Crossover for Cardinality Constrained Optimization

Tobias Friedrich
tobias.friedrich@hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

Timo Kötzing
timo.koetzing@hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

Aishwarya Radhakrishnan
aishwarya.radhakrishnan@hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

Leon Schiller
leon.schiller@student.hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

Martin Schirneck
martin.schirneck@hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

Georg Tennigkeit
georg.tennigkeit@student.hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

Simon Wietheger
simon.wietheger@student.hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

ABSTRACT

In order to understand better how and why crossover can benefit optimization, we consider pseudo-Boolean functions with an upper bound B on the number of 1s allowed in the bit string (cardinality constraint). We consider the natural translation of the OneMax test function, a linear function where B bits have a weight of $1 + \varepsilon$ and the remaining bits have a weight of 1. The literature gives a bound of $\Theta(n^2)$ for the (1+1) EA on this function.

Part of the difficulty when optimizing this problem lies in having to improve individuals meeting the cardinality constraint by flipping both a 1 and a 0. The experimental literature proposes balanced operators, preserving the number of 1s, as a remedy. We show that a balanced mutation operator optimizes the problem in $O(n \log n)$ if $n - B = O(1)$. However, if $n - B = \Theta(n)$, we show a bound of $\Omega(n^2)$, just as classic bit flip mutation. Crossover and a simple island model gives $O(n^2/\log n)$ (uniform crossover) and $O(n\sqrt{n})$ (3-ary majority vote crossover). For balanced uniform crossover with Hamming distance maximization for diversity we show a bound of $O(n \log n)$.

As an additional contribution we analyze and discuss different balanced crossover operators from the literature.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**; *Optimization with randomized search heuristics*; *Evolutionary algorithms.*

GECCO '22, July 9–13, 2022, Boston, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9237-2/22/07…$15.00
https://doi.org/10.1145/3512290.3528713

KEYWORDS

Balanced crossover, balanced mutation, constraint optimization, run time analysis.

ACM Reference Format:
Tobias Friedrich, Timo Kötzing, Aishwarya Radhakrishnan, Leon Schiller, Martin Schirneck, Georg Tennigkeit, and Simon Wietheger. 2022. Crossover for Cardinality Constrained Optimization. In *Proceedings of The Genetic and Evolutionary Computation Conference 2022 (GECCO '22).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3512290.3528713

1 INTRODUCTION

The analysis of crossover, asking how, why and when it helps optimization, is one of the most important topics for understanding evolutionary computation. In practice, hardly any application in the field proceeds without crossover. It is known to aid the search, but its theoretical background still remains somewhat in the dark.

Focusing on the optimization of pseudo-Boolean functions (that is, $f\colon \{0,1\}^n \to \mathbb{R}$), most theory literature on the topic has used the Jump function as a setting to understand crossover, showing that crossover can jump a fitness valley effectively where mutation fails [Jansen and Wegener 2002; Kötzing et al. 2011]. Extensions of this work studied how the required diversity emerges [Dang et al. 2018] and the use of diversity mechanisms [Dang et al. 2016].

A notable exception from this topical focus is the work by Sudholt [2017] on how crossover can speed-up the optimization of the so-called OneMax function[1] by a factor of 2. This was a seminal result since it considered crossover in a more natural setting than Jump, which gives crossover an unfair advantage by placing the optimum where crossover frequently generates its output [Jansen 2015]. Similarly, Antipov et al. [2020] showed speedups for OneMax in a more intricate setting of crossover.

In this work, we investigate a natural model for constrained optimization in evolutionary computation. Friedrich et al. [2020] suggested an extension of OneMax, which we call BoundMax$_B$,

[1]OneMax maps a bit string to its number of 1-bits.

parameterized by the bound $B \in \mathbb{N}$ on the maximum allowed number of ones. This function is defined so that, for all $x \in \{0,1\}^n$ and with $|x|_1$ being the number of ones in x,

$$\textsc{BoundMax}_B(x) = \begin{cases} \sum_{i=1}^{B}(1+1/n)x_i + \sum_{i=B+1}^{n} x_i & \text{if } |x|_1 \le B; \\ -|x|_1 & \text{otherwise.} \end{cases}$$

This linear function has a unique optimum with exactly B many ones in positions 1 through B. The first phase of the (1+1) EA[2] when maximizing $\textsc{BoundMax}_B$ typically collects 1-bits with only a slight preference for the first B positions (which we call *heavy*; in contrast to the other positions, which we call *light*). Once a solution at the cardinality bound B is found, it can only be improved by swapping out light bits for heavy ones. As shown by Friedrich et al. [2020], for some choice of B, the (1+1) EA takes time $\Theta(n^2)$ to optimize this function. The given B was cn for some c with $1/2 < c < 1$, we extend this to values of $B = n - d$, where d is a constant.

Intuitively, the problem when optimizing at the constraint is that both a 0 and a 1 need to be flipped (in particular, for an almost optimal solution, a specific heavy 0 and a specific light 1). The first remedy that comes to mind is to use an operator that does exactly that. *Swap mutation* has been suggested in experimental works [Chen and Hou 2006; Manzoni et al. 2020; Meinl and Berthold 2009] alongside the *balanced crossover*, that is, crossover that keeps the same number of ones in the output. In a sense, this opens up the "black box" setting (which assumes no knowledge about the fitness function) into more of a "gray box", taking the specific problem knowledge into account that the number of ones in a solution carries importance. This is not only useful in our constrained setting, a special case of the knapsack problem, but also for other combinatorial problems that assign importance to the number of ones. For example, finding a minimum spanning tree requires finding exactly $n-1$ edges, a minimal vertex cover tries to minimize the number of picked vertices, and so on. Understanding which general structural properties lead to better optimization performance on a wide range of different problems can thus be very beneficial [Whitley et al. 2016]. An operator swapping a 1 and a 0 would sacrifice the full unbiasedness[3] but maintains problem-specific unbiasedness (see [Rowe and Vose 2011]). Note that a balanced operator always needs to be paired with regular mutation, otherwise it is confined to only explore solutions with the same number of 1-bits.

We define swap mutation formally in Section 3.2 and give a rigorous analysis in Section 5. In particular, we show that it achieves an optimization time of $O(n\log(n))$ on $\textsc{BoundMax}_B$ for $B = n - d$ with d constant, intuitively being able to find a heavy 0 among constantly many 0s to swap easily (Theorem 5.5). However, for $B = cn$ with $1/2 < c < 1$, we get a lower bound of $\Omega(n^2)$ also for this operator, intuitively because there are a linear number of both 1- and 0-bits at the constraint.

We also use the regime of $B = cn$ with $1/2 < c < 1$ to investigate the effectiveness of crossover. (The bound $B = n - d$ for constant d gives a quadratic optimization time for all considered cases.) The use of crossover typically requires some kind of diversity, and while some diversity might emerge on its own [Dang et al. 2018], we employ specific diversity mechanisms. We start by considering *island models* [Doerr et al. 2019; Neumann et al. 2011], in particular the *single-receiver model* [Watson and Jansen 2007]. Mutation-based algorithms are run in parallel on *islands* and their individuals are then considered for crossover at the single receiver. This maintains perfect independence of the individuals for maximum diversity.

