Memetic Algorithms Outperform Evolutionary Algorithms in Multimodal Optimisation[⋆]

Phan Trung Hai Nguyen[a], Dirk Sudholt[b,*]

[a]*School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K.*
[b]*Department of Computer Science, University of Sheffield, Sheffield S1 4DP, U.K.*

Abstract

Memetic algorithms integrate local search into an evolutionary algorithm to combine the advantages of rapid exploitation and global optimisation. We provide a rigorous runtime analysis of memetic algorithms on the Hurdle problem, a landscape class of tuneable difficulty with a "big valley structure", a characteristic feature of many hard combinatorial optimisation problems. A parameter called hurdle width describes the length of fitness valleys that need to be overcome. We show that the expected runtime of plain evolutionary algorithms like the (1+1) EA increases steeply with the hurdle width, yielding superpolynomial times to find the optimum, whereas a simple memetic algorithm, (1+1) MA, only needs polynomial expected time. Surprisingly, while increasing the hurdle width makes the problem harder for evolutionary algorithms, it becomes easier for memetic algorithms.

We further give the first rigorous proof that crossover can decrease the expected runtime in memetic algorithms. A (2+1) MA using mutation, crossover and local search outperforms any other combination of these operators. Our results demonstrate the power of memetic algorithms for problems with big valley structures and the benefits of hybridising multiple search operators.

Keywords: Search heuristics, hybridisation, evolutionary algorithms, black-box optimisation, memetic algorithms, time complexity

1. Introduction

1.1. Motivation

Memetic Algorithms (MAs) are hybrid evolutionary search methods that incorporate local search into the search process of an evolutionary algorithm. They are known by various other names, including evolutionary local search [2], genetic local search [3, 4] or global-local search hybrids [5], to combine the exploration capabilities of evolutionary algorithms with the rapid exploitation provided by local search. There are many examples where this strategy has

[⋆]This paper extends results presented at the 2018 Genetic and Evolutionary Computation Conference (GECCO 2018) [1]. Part of the work was done when the first author was a student at the University of Sheffield.

[*]Corresponding author

Email address: `d.sudholt@sheffield.ac.uk` (Dirk Sudholt) proven effective, including many hard problems like the Quadratic Assignment Problem [6], arc routing problems [7], job shop scheduling problems [8] and generating test inputs in software testing [9], just to name a few; further references can be found in books and surveys [10, 11, 12, 13].

The motivation for using local search is that it can quickly find improvements of new offspring, enabling evolution to rapidly find high-fitness individuals. Local search can include problem-specific aspects to boost exploitation. This is even possible for complex problems where it is hard to design a global search strategy.

The search dynamics of memetic algorithms can differ significantly from plain evolutionary algorithms. When new offspring are created, the selection is only performed after a local search has refined the offspring to higher fitness. This is particularly beneficial for offspring of poor fitness that is located in the basin of attraction of a local optimum with a high fitness as in a plain evolutionary algorithm such an offspring would be quickly removed during selection. For constrained problems with penalties for violated constraints, local search can effectively act as a repair mechanism [13].

One particular subclass of memetic algorithms is called *iterated local search* [14]. Iterated local search algorithms maintain a population (or a single search point) and apply mutation followed by local search. Local search is applied in every iteration, usually until a local optimum has been reached. This means that local search always produces a local optimum, the population always consists of local optima and evolution thus acts on the sub-space of local optima. The hope is that mutation can help an MA to leave its current local optimum and to reach the basin of attraction of a better one.

A major challenge when designing MAs is to find an effective combination of different search operators like mutation, crossover and selection as well as an effective local search. The performance of MAs is often determined by the interplay of different operators. The outcome and effectiveness of local search applications crucially depend on the result of variation operators like mutation or crossover followed by mutation. The output of local search, after selection, yields the next generation to which further variation operators are applied. This yields intricate search dynamics that are hard to predict as the execution of different search operators, including local search, is intertwined in non-trivial ways.

It is therefore not well understood when memetic algorithms perform well and why. Most work in this area is empirical, and the reasons for the success (or failure) of memetic algorithms are often not clear. As put in [15], *"While memetic algorithms have achieved remarkable success in a wide range of real-world applications, the key to successful exploration-exploitation synergies still remains obscure."* The field is lacking a solid understanding of how memetic algorithms work and how to design effective memetic algorithms in an informed fashion.

