278	0.2230	0.0082	5.6473	—	0	2
			<i>insert</i>	5281	99007	37.4956	0.2841	0.0070	2.1888	—	0	9
			<i>swap</i>	3777	58117	30.7742	0.3377	0.0079	4.3818	—	0	14
			<i>flipinsert</i>	1683	47481	56.4242	0.2861	0.0336	5.9673	2.8311	1	1
		$fitness_2$	<i>swapflip</i>	3776	78943	41.8130	0.2926	0.0110	4.3829	4.1114	1	1
			<i>swapflipinsert</i>	3819	111198	58.2341	0.2819	0.0153	4.3263	3.7668	1	1
			<i>swapinsert</i>	3820	89878	47.0565	0.3470	0.0123	4.3251	—	0	14
			<i>flip</i>	9698	11	0.0023	0.0000	0.0000	3.1448	—	0	9687
			<i>insert</i>	8335	29	0.0070	0.0000	0.0000	2.1445	—	0	8306
			<i>swap</i>	9698	93	0.0192	0.0003	0.0000	2.6172	—	0	9606
		$fitness_3$	<i>flipinsert</i>	9831	88	0.0179	0.0000	0.0000	2.5442	—	0	9743
			<i>swapflip</i>	9697	130	0.0268	0.0005	0.0000	2.6176	—	0	9569
			<i>swapflipinsert</i>	9828	194	0.0395	0.0002	0.0000	2.5308	—	0	9635
			<i>swapinsert</i>	9829	153	0.0311	0.0000	0.0000	2.5305	—	0	9676
			<i>flip</i>	8033	5	0.0012	0.0000	0.0000	2.9512	—	0	8028
			<i>insert</i>	6848	28	0.0082	0.0000	0.0000	2.3112	—	0	6820
10 000	random	$fitness_1$	<i>swap</i>	8332	534	0.1282	0.0061	0.0000	2.9399	—	0	7881
			<i>flipinsert</i>	8353	43	0.0103	0.0000	0.0000	2.9251	—	0	8310
			<i>swapflip</i>	8326	744	0.1787	0.0084	0.0000	2.9455	—	0	7774
			<i>swapflipinsert</i>	8325	1108	0.2662	0.0115	0.0000	2.9348	—	0	7559
			<i>swapinsert</i>	8331	881	0.2115	0.0108	0.0000	2.9292	—	0	7648
			<i>flip</i>	8033	4	0.0010	0.0000	0.0000	2.9512	—	0	8029
		$fitness_2$	<i>insert</i>	6963	29	0.0083	0.0000	0.0000	2.3078	—	0	6934
			<i>swap</i>	8445	505	0.1196	0.0065	0.0000	2.9275	—	0	8010
			<i>flipinsert</i>	8468	40	0.0094	0.0000	0.0000	2.9127	—	0	8428
			<i>swapflip</i>	8438	690	0.1635	0.0082	0.0000	2.9333	—	0	7899
			<i>swapflipinsert</i>	8437	1058	0.2508	0.0102	0.0000	2.9230	—	0	7679
			<i>swapinsert</i>	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.