We show that *uniform crossover*[4] with the single-receiver island model can now optimize $\textsc{BoundMax}_B$ in time $O(n^2/\log n)$, an asymptotic speedup over mutation-based algorithms. Roughly speaking, when only logarithmically many incorrect bits are left on each island, they are likely to be in different positions and crossover finds the correct offspring in time exponential in the number of incorrect bits. Note that, from the birthday paradox, we know that if at each island the resident individual has only $O(\sqrt{n})$ many incorrect bits, then likely the islands have their incorrect bits at different positions. This motivates us to use a ternary operator, the deterministic *majority vote* [Friedrich et al. 2016], taking three parents and using the majority bit at each position for the offspring. Here we can show a run time bound of $O(n\sqrt{n})$ on $\textsc{BoundMax}_B$. Mutation reduces the number of incorrect bits at each island within $O(n\sqrt{n})$ iterations to be $O(\sqrt{n})$, then majority vote is successful. Note that we only provide upper bounds, so a final decision on how the two algorithms compare cannot be made. However, we believe that these bounds are tight, as mutation alone would be slower, and crossover can be applied only once to speed up optimization (since any offspring of crossover can never be the parent for another mutation nor crossover) in the single-receiver island model.

In order to mitigate the disadvantage of the single-receiver setting, we consider a genetic algorithm (GA) with the diversity mechanism of *Hamming distance maximization* (but many others would serve equally well, see [Dang et al. 2016]). We use an idea of Sudholt [2017] and show that a population of equally good individuals first spreads in Hamming distance and then gains in quality with crossover, making the process so fast that we obtain the desirable bound of $O(n\log n)$ for the optimization of $\textsc{BoundMax}_B$. Note that this improvement arises naturally from the interplay of crossover and mutation and our analysis gives insight into this interplay.

Note that, for the last result, we employ a *balanced* uniform crossover: given two parents, the matching bits are inherited and exactly half of the remaining bits are set to 1. This operator produces the same number of 1s in the offspring as are in the parents. Using standard uniform crossover instead can delay optimization by producing offspring with the wrong number of ones.

The literature already discusses a wide range of balanced crossover operators [Chen and Hou 2006; Manzoni et al. 2020; Meinl and Berthold 2009], but we find that all of them lack other useful properties. They either do not bequeath matching bits or they are not order-unbiased (as defined in [Lehre and Witt 2012]). To close this gap, we introduce the *balanced uniform crossover*, which is fully unbiased and has all the desired properties.

Table 1 gives a summary of our results. The remainder of this paper is structured as follows. After fixing the notation in Section 2,

[2]The (1+1) *evolutionary algorithm* ((1+1) EA) maintains a best-so-far solution and, in each iteration, makes a copy of it, flips each bit independently with probability $1/n$ and keeps the new solution if it is at least as good as the previous one.

[3]An operator is *unbiased* if, when given two problem instances where one is a perturbation – permutation of bit order and bit flips – of the other, it performs analogously. See [Lehre and Witt 2012].

[4]The output of the uniform crossover takes each bit independently from a parent chosen uniformly at random.

Algorithm variant	$n - B = O(1)$	$n - B = \Theta(n)$	Reference
(1+1) EA	$\Theta\left(n^2\right)$	$\Theta\left(n^2\right)$	[Friedrich et al. 2020] Proposition 5.2
(1+1) Swap-EA	$O(n\log(n))$	$\Theta(n^2)$	Theorem 5.5 Theorem 5.6
(2+1) islands, (balanced) uniform	$\Theta\left(n^2\right)$	$O\left(\frac{n^2}{\log(n)}\right)$	Theorem 6.9 Theorem 6.4
(3+1) islands, majority vote	$\Theta\left(n^2\right)$	$O(n\sqrt{n})$	Theorem 6.9 Theorem 6.5
(2+1) GA, balanced uniform, with diversity	$\Theta\left(n^2\right)$	$O(n\log(n))$	Theorem 6.9 Theorem 6.8
(2+1) Swap-GA, balanced uniform, with diversity	$O(n\log(n))$	$O(n\log(n))$	Theorem 6.10

Table 1: Overview of expected run times on BoundMax$_B$. The Swap-EA uses swap and standard mutation. Islands run the (1+1) EA and a single receiver with the stated crossover. The GA uses standard mutation and balanced uniform crossover, the Swap-GA additionally has swap mutation.

we review crossover and mutation operators from the literature in Section 3. We introduce the balanced uniform crossover operator in Section 4 and then examine the optimization time for BoundMax$_B$ under balanced mutation (Section 5) and crossover (Section 6).

2 PRELIMINARIES

For some positive integer $n \in \mathbb{N}^+$, symbol $[n]$ stands for the set $\{1, \dots, n\}$. An array $A = [a_1, a_2, \dots, a_n]$ is a sequence of natural numbers (incl. 0). For $i, \ell, r \in [n]$, we use $A[i] = a_i$ to refer to its ith entry, and $A[\ell, r] = [a_\ell, a_{\ell+1}, \dots, a_r]$ for the subarray from indices ℓ to r (if $\ell > r$, then $A[\ell, r]$ is empty). A bit string of length n is any element of $\{0, 1\}^n$. We refer to its ith bit by x_i, so $x = x_1x_2 \dots x_n$; similarly, for $M \subseteq [n]$, x_M denotes the bit string obtained by removing all bits with indices not in M, and $x[\ell, r]$ is the bit string $x_\ell x_{\ell+1} \dots x_r$. We use $|x|_1 = \sum_{i=1}^n x_i$ and $|x|_0 = n - |x|_1$ the number of 1- and 0-bits of x, respectively. For two bit strings x, y, we denote their concatenation as xy. The strings with n 0- or 1-bits are 0^n and 1^n, respectively. A permutation is bijective function $\sigma\colon [n] \to [n]$. We write $\sigma = [a_1, a_2, \dots, a_n]$ to refer to the permutation where, for every $i \in [n]$, $\sigma(i) = a_i$. For every permutation σ of $[n]$ and every bit string x, we define the permutation of x respective to σ as $\sigma_b(x) = x_{\sigma(1)}x_{\sigma(2)} \dots x_{\sigma(n)}$. For example, for $x = 10011$ and $\sigma = [3, 1, 5, 2, 4]$, we get $\sigma_b(x) = 01101$. An event occurs *with high probability* (w.h.p.) with respect to n, if it has probability at least $1 - 1/n^c$ for some constant $c > 0$.

3 BALANCED CROSSOVER AND MUTATION

A *crossover operator* is a potentially randomized algorithm that maps two parent bit strings $x, y \in \{0, 1\}^n$ to an output bit string $z \in \{0, 1\}^n$. We identify operators with their output distribution, where $p_A(z \mid x, y)$ denotes the probability that operator A samples z given inputs x, y. In this paper, we investigate *balanced* crossover operators that retain the same number of ones as the input strings. We would also like to have certain additional regularity properties such as invariance under permutations (*order unbiasedness*[5]) and that, whenever the parents agree on a bit position, this bit is sampled in the output (*inheritance respect*[6]).