We make a significant step towards establishing a rigorous theoretical foundation of memetic algorithms by analysing the performance of simple memetic algorithms on a class of multimodal problems that resemble a "big valley structure" found in many combinatorial problems [16, 17]. Our main contribution is a runtime analysis of memetic algorithms showing that—and how—these hybrids can drastically outperform their algorithmic components.

We further address an issue highlighted in a survey on hot topics in evolutionary computation at AAAI 2017 [18]: "*The role of crossover in evolutionary* *computation is still a major open problem in the theory of evolutionary algorithms.*" We make a step forward towards solving this long-standing open problem by providing the first runtime analysis of a crossover-based memetic algorithm. We give a rigorous proof that crossover (followed by mutation) can significantly improve performance compared to memetic algorithms using only mutation as variation operator.

More specifically, we consider a class of problems introduced by Prügel-Bennett [19] as example problems where genetic algorithms using crossover perform better than hill climbers. The Hurdle problem class (formally defined in Section 3) is a function of *unitation*, that only depends on the number of ones, with an underlying gradient leading towards the global optimum and several "hurdles" that need to be overcome. These hurdles consist of a local optimum and a fitness valley that has to be traversed to reach the next local optimum. The width of these fitness valleys is determined by a parameter w called the *hurdle width*. For simple EAs like the (1+1) EA, a larger hurdle width makes the problem harder. This effect was analysed in [19] with non-rigorous arguments based on simplifying assumptions that led to approximations of the expected time for finding the unique global optimum.

1.2. Our Contributions

We provide a rigorous analysis of the expected *runtime* (synonymously, *optimisation time*), which is the number of function evaluations[1] needed until a global optimum is seen for the first time.

For the (1+1) EA we give a tight bound of $\Theta(n^w)$ for the expected runtime, confirming that the performance degrades very rapidly with increasing hurdle width w. For hurdle widths growing with n, i.e., $w = \omega(1)$, this expected runtime is superpolynomial. The latter still holds when increasing the mutation rate from the default value $1/n$ to the optimal value w/n.

We also show that the local searches on their own perform very poorly. Even when appropriate restart strategies are being used, they require exponential runtime.

In contrast, we show that MAs perform very effectively on this problem class, owing to the combination of evolutionary operators and local search. We study a simple iterated local search algorithm called (1+1) MA that evolves a single search point by using only mutation and local search. We consider two different local searches: *First-Improvement Local Search* (FILS) searches all Hamming neighbours of the current search point and moves to the first improvement it finds in this way. *Best-Improvement Local Search* (BILS) [20] on the other hand first evaluates the whole neighbourhood and then selects a best improvement found. Both local searches stop when no improvement can be found. We show that the (1+1) MA with BILS takes expected time $\Theta(n^2 + n^3/w^2)$, and the (1+1) MA with FILS takes expected time $\Theta(n^3/w^2)$ to find the optimum on Hurdle. These times are polynomial for all choices of the hurdle width.

[1]Considering function evaluations is motivated by the fact that these are often the most expensive operations, whereas other operators can usually be executed very quickly. For steady-state algorithms like the simple (1+1) EA, the number of function evaluations equals the number of iterations. For MAs, we further need to account for the number of function evaluations made during local search.

Another major contribution of this work lies in presenting the first rigorous proof that crossover is beneficial in memetic algorithms. To this end, we consider a simple (2+1) MA that uses mutation, uniform crossover and local search. The (2+1) MA with FILS takes an expected runtime of $\mathcal{O}\big(n^2 \log(n/w)/w\big)$, while the (2+1) MA with BILS requires time $\mathcal{O}\big(n^2 + n^2 \log(n/w)/w\big)$ in expectation on Hurdle. These results show that crossover provides a substantial advantage over the (1+1) EA and (1+1) MA, leading to speedups of $\Omega(n/\log n)$ for $w = \mathcal{O}(1)$.

Note that the terms n^3/w^2 and $n^2 \log(n/w)/w$ decrease with the hurdle width w; hence, the surprising conclusion is that larger hurdle widths make the problem much harder for EAs, while making the problem easier for MAs. As our analysis will show, the reason behind this effect is that for MAs it is sufficient to flip two 0-bits to escape from the basin of attraction of a local optimum and to reach the basin of attraction of a better local optimum. This way, MAs can efficiently progress from one local optimum to the next. For the most difficult local optimum, the number of 0-bits equals the hurdle width w. A larger hurdle width provides more opportunities for flipping two 0-bits and increases the probability of such a jump being made. This explains why for MAs, a larger hurdle width makes the problem easier.