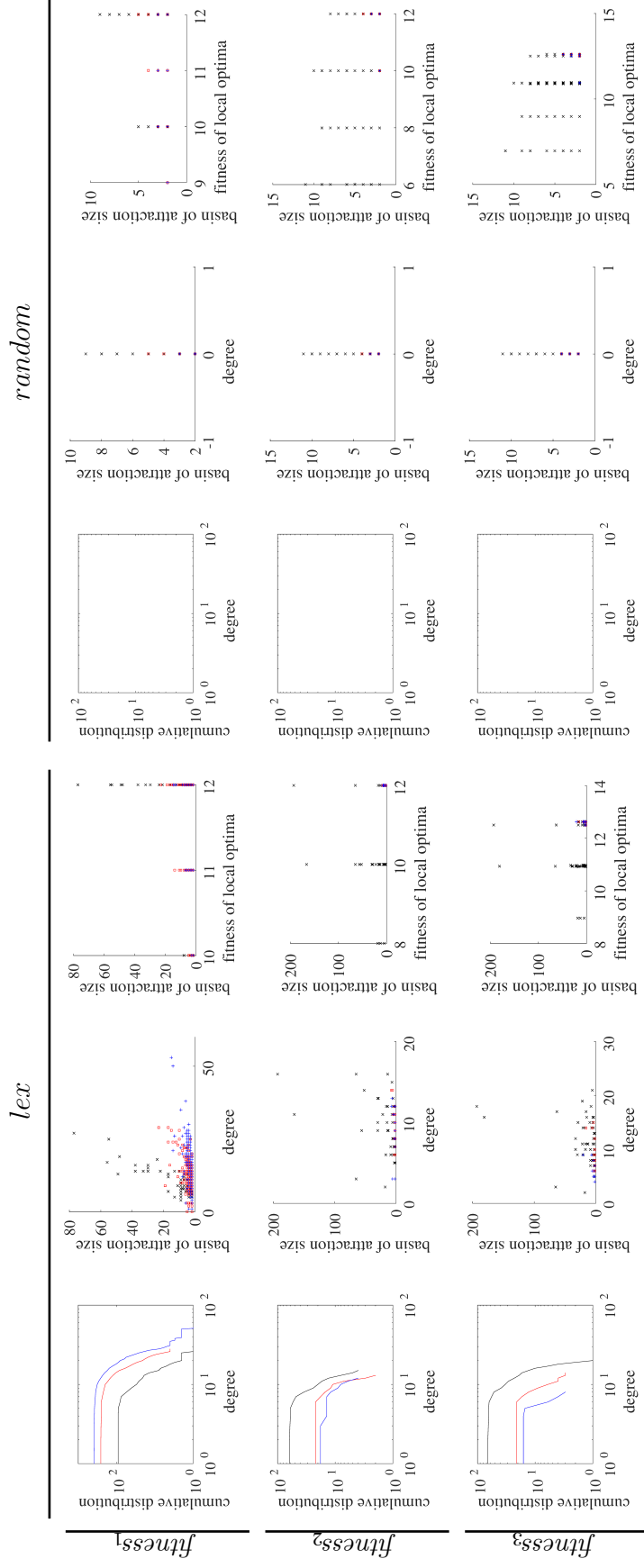


Figure 13: Statistical measures for *lex* (left) and *random* (right) initialization on $n = 5$ with the three fitness functions for operators flip (black), insert (blue), swap (red) using 1 000 samples (within each block of three): Cumulative degree distribution in a log-log scale (left), Correlation 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).