Definition 3.1 (Regularity properties). A crossover operator A is

(1) *balanced* if $|x|_1 = |y|_1 \neq |z|_1$ implies $p_A(z \mid x, y) = 0$;
(2) *order-unbiased* if $p_A(z \mid x, y) = p_A(\sigma_b(z) \mid \sigma_b(x), \sigma_b(y))$ holds for all permutations σ of $[n]$;
(3) *inheritance-respectful* if $x_i = y_i \neq z_i$ for some $i \in [n]$ implies $p_A(z \mid x, y) = 0$.

The idea of order unbiasedness is that the order of encoding the bits in the string should not matter for the algorithm. A crossover should be inheritance-respectful if it wants to combine strengths of different solutions (and leave creating novelty to mutation). This was studied in the literature as *geometric crossovers* [Moraglio and Poli 2004]: crossovers are inheritance-respectful if and only if they are geometric crossovers under the Hamming distance metric.

3.1 Existing Crossover Operators

We review several crossover operators from the literature and classify them according to the regularity properties. We find that none of the existing crossovers are balanced, order-unbiased and inheritance-respectful at the same time. We propose the *balanced uniform crossover* as an alternative in Section 4. (The *boring crossover* that returns some parent also satisfies all three properties.)

We compare nine crossover operators: the uniform, single- and two-point crossover from the survey by Katoch et al. [2021], the counter-based, zero lengths and map-of-ones crossover by Manzoni et al. [2020], the shrinking crossover of Chen and Hou [2006] and finally the balanced two-point crossover and alternating crossover[7] by Meinl and Berthold [2009]. Our results are summarized in Table 2. We give some intuition below, but omit proofs due to space constraints. In Section 3.2, we briefly discuss balanced mutation.

Uniform, single-point and two-point crossovers. The very common *uniform crossover operator* iterates over all bit positions and, for each of them, selects a random parent and takes its bit at this position. The *single-point crossover* chooses a random position in $[n]$ and swaps all bits after that position between the parents. For swap position s, this results in the two strings $x[1, s]\, y[s{+}1, n]$ and $y[1, s]\, x[s{+}1, n]$. Without loosing generality, we define as output the string that starts with the bits of x. The *two-point crossover* samples two indices and swaps the bit-range between them among the parents. Intuitively, none of those operators can be balanced as they take 1-bits from the parents regardless of their total number.

Throughout this section, we let s be an index drawn uniformly at random from $[n]$. For two indices that are independently and

[5]See Lehre and Witt [2012] for a discussion; they call this property "σ-invariance".
[6]See Radcliffe [1994]. He distincts respect (shared properties of the parents are inherited) and gene transmission (every property of an offspring is inherited from at least one parent). These concepts are equivalent when limited to bit strings. We use the more telling term *inheritance respect*.
[7]The alternating crossover is called "balanced uniform crossover" in the original work [Meinl and Berthold 2009]. We reserve this name for Section 4.

Crossover Operator	Balanced	Order-Unbiased	Inheritance-Respectful
Uniform	×	✓	✓
Single-Point	×	×	✓
Two-Point	×	×	✓
Counter-Based	✓	×	×
Zero Lengths	✓	×	×
Map-of-Ones	✓	✓	×
Shrinking	✓	×	✓
Balanced Two-Point	✓	×	×
Alternating	✓	×	✓
Boring	✓	✓	✓
Balanced Uniform	✓	✓	✓

Table 2: Overview of different crossover operators, where a ✓ denotes that the given operator does have the stated property, and × that it does not.

uniformly distributed in $[n]$, we use symbol ℓ to denote the smaller one and r for the larger.

Definition 3.2 (Uniform, single-point and two-point crossover). On input $x, y \in \{0,1\}^n$, the output string $z \in \{0,1\}^n$ is as follows.

(1) *Uniform crossover*: For every $i \in [n]$, bit z_i is uniformly distributed in $\{x_i, y_i\}$, independently of all other choices.
(2) *Single-point crossover*: $z = x[1, s{-}1]\, y[s, n]$.
(3) *Two-point crossover*: $z = x[1, \ell{-}1]\, y[\ell, r]\, x[r{+}1, n]$.

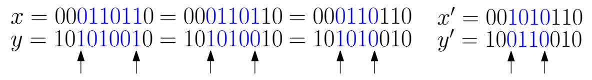
Counter-based, zero lengths and map-of-ones crossovers. To heal the unbalancedness of the uniform crossover, the *counter-based crossover* stops once the number of ones/zeros in the output are equal to that of the parent x. As the bits are fixed one by one in order of increasing index, the operator cannot be order-unbiased.

Definition 3.3 (Counter-based crossover). Inductively for all indices $i \in [n]$, let $z_{\leq i}$ be the string of already bits set after the ith iteration. The bit z_i is a uniform random value in $\{x_i, y_i\}$, independently of the previous choices. If $|z_{\leq i}|_1 = |x|_1$, the final output is $z = z_{\leq i}\, 0^{n-i}$; if $|z_{\leq i}|_0 = |x|_0$, the output is $z_{\leq i}\, 1^{n-i}$. Otherwise, the operator continues to the next iteration.

For the other two operators, we need alternative ways to represent bit strings. The zero lengths representation is effectively a run-length encoding of consecutive 0-bits. Let $x \in \{0,1\}^n$ be such that $|x|_1 = k$ and $x = 0^{a_1} 1 0^{a_2} 1 \ldots 0^{a_k} 1 0^{a_{k+1}}$ with $a_i \in \mathbb{N}_0$. Then, x's *zero lengths array* is $Z_x = [a_1, a_2, \ldots, a_k, a_{k+1}]$. The $k+1$ elements[8] of Z_x necessarily sum to $n - k$. Conversely, the *map-of-ones array* M_x contains the indices of all 1-bits in x. Any permutation of M_x also represents x, we tacitly use set notation if no ambiguity arises.

The *zero lengths crossover* constructs the zero lengths array of the output from that of the parents by picking each entry uniformly at random from the corresponding parent entries. It additionally

[8]In [Manzoni et al. 2020], the authors claim that the zero lengths array has $n - k + 1$ entries. This is probably a typo, their number really is $k + 1$.



$x = 000110110 = 000110110 = 000110110 \quad x' = 001010110$
$y = 101010010 = 101010010 = 101010010 \quad y' = 100110010$

Figure 1: Example run of the shrinking crossover operator. The initial range is shrinked twice. Then, the bits inside the range contain the same number of ones and are swapped.

ensures that the sum of elements does not exceed the number of zeros. The *map-of-ones crossover*, in each iteration, chooses a random parent string and samples one of its stored 1-indices that has not already been selected. Both operators enforce balancedness, but lose the inheritance respect through the representations.

Definition 3.4 (Zero lengths and map-of-ones crossover). Let $k = \min(|x|_1, |y|_1)$. The representation of output z is defined as follows.

(1) *Zero lengths crossover*: For all $i \in [k]$, let $s_{i-1} = \sum_{j=1}^{i-1} Z_z[j]$ be the sum of set entries. $Z_z[i] = \min(a_i, n{-}k{-}s_{i-1})$, with a_i uniformly distributed in $\{Z_x[i], Z_y[i]\}$; $Z_z[k{+}1] = n{-}k{-}s_k$.
(2) *Map-of-ones crossover*: For all $i \in [k]$, let u_i be uniformly distributed vector in $\{x, y\}$, entry $M_z[i]$ is set to a uniform random index in $M_{u_i}[1,k] \backslash M_z[1, i{-}1]$.