Our findings are particularly significant when regarded in the light of big valley structures, an important characteristic of many hard problems from combinatorial optimisation [16, 17], where "many local optima may exist, but they are easy to escape and the gradient, when viewed at a coarse level, leads to the global optimum" [21]. The Hurdle problem (albeit having been defined for a very different purpose [19]) is a perfect and very illustrative example of a big valley landscape. By studying the (1+1) MA on the Hurdle problem class, we hope to gain insights into how MAs perform on big valley structures, which in turn may explain why state-of-the-art MAs perform well on problems with big valley structures [22, 23].

1.3. Related Work

There are other examples of functions where MAs were theoretically proven to perform well (see Sudholt [13] for a more extensive survey). In [24], examples of constructed functions were given where the (1+1) EA, the (1+1) MA, and Randomised Local Search (RLS) can mutually outperform each other. The paper [25] investigates the impact of the *local search depth*, which is often used to limit the number of iterations local search is run for. The author gives a class of example functions where only specific choices for the local search depth are effective, and other parameter settings, including plain EAs without local search, fail badly. Similar results were obtained for the choice of the *local search frequency*, that is, how often local search is run [26].

Sudholt [27] showed for certain instances of classical problems from combinatorial optimisation, MinCut, Knapsack and MaxSat, that MAs with a different kind of local search, *variable-depth local search*, can efficiently cross huge fitness valleys that are nearly impossible to cross with EAs. Exponential times are reduced to polynomial expected times. Witt [28] analysed the performance of an MA, iterated local search, for the Vertex Cover problem. He showed that the algorithm can find optimal vertex covers on sparse random graphs in polynomial time, with probability $1 - \Omega(1)$. Sudholt and Zarges [29] investigated the use of MAs for the graph colouring problem and showed that

MAs can efficiently colour bipartite graphs with 2 colours and colour planar graphs with maximum degree up to 6 with up to 5 colours. Wei and Dinneen analysed MAs for solving the Clique problem, investigating the choice of the fitness function [30] as well as the choice of the local search operator [31]. They gave families of graphs where one choice of fitness function outperforms another and where the (1+1) MA with first-improvement local search (FILS) and the (1+1) MA with best-improvement local search (BILS) outperform one another.

Gießen [32] presented another example function class based on a discretised version of the well-known Rastrigin function. He designed an MA using a new local search method called *opportunistic local search*, where the search direction switches between minimisation and maximisation whenever a local optimum is reached. This function also resembles a big valley structure in two dimensions as the bit string is mapped onto a two-dimensional space.

Memetic algorithms also include *adaptive memetic algorithms* that combine different operators ("memes") and learn which one performs best throughout the algorithm run; such algorithms are also known as *hyperheuristics* [33]. Alanazi and Lehre [34] demonstrated the usefulness of hyperheuristics for the well-known unimodal LeadingOnes function. Lissovoi *et al.* [35] showed that selection hyperheuristics that can switch between elitist and non-elitist heuristics can be very beneficial on multimodal optimisation problems such as the function class Cliff. Cliff functions are related to the Hurdle problem as both feature local optima on an underlying gradient towards the optimum. However, Cliff only has local optima in one location whereas on Hurdle a sequence of many local optima has to be overcome. The latter authors [36] as well as Doerr *et al.* [37] presented novel, provably efficient hyperheuristic algorithms for LeadingOnes. The difference to MAs is that while hyperheuristics typically apply one operator, while learning which operator performs best, MAs apply different operators, variation and local search, in sequence. The interplay of variation and local search is a major challenge when analysing MAs.

In terms of crossover, to our knowledge, there are no rigorous runtime analyses of MAs using crossover. Despite a long-standing debate on the usefulness of crossover in evolutionary computation, only a few papers were able to rigorously prove that crossover is beneficial in evolutionary computation.

Jansen and Wegener [38] gave the first examples where crossover can reduce a superpolynomial expected time towards a polynomial time for the function Jump. These results were later refined in [39, 40]. Dang *et al.* [41] showed that further improvements can be obtained by using explicit diversity mechanisms. Jansen and Wegener [42] also defined so-called Real Royal Road functions where crossover makes a difference between polynomial and exponential times. While these functions require a linear population size, Storch and Wegener [43] defined examples where constant population size is sufficient to achieve similar performance gaps. Fischer and Wegener [44] showed that crossover can improve performance on a simple vertex colouring problem inspired by the Ising model in physics, on cycle graphs. Sudholt [45] proved an exponential performance gap for the Ising model on binary trees. Doerr