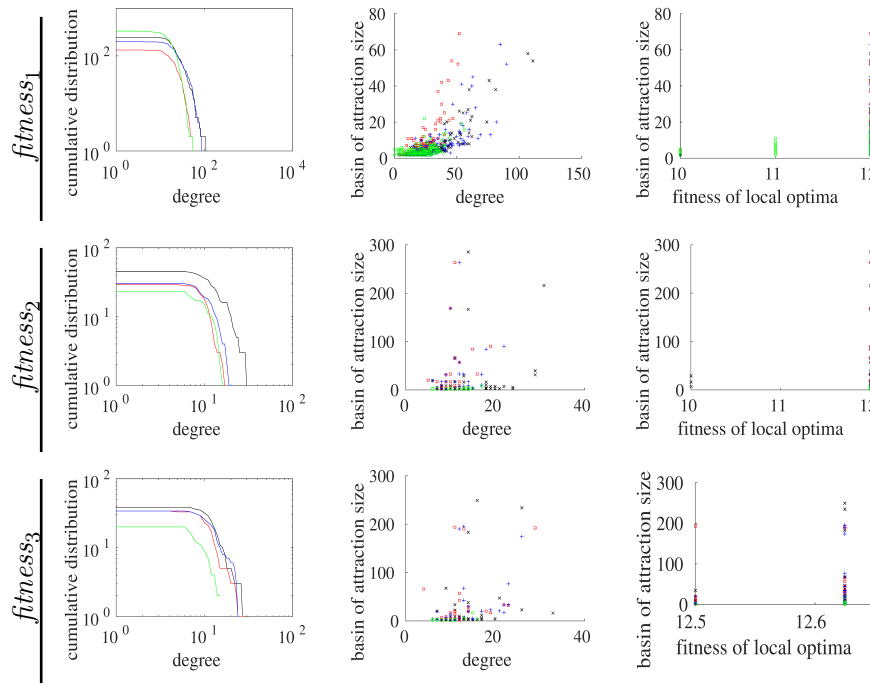


Figure .14: Statistical measures for *lex* initialization on $n = 5$ with the three fitness functions for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green) using 1 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation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

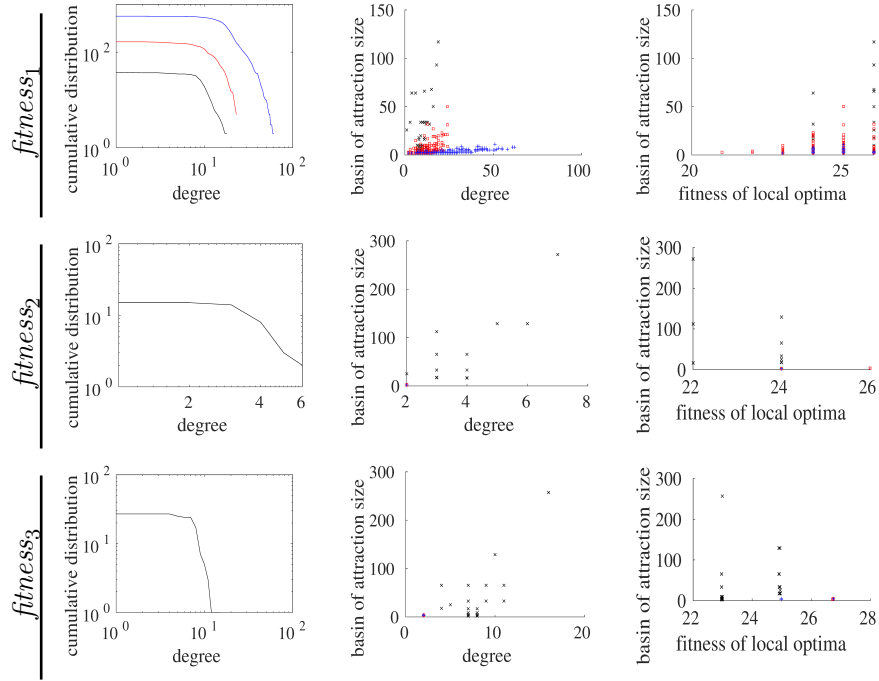


Figure .15: Statistical measures for *lex* initialization on $n = 6$ with the three fitness functions for operators flip (black), insert (blue), swap (red) using 1 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and their corresponding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