Shrinking, balanced two-point and alternating crossover. Both the shrinking and balanced two-point crossover operator are based on the regular two-point version. The *shrinking crossover* works on the classical bit string representation. Before swapping the sampled bit-ranges the contained ones are counted. If they are not equal between the parents, the index r is reduced until they are. See Figure 1. The reduction may result in $r = \ell - 1$, that is, an empty range, the output then equals x. The *balanced two-point crossover* applies the two-point operator to the map-of-ones representations of the parent strings. This may sample duplicate 1-indices. To recover balancedness, those are replaced by elements from the unchosen range of M_x. Finally, the *alternating crossover* also works on map-of-ones but is deterministic. It combines M_x, M_y in a sorted manner, keeping duplicates, and takes every other 1-index.

Below, let $A \diamond B$ denote the concatenation of arrays A and B.

Definition 3.5 (Shrinking, balanced two-point and alternating crossover.). Let $k = \min(|x|_1, |y|_1)$.

(1) *Shrinking crossover*: Let r' be the largest index $\ell \leq r' \leq r$ such that $|x[\ell, r']|_1 = |y[\ell, r']|_1$, or $r' = \ell - 1$ if there is none. The output string is $z = x[1, \ell{-}1]\, y[\ell, r']\, x[r'{+}1, n]$.
(2) *Balanced two-point*: Let u, v, with $u \leq v$, be independently and uniformly distributed in $[k]$ and $M = M_x[1, u{-}1] \diamond M_y[u, v] \diamond M_x[v{+}1, |x|_1]$. The output representation M_z is generated from M by removing duplicates, sampling uniform random replacements from $M_x[u, v] \backslash M_z$.
(3) *Alternating crossover*: Let M be the result of sorting $M_x \diamond M_y$. The output is $M_z = M[1] \diamond M[3] \diamond \cdots \diamond M[2k-1]$.

Theorem 3.6. *The operators have the properties as given in Table 2.*

3.2 Balanced Mutation

A *mutation operator* is an algorithm mapping a single parent bit string $x \in \{0,1\}^n$ to an offspring $z \in \{0,1\}^n$. The most common mutation operator is *standard bit mutation* (also called *uniform mutation*), flipping each bit of x independently with probability $\frac{1}{n}$. This does not preserve the number of ones. The asymmetric mutation operator from [Jansen and Sudholt 2010; Neumann and Wegener 2007] does so in expectation, but does not guarantee it. Adapting the notation of Definition 3.1, a mutation operator is *balanced* if $|z|_1 \neq |x|_1$ implies $p_A(z \mid x) = 0$. The literature discusses balanced mutation just very briefly, see [Chen and Hou 2006; Manzoni et al. 2020; Meinl and Berthold 2009]. The only notable operator is *swap mutation*. It chooses a 1- and a 0-bit of x uniformly at random and swaps them. (The strings 1^n and 0^n remain unchanged.) We use the name (1+1) *swap-evolutionary algorithm* ((1+1) Swap-EA) for the variant of the (1+1) EA that, in the mutation step, uses the swap operator with probability p_b and standard bit mutation otherwise.

4 NEW BALANCED CROSSOVER OPERATOR

None of the crossovers discussed are balanced, order-unbiased and inheritance-respectful. Of course, one could achieve all this by using what we call the *boring crossover* that merely returns one of the parents (say, at random), but it defeats the purpose of recombining the strengths of multiple individuals. We propose the *balanced uniform crossover*[9] operator instead. To ensure inheritance respect, it first fixes all positions in which the input strings are equal. For the others, it randomly samples as many positions as required to match the number of ones in the input strings.

Definition 4.1 (Balanced uniform crossover). Define the sets $F = \{i \in [n] \mid x_i = y_i\}$ and $\overline{F} = [n]\backslash F$. Let the set $I \subseteq \overline{F}$ with $|I| = \lfloor |\overline{F}|/2 \rfloor$ be drawn uniformly at random. The output bit string z has $z_i = x_i$ for all $i \in F$, $z_i = 1$ for $i \in I$ and $z_i = 0$ for $i \in \overline{F}\backslash I$.

Theorem 4.2. *The balanced uniform crossover operator is balanced, inheritance-respectful and order-unbiased.*

The next theorem gives an alternative distributional characterization of the balanced uniform crossover.

Theorem 4.3. *The output distribution of the balanced uniform crossover is the same as that of the uniform crossover conditioned on having $|z|_1 = \lfloor (|x|_1 + |y|_1)/2 \rfloor$ 1-bits in the output.*

5 OPTIMIZING WITHOUT CROSSOVER

Here, we first show that the (1+1) EA has a quadratic lower bound on BoundMax$_B$, even if $n - B = O(1)$. For this case, we then break this bound by introducing balanced mutation: the (1+1) Swap-EA has an expected run time in $O(n\log(n))$. For the case $n - B = \Theta(n)$, however, the run time remains in $\Omega(n^2)$. In Section 6, we show how crossover can improve optimization time in this case.

Theorem 5.1 ([Friedrich et al. 2020], Theorem 10). *There exists a constant $0 < c < 1$ such that the (1+1) EA using only standard mutations takes time $\Omega(n^2)$ to optimize BoundMax$_B$ under uniform constraint $B = cn$, not only in expectation but even w.h.p.*

It is easy to verify that their proof remains valid for $B = cn$ for *all* constants $0 < c < 1$ and, in fact, even for $B = n - n^c$. Below, we extend this to the case that the bound differs from the maximum n only by an *additive* constant, which is not covered by [Friedrich et al. 2020].

[9]As mentioned, we discuss what is called "balanced uniform crossover" in [Meinl and Berthold 2009] as the alternating crossover.

Proposition 5.2. *There is an integer constant $d > 0$ such that the (1+1) EA using only standard mutation takes time $\Omega(n^2)$ to optimize BoundMax$_B$ under uniform constraint $B = n - d$, not only in expectation but even w.h.p.*

Recall that the (1+1) Swap-EA applies swap mutation in each round with probability p_b. Next, we prove that it outperforms the (1+1) EA if $n - B = O(1)$. To this end, we start with showing that the expected optimization time from the point on where the first individual with B 1-bits is sampled is $O(B(n - B))$.

Lemma 5.3. *For any $0 < p_b < 1$, consider the (1+1) Swap-EA on BoundMax$_B$ with uniform constraint B. Assume that the initial individual has exactly B 1-bits and let T be the random variable describing the number of steps until the algorithm finds the optimal solution. Then $E[T] \leq \frac{B(n-B)\pi^2}{6p_b}$.*

Proof. We use the fitness level method to derive the upper bound. For this, consider the fitness-based partition $A_0, \ldots, A_B$ of $\{0,1\}^n$ where A_i is the set of bit strings that have exactly i heavy 1-bits. Since the number of 1-bits in the offspring can neither increase nor decrease compared to its parent, this is a valid partitioning and the optimum is reached when an individual in A_B is sampled.

Let p_i be a lower bound on the probability of jumping to a fitness level above A_i assuming that the current individual is in A_i. Then, the fitness level method allows us to estimate an upper bound for $E[T]$ as $E[T] \leq \sum_{i=0}^{B-1} \frac{1}{p_i}$ (cf. [Wegener 2002, Section 8 Lemma 1]).

We show that the probability p_i is lower-bounded by $p_i \geq p_b \frac{(B-i)^2}{B(n-B)}$. For this, consider a bit string $x \in A_i$ recall that it has i heavy 1-bits. The swap mutation operator converts x to a higher fitness level if it swaps one of the $B - i$ heavy 0-bits and one of the $B - i$ light 1-bits. There are B 1-bits and $n - B$ 0-bits, since we start with an individual with B 1-bits and cannot lose 1-bits as this would decrease fitness. Hence, an improving swap happens with probability $\frac{B-i}{B} \cdot \frac{B-i}{n-B} = \frac{(B-i)^2}{B(n-B)}$. Since the swap mutation operator is invoked with probability p_b, we obtain $p_i \geq p_b \frac{(B-i)^2}{B(n-B)}$.

Using the fitness level method, this means $E[T]$ is at most

$$\sum_{i=0}^{B-1} \frac{1}{p_i} \leq \frac{B(n-B)}{p_b} \sum_{i=0}^{B-1} \frac{1}{(B-i)^2} = \frac{B(n-B)}{p_b} \sum_{i=1}^{B} \frac{1}{i^2} \leq \frac{\pi^2 B(n-B)}{6p_b}$$

since the series $\sum_{i=1}^{\infty} \frac{1}{i^2}$ converges to $\pi^2/6$ (the Basel problem). □

For the time until the first individual with B 1-bits is sampled, we get the following upper bound. The proof extends the argument of Friedrich et al. [2020, Theorem 9] to the notion that using balanced mutation with constant probability only adds a constant factor.

Lemma 5.4. *For any $0 \leq p_b < 1$, consider the (1+1) Swap-EA on BoundMax$_B$ with uniform constraint B. Let T be the random variable describing the number of steps until the algorithm finds a solution with exactly B 1-bits. Then, $E[T]$ is in $O(n\log(n))$.*

Combining Lemmas 5.3 and 5.4 now yields the following bounds.

Theorem 5.5. *For any $0 < p_b < 1$, consider the (1+1) Swap-EA on BoundMax$_B$. Let T be the random variable describing the number of steps until the algorithm finds the optimum. Assume that p_b is constant and that $B = n - d$. If $d = O(\log(n))$, then $E[T]$ is in $O(n\log(n))$. If instead $d = \Omega(\log(n))$, $E[T]$ is in $O(nd)$.*

Next, we show that the run time bound of $O(n^2)$ for the case of $B = cn$ is tight by giving a matching lower bound.

Theorem 5.6. *For any $0 \le p_b < 1$, consider the* (1+1) Swap-EA *on* BoundMax$_B$. *Let T be the random variable describing the number of steps until the algorithm finds the optimum. Then, there is a constant $0 < c < 1$ such that, for $B = cn$, $E[T]$ is in $\Omega(n^2)$.*

The proof follows a similar argument as in [Friedrich et al. 2020, Theorem 10]. We show that w.h.p. either $O(n^2)$ iterations pass without finding the optimum or the process samples an individual with a constant, positive Hamming distance to the optimum. Fixing the last missing positions then takes quadratic time.

We have hence shown that swap mutation is superior to balanced mutation for BoundMax$_B$ at least if $n - B = O(1)$ since then, the number of possible outcomes of swap mutation is small. However, this advantage vanishes if $n-B = \Theta(n)$ (Theorem 5.6) as this results in a significantly higher number of possible mutation outcomes.

6 OPTIMIZATION WITH CROSSOVER

To break the quadratic bound even in the case of $n - B = \Theta(n)$, we analyze the performance of crossover operators on BoundMax$_B$ in different scenarios. For this, we first examine a $(2+1)$ single-receiver island model as described in [Watson and Jansen 2007] with unbalanced and balanced crossover. We then extend our analysis to a $(3+1)$ single-receiver island model with majority vote crossover as described in [Friedrich et al. 2016].

In the $(\mu + 1)$ single-receiver island model, μ instances of the (1+1) EA are running independently on their own island using standard mutation. After each iteration, crossover is applied to two randomly selected residents of these μ islands. The offspring replaces the resident on the receiver island if it has higher fitness. In case of using majority vote crossover, three instead of two out of μ individuals are chosen. We only focus on the cases $\mu = 2$ and $\mu = 3$ respectively, since a higher number of individuals would not improve the asymptotic run time.

After having analyzed the island models, we consider the $(\mu+1)$ GA with balanced uniform crossover and Hamming distance maximization as diversity mechanism. In this setting, a population of μ individuals is maintained whereby in each iteration, one offspring is created. This happens with probability p_c by means of crossover on two randomly chosen individuals, and otherwise by means of standard mutation on one randomly selected individual. Out of the $\mu+1$ individuals at the end of each iteration, one individual with lowest fitness is removed according to a diversity mechanism/tie-breaking rule. In the case of Hamming distance maximization, this individual is chosen such that the sum of the pairwise Hamming distances of the remaining population is maximized. We refer to [Dang et al. 2016, Section 4.2]. Again, our analysis focuses on the case $\mu = 2$ as a larger population would not improve time complexity.

In the end, we analyze a variant of the above (2+1) GA that incorporates both balanced crossover and balanced mutation which we call the (2+1) Swap-GA. Here in each iteration, crossover is applied with probability p_c and otherwise it performs balanced mutation with probability p_b and unbalanced mutation else. A summary of our results is given in Table 1.

Analysis of $(2+1)$ island model with balanced and unbalanced crossover. We show that the run time is reduced from $O(n^2)$ for the (1+1) Swap-EA to $O(n^2/\log(n))$ in this setting. For this, we divide the optimization process into two stages. First we wait until the individuals on all non-receiver-islands have B 1-bits and only a certain number of heavy 0-bits. This step is captured in Lemma 6.1. Then, we analyze the time it takes the crossover to sample the optimum given that we have passed the first stage. We note that our fitness function ensures that we never go back to the first stage.

Lemma 6.1. *Consider the* (1+1) EA *using only standard mutations on* BoundMax$_B$ *with constraint B and starting on an individual with exactly B 1-bits. Let T be the random variable describing the number of steps until the algorithm finds a solution with at most $k \in \mathbb{N}^+$ heavy 0-bits. Then, $E[T]$ is in $O\left(\frac{n^2}{k}\right)$.*

Proof. We find an upper bound by employing the fitness level method with fitness levels $A_{\le k}, A_{k+1}, \ldots, A_B$, where A_i is the set of bit strings with i heavy 0-bits and $A_{\le k}$ contains all bit strings with at most k such bits. The initial solution is at worst in A_B and we are interested in the time until a solution in $A_{\le k}$ is first sampled.

Let p_i be the probability of leaving the fitness level A_i in an iteration. We show that $p_i \ge \frac{i^2}{en^2}$. To leave the level A_i, it suffices to flip one of the i heavy 0-bits as well as one of the i light 1-bits while leaving all other bits unchanged. The probability for this event is $\frac{i}{n} \cdot \frac{i}{n} \cdot (1 - \frac{1}{n})^{n-2} \ge \frac{i^2}{en^2}$.