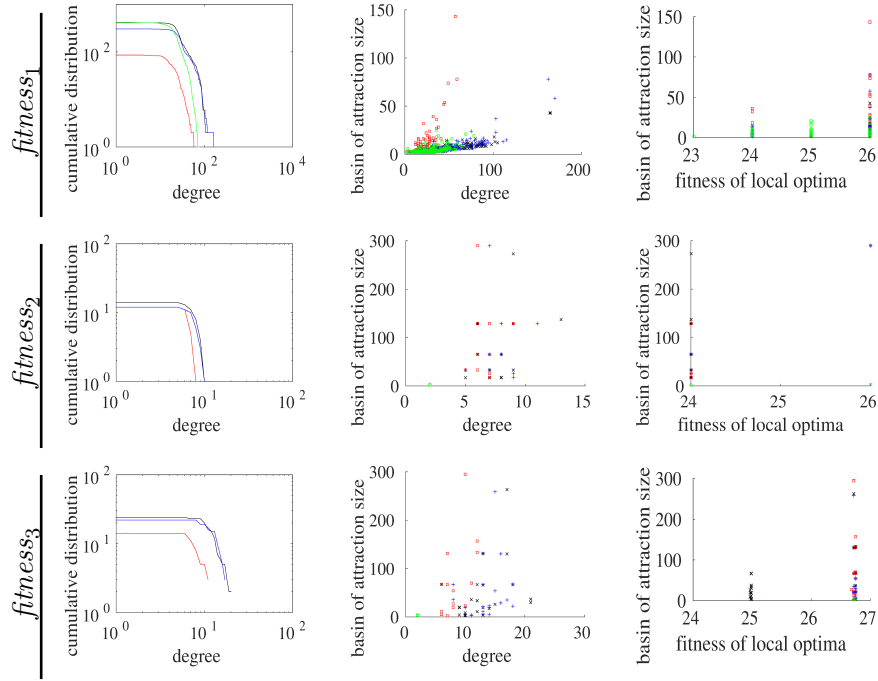


Figure .16: Statistical measures for *lex* initialization on $n = 6$ with fitness functions $fitness_2$ and $fitness_3$ for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green) using 1 000 samples: Cumulative degree distribution in a log-log scale (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).

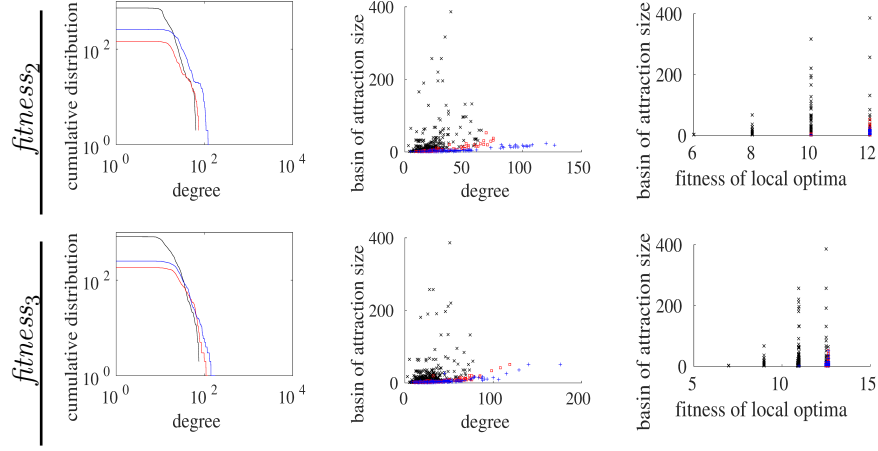


Figure .17: Statistical measures for lex initialization on $n = 5$ with fitness functions $fitness_2$ and $fitness_3$ for operators flip (black), insert (blue), swap (red) using 10 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and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