We now apply the fitness level method to find the desired upper bound. Let T' be the random variable denoting the number of iterations to reach level $A_{\le k}$. We get

$$E[T'] \le \sum_{i=k+1}^{B} \frac{1}{p_i} \le \sum_{i=k+1}^{B} \frac{en^2}{i^2} = en^2 \sum_{i=k+1}^{B} \frac{1}{i^2}.$$

Then we use the integral to bound the sum. Because $\frac{1}{i^2}$ is strictly decreasing and we decrease the lower end by one, we get a valid upper bound of

$$E[T'] \le en^2 \int_{k+1-1}^{B} i^{-2}\, di = en^2 \left[-1 \cdot i^{-1}\right]_k^B = en^2 \left(-\frac{1}{B} + \frac{1}{k}\right).$$

This implies a total run time in $O\left(\frac{n^2}{k}\right)$. □

In order for crossover to be able to create the optimal solution, we need the individuals x and y on the first two islands to be free of so called *blocking bits*. We define a blocking bit as a position i such that $x_i = y_i = 0$ if $i \le B$ and $x_i = y_i = 1$ if $i > B$. Intuitively, a blocking bit is a position for which both individuals differ from the optimal solution. By counting the possibilities to remove a blocking bit by means of flipping two bits, one gets an estimate for the probability to remove blocking bits.

Lemma 6.2. *Let $x, y \in \{0,1\}^n$ be two individuals that both have exactly B 1-bits where $B = cn$ for some constant $0 < c < 1$. Assume further that both x and y have at most $i = o(n)$ heavy 0-bits and there is at least one blocking bit. The probability that a standard mutation on one of x and y removes a blocking bit without decreasing fitness is at least $\frac{\min(c,1-c)}{2en}$ for large enough n.*

Next, we analyze the probability that, assuming the parent individuals are free of blocking bits, balanced uniform crossover samples the optimal solution.

Lemma 6.3. *Let $x, y \in \{0,1\}^n$ be two individuals without blocking bits that both have exactly B 1-bits and Hamming distance h. For any $z \in \{0,1\}^n$, the sampling probability $p(z \mid x, y)$ of the balanced uniform crossover is either 0 or $\Omega(\sqrt{h}/2^h)$.*

Proof. Note that for each heavy bit position i on which x and y agree, we have $x_i = y_i = 1$ as otherwise there would be blocking bits. Let k be the number of heavy bits in which the individuals differ. Since both have B 1-bits, the number of differing light bits is also k. Each position in the light block for which one of the individuals is 1 is a differing position as otherwise there would be blocking bits. We thus have $h = 2k$ for the Hamming distance. Among those h positions, both x and y have k ones and k zeros.

The balanced uniform crossover chooses exactly k 1-bits at the differing positions with each of the $\binom{h}{k}$ possible outcomes being equally likely. We lower-bound the probability $1/\binom{2k}{k}$ via Stirling's approximation as $\sqrt{\pi k}/2^{2k}$. As $k = h/2$, we get that the crossover finds any specific solution with probability at least $\sqrt{\frac{\pi}{2}}\frac{\sqrt{h}}{2^h}$. □

With this, we derive the following upper bound for the $(2+1)$ single-receiver island model. It is independent of whether the balanced uniform crossover or the regular variant is used.

Theorem 6.4. *The expected optimization time for* BoundMax$_B$ *with $B = cn$ for constant $0 < c < 1$ on the (2+1) single-receiver island model using (balanced) uniform crossover is in $O\left(\frac{n^2}{\log(n)}\right)$.*

The idea is to first estimate the time until both individuals have B 1-bits and at most $\log_2(n/4)$ heavy 0-bits by using Lemmas 6.1 and 5.4. Then, drift analysis is employed on a potential function reflecting the number of blocking bits as well as the probability of the crossover sampling the optimum given that there are no more blocking bits, by employing Lemmas 6.2 and 6.3.

Analysis of $(3+1)$ island model with majority vote crossover. We further reduce the optimization time to $O\left(n\sqrt{n}\right)$ by instead using the deterministic majority vote crossover as introduced in [Friedrich et al. 2016]. This operator requires three parent individuals and sets each bit in the offspring to the value that the majority of the parents exhibits at the respective position.

Theorem 6.5. *The expected optimization time for* BoundMax$_B$ *with $B = cn$ for constant $0 < c < 1$ in the (3+1) single-receiver island model with majority vote crossover operator is in $O\left(n\sqrt{n}\right)$.*

Proof. We first wait until all EA islands have sampled individuals with B 1-bits. By Lemma 5.4, this takes parallel time $O(n\log(n))$.

Let T be the random variable describing the number of iterations until the optimum is sampled starting with three individuals with B 1-bits. By rewriting the definition of the expected value, we have

$$E[T] = \sum_{t=1}^{\infty} \Pr[T \geq t] \leq n\sqrt{n} \cdot \sum_{r=0}^{\infty} \Pr\left[T \geq rn\sqrt{n}\right]. \qquad (1)$$

Note that we switched from the number of iterations t to the number r of cycles of $n\sqrt{n}$ iterations. Further, $\Pr[T \geq 0] = 1$ and $\Pr[T \geq t+1] = \Pr[T \geq t] \cdot (1 - \Pr[T = t \mid T \geq t])$. Hence,

$$\Pr[T \geq t+1] = \prod_{i=1}^{t} (1 - \Pr[T = i \mid T \geq i]) \leq 1 - \Pr[T = t \mid T \geq t].$$

In order to estimate $\Pr[T = t \mid T \geq t]$, we define $\mathbf{S}$ as the event that majority vote crossover succeeds in creating the optimum in a single step, and $\mathbf{E}_k$ that there are at most k wrong bits in each of the two blocks of each individual. The occurrence of $\mathbf{S}$ given $\mathbf{E}_k$ holds means that necessarily no wrongly set bit is shared by two or more individuals. Suppose these errors inside each block are uniformly distributed and independent from the previous cycle, then

$$\begin{aligned}\Pr[\mathbf{S} \mid \mathbf{E}_k] &\geq \frac{\binom{B-k}{k}}{\binom{B}{k}} \frac{\binom{B-2k}{k}}{\binom{B}{k}} \cdot \frac{\binom{n-B-k}{k}}{\binom{n-B}{k}} \frac{\binom{n-B-2k}{k}}{\binom{n-B}{k}} \\ &\geq \left(1 - \frac{6k^2}{B}\right) \cdot \left(1 - \frac{6k^2}{n-B}\right) = 1 - \frac{6k^2 n + 36k^4}{(c-c^2)n^2},\end{aligned}$$

using the same estimate as in [Friedrich et al. 2016, Theorem 3.3].

By Lemma 6.1 we get that the expected number of heavy 0-bits after $r \cdot n\sqrt{n} - 1$ iterations is at most $c'\sqrt{n}/r$ for a constant c' and large enough n. As each individual has B 1-bits, the number of light 1-bits is the same. Further, we have $\Pr[T = t \mid T \geq t] \geq \Pr[\mathbf{S} \mid \mathbf{E}_k]$ with k being the maximum number of wrongly set bits in each block after t iterations. Hence, there is a constant c'' such that $\Pr\left[T = rn\sqrt{n} - 1 \mid T \geq rn\sqrt{n} - 1\right]$ is at least

$$1 - \frac{6(c'/r)^2 + 36(c'/r)^4}{(c-c^2)} \geq 1 - \frac{c''}{r^2}.$$

We get $\Pr\left[T \geq rn\sqrt{n}\right] \leq c''/r^2$ and by inserting this back in Equation (1) and once more applying the Basel problem, we get

$$E[T] \leq n\sqrt{n} \sum_{r=0}^{\infty} \frac{c''}{r^2} \leq n\sqrt{n} \cdot \frac{c''\pi^2}{6}.$$

It remains to argue that w.h.p. at the end of each cycle the wrongly set bits are uniformly distributed and independent from the previous cycle. This holds for the first cycle, as all weights inside both blocks are equal and standard mutation is order-unbiased. For all other cycles it holds w.h.p., as there are sufficiently many correctly set bits such that the probability of a wrongly set bit being swapped to another position by flipping a correctly and and incorrectly set bit is in $O(\frac{1}{n})$, yielding w.h.p. each wrongly set bit position is swapped inside its block at least twice each cycle. The desired uniformity and independence follows by an easy mixing argument. □

Analysis of (2+1) GA *with Hamming distance maximization.* We show that we can reduce the run time to $O(n\log(n))$ by employing the (2+1) GA and balanced crossover. This algorithm can reduce the number of wrong bits also by means of crossover, since the offspring produced by crossover is not put on an individual island, but can instead compete with the other individuals directly. The probability that balanced crossover makes progress is now constant when there is at least one non-blocking bit in each block.

Lemma 6.6. *Let $x, y \in \{0,1\}^n$ be two individuals with exactly B 1-bits and exactly i heavy 1-bits. Assume that there are $2a$ heavy bit positions and $2b$ light bit positions ($a, b \geq 1$) at which x and y differ. The probability that balanced crossover samples a solution with more than i heavy 1-bits is at least $\frac{1}{6}$ and thus in $\Omega(1)$.*

The bound is due to the probability of having exactly i heavy 1-bits being maximal at $a = b = 1$ with $2/3$ and the symmetry of the probabilities of having more or less than i heavy 1-bits.

In contrast to that, unbalanced crossover has a strictly worse probability of achieving improvement that is sub-constant and decreases with the number of positions at which two individuals differ. We use Stirling's approximation and exploit that even the probability of maintaining the number of 1-bits is less than constant.

Proposition 6.7. *Let $x, y \in \{0,1\}^n$ be two individuals with exactly B 1-bits and exactly i heavy 1-bits. Assume that there are $2a$ heavy bit positions and $2b$ light bit positions ($a, b \geq 1$) at which x and y differ. The probability that unbalanced crossover samples a solution with B many 1-bits and more than i heavy 1-bits is in $O\left(1/\sqrt{a+b}\right)$.*

This illustrates the advantage of balanced crossover versus its unbalanced counterpart. However, this advantage is only relevant if the number of positions at which x and y differ becomes large.

Exploiting the above statements, the expected optimization time of the (2+1) GA with Hamming distance maximization and balanced crossover improves to $O(n\log(n))$ for $n - B = \Theta(n)$.

Theorem 6.8. *For constant $0 < p_c < 1$, consider the (2+1) GA on BoundMax$_B$ with constraint $B = cn$ for constant c, using Hamming distance maximization for tie-breaking. The expected optimization time is in $O(n\log(n))$.*

Proof. We split the proof in three stages. We first analyze the time until both individuals have exactly B 1-bits, then the time until both individuals have at most $\frac{\min(c,1-c)n}{2}$ many heavy 1-bits, and lastly the time until the optimal solution is created. In the following, we denote the two individuals of the GA with x and y.

For the first part, by employing Lemma 5.4, the expected number of iterations until both individuals have exactly B 1-bits is in $O(n\log(n))$. For the second part, we get from Lemma 6.1 that the time until the (1+1) EA creates an individual with at most $k := \frac{\min(c,1-c)n}{2}$ heavy 1-bits is $O(n)$. Since the number of heavy 0-bits of the best individual does not decrease if both individuals have B 1-bits, and any worsening by crossover only contributes a constant factor to the run time, the expected time until the first individual has at most k heavy 1-bits is in $O(n)$. The expected time until this holds for both individuals is at most twice this.

For the third part of the proof, we employ the fitness level method [Wegener 2002] and define the fitness levels $A_{k,0}, A_{k,1}, A_{k,2}, A_{k,3}, A_{k-1,0}, A_{k-1,1}, A_{k-1,2}, A_{k-1,3} \ldots, A_{0,0}$ where the algorithm is in level $A_{i,j}$ for $j \in \{0,1,2,3\}$ if the individual with highest fitness has exactly i heavy 0-bits. Furthermore, each fitness level is partitioned in four sub levels with the following interpretations:

- $A_{i,0}$: exactly one individual has i heavy 0-bits.
- $A_{i,1}$: both individuals have i heavy 0-bits and are duplicates.
- $A_{i,2}$: both individuals have i heavy 0-bits and differ in at least one position in the first or second block but not in both.
- $A_{i,3}$: both individuals have i heavy 0-bits and differ in at least one position in the first and in the second block.

We analyze the probability $p_{i,j}$ of advancing from level $A_{i,j}$. Following the idea of the fitness-level method, the expected run time is then upper bounded by $E[T] \leq \sum_{i=1}^{B}\sum_{j=0}^{3}\frac{1}{p_{i,j}}$.

Note that, in contrast to usual application of the fitness level method, it is possible to fall back from level $A_{i,3}$ to $A_{i,2}$. However, the probability that this happens before we transition to levels above $A_{i,3}$ is $o(1)$, so a simple restart argument shows that this has only a lower order impact on the run time which we will ignore.

For probability $p_{i,0}$, we consider the event that the fittest individual gets duplicated by means of a standard mutation that does not flip any bit. This happens with a probability of at least $\frac{(1-p_c)}{2}\left(1-\frac{1}{n}\right)^n \geq \frac{1-p_c}{4e} = \Omega(1)$ which shows that $p_{i,0} = \Omega(1)$.

For $p_{i,1}$ and $p_{i,2}$, we note that in the levels $A_{i,1}$ and $A_{i,2}$, the first or the second block of x and y are identical. We consider the event of advancing to a following level by swapping a one and a zero within one of the identical blocks by means of the mutation operator. Since such a mutation does not change the fitness of the offspring but increases its Hamming distance to the other individuals by 2, it is always accepted. From the fact that there are $i \leq \min(c, 1-c)n/2$ heavy 0-bits and light 1-bits, respectively, in both x and y, we get that the probability of the described event is at least

$$\frac{i(\min(c,1-c)n - i)}{n^2}\left(1-\frac{1}{n}\right)^{n-2} \geq \frac{i\min(c,1-c)n/2}{en^2}.$$

Hence, both $p_{i,1}$ and $p_{i,2}$ are in $\Omega(i/n)$. For $p_{i,3}$, we consider the event that crossover creates a solution with exactly B 1-bits and more than i heavy 1-bits. Since they differ in at least two positions in each block, we get from Lemma 6.6 that this probability is $\geq 1/6$.