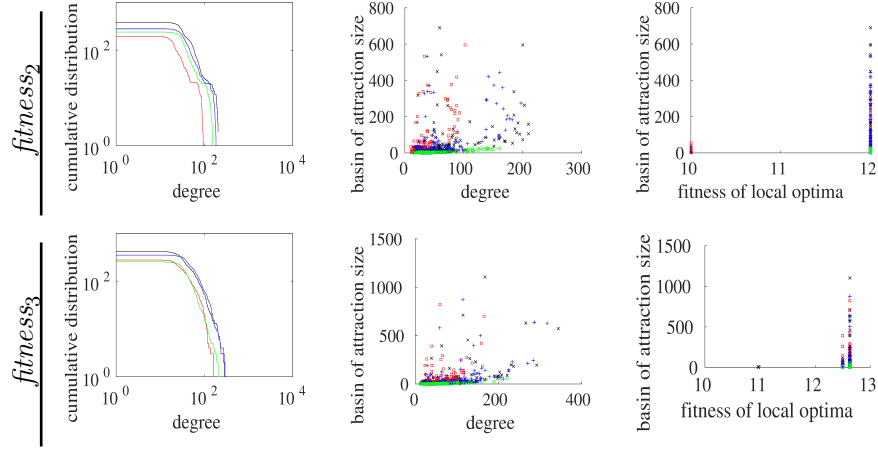


Figure .18: Statistical measures for lex initialization on $n = 5$ with fitness functions $fitness_2$ and $fitness_3$ for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green) using 10 000 samples: Cumulative degree distribution in a log-log scale (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).

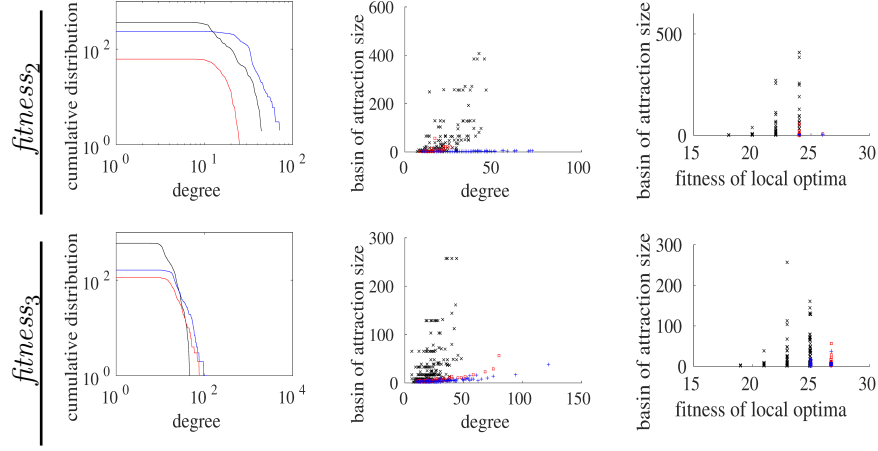


Figure .19: Statistical measures for lex initialization on $n = 6$ with fitness functions $fitness_2$ and $fitness_3$ for operators flip (black), insert (blue), swap (red) using 10 000 samples: Cumulative degree distribution in a log-log scale (left), Correlation between the degree of local optima and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

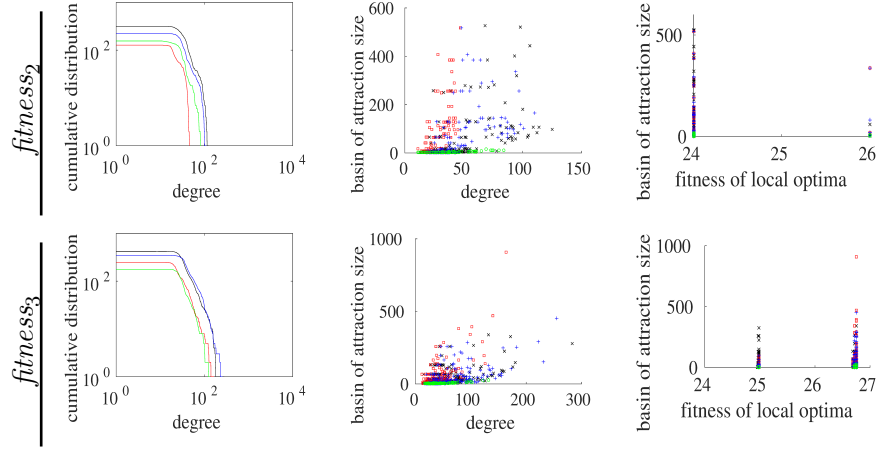


Figure .20: Statistical measures for lex initialization on $n = 6$ with fitness functions $fitness_2$ and $fitness_3$ for combination using operators flipinsert (black), swapflipinsert (blue), swapflip (red), and swapinsert (green) using 10 000 samples: Cumulative degree distribution in a log-log scale (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).