We can thus estimate the expected run time as

$$E[T] \leq \sum_{i=1}^{B}\sum_{j=0}^{3}\frac{1}{p_{i,j}} \leq \sum_{i=1}^{B}\left(\alpha + \beta\frac{n}{i} + \gamma\right)$$

where α, β, γ are positive constants. Hence, for some constant $\delta > 0$

$$E[T] \leq \sum_{i=1}^{B}\left(\alpha + \beta\frac{n}{i} + \gamma\right) \leq \delta n\log(B) = O(n\log(n)).$$

This bounds the expected total run time by $O(n\log(n))$. □

If B is only a constant away from n, we can show that none of the analyzed crossover scenarios brings any improvements over just a (1+1) EA with standard mutation. There is a constant probability of having a blocking light bit once the constraint is met, and the probability to remove it is in $O(1/n^2)$ for small d.

Theorem 6.9. *For the (2+1) GA using Hamming distance maximization as well as for the island models employing balanced uniform or majority vote crossover, there is a constant d such that the expected optimization time when optimizing BoundMax$_B$ with constraint $B = n - d$ is in $\Theta\left(n^2\right)$.*

Analysis of the (2+1) Swap-GA. For the purpose of achieving good run times both in case of $n - B = O(1)$ as well as $n - B = O(n)$, we combine the benefits of (1+1) Swap-EA and (2+1) GA as explained at the beginning of the section. One can quickly see that neither the swap mutation nor the crossover hinder one another, so both asymptotically work as least as well as they did before.

Theorem 6.10. *For constant $0 < p_b, p_c < 1$, consider the (2+1) Swap-GA on BoundMax$_B$, using Hamming distance maximization for tie-breaking. The expected optimization time is in $O(n\log(n))$ for both $B = cn$ and $B = n - d$ with constant c, d.*

REFERENCES

Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. 2020. Fast Mutation in Crossover-Based Algorithms. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO '20)*. 1268–1276. https://doi.org/10.1145/3377930.3390172

Jiah-Shing Chen and Jia-Leh Hou. 2006. A Combination Genetic Algorithm with Applications on Portfolio Optimization. In *Advances in Applied Artificial Intelligence (Lecture Notes in Computer Science)*, Moonis Ali and Richard Dapoigny (Eds.). Springer, 197–206. https://doi.org/10.1007/11779568_23

Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2018. Escaping Local Optima Using Crossover With Emergent Diversity. *IEEE Transaction on Evolutionary Computation* 22 (2018), 484–497. https://doi.org/10.1109/TEVC.2017.2724201

Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2016. Escaping Local Optima with Diversity Mechanisms and Crossover. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference (GECCO '16)*. 645–652. https://doi.org/10.1145/2908812.2908956

Benjamin Doerr, Philipp Fischbeck, Clemens Frahnow, Tobias Friedrich, Timo Kötzing, and Martin Schirneck. 2019. Island Models Meet Rumor Spreading. *Algorithmica* 81 (2019), 886–915. https://doi.org/10.1007/s00453-018-0445-2

Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Samadhi Nallaperuma, Frank Neumann, and Martin Schirneck. 2016. Fast Building Block Assembly by Majority Vote Crossover. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference (GECCO '16)*. 661–668. https://doi.org/10.1145/2908812.2908884

Tobias Friedrich, Timo Kötzing, J. A. Gregor Lagodzinski, Frank Neumann, and Martin Schirneck. 2020. Analysis of the (1+1) EA on Subclasses of Linear Functions under Uniform and Linear Constraints. *Theoretical Computer Science* 832 (2020), 3–19. https://doi.org/10.1016/j.tcs.2018.04.051

Thomas Jansen. 2015. On the Black-Box Complexity of Example Functions: The Real Jump Function. In *Proceedings of the 13th Conference on Foundations of Genetic Algorithms (FOGA '17)*. 16–24. https://doi.org/10.1145/2725494.2725507

Thomas Jansen and Dirk Sudholt. 2010. Analysis of an asymmetric mutation operator. *Evolutionary Computation* 18, 1 (Mar 2010), 1–26. https://doi.org/10.1162/evco.2010.18.1.18101

Thomas Jansen and Ingo Wegener. 2002. The Analysis of Evolutionary Algorithms - A Proof That Crossover Really Can Help. *Algorithmica* 34 (2002), 47–66. https://doi.org/10.1007/s00453-002-0940-2

Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A Review on Genetic Algorithm: Past, Present, and Future. *Multimedia Tools and Applications* 80 (2021), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

Timo Kötzing, Dirk Sudholt, and Madeleine Theile. 2011. How Crossover Helps in Pseudo-Boolean Optimization. In *Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO '11)*. 989–996. https://doi.org/10.1145/2001576.2001711

Per Kristian Lehre and Carsten Witt. 2012. Black-Box Search by Unbiased Variation. *Algorithmica* 64, 4 (2012), 623–642. https://doi.org/10.1007/s00453-012-9616-8

Luca Manzoni, Luca Mariot, and Eva Tuba. 2020. Balanced Crossover Operators in Genetic Algorithms. *Swarm and Evolutionary Computation* 54 (2020), 100646. https://doi.org/10.1016/j.swevo.2020.100646

Thorsten Meinl and Michael R. Berthold. 2009. Crossover Operators for Multiobjective k-Subset Selection. In *Proceedings of the 2009 Genetic and Evolutionary Computation Conference (GECCO '09)*. 1809–1810. https://doi.org/10.1145/1569901.1570173

Alberto Moraglio and Riccardo Poli. 2004. Topological Interpretation of Crossover. In *Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO '04)*, Kalyanmoy Deb (Ed.). Springer, 1377–1388. https://doi.org/10.1007/978-3-540-24854-5_131

Frank Neumann, Pietro Simone Oliveto, Günter Rudolph, and Dirk Sudholt. 2011. On the effectiveness of crossover for migration in parallel evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO '11)*. Association for Computing Machinery, 1587–1594. https://doi.org/10.1145/2001576.2001790

Frank Neumann and Ingo Wegener. 2007. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science* 378, 1 (Jun 2007), 32–40. https://doi.org/10.1016/j.tcs.2006.11.002

Nicholas J. Radcliffe. 1994. The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence* 10, 4 (Dec 1994), 339–384. https://doi.org/10.1007/BF01531276

Jonathan E. Rowe and Michael D. Vose. 2011. Unbiased Black Box Search Algorithms. In *Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO '11)*. 2035–2042. https://doi.org/10.1145/2001576.2001850

Dirk Sudholt. 2017. How Crossover Speeds up Building Block Assembly in Genetic Algorithms. *Evolutionary Computation* 25 (2017), 237–274. https://doi.org/10.1162/EVCO_a_00171

Richard A. Watson and Thomas Jansen. 2007. A building-block royal road where crossover is provably essential. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO '07)*. Association for Computing Machinery, 1452–1459. https://doi.org/10.1145/1276958.1277224

Ingo Wegener. 2002. Methods for the Analysis of Evolutionary Algorithms on Pseudo-Boolean Functions. In *Evolutionary Optimization*, Ruhul Sarker, Masoud Mohammadian, and Xin Yao (Eds.). Springer, 349–369. https://doi.org/10.1007/0-306-48041-7_14

L. Darrell Whitley, Francisco Chicano, and Brian W. Goldman. 2016. Gray Box Optimization for Mk Landscapes (NK Landscapes and MAX-kSAT). *Evolutionary Computation* 24 (2016), 491–519. https://doi.org/10.1162/EVCO_a_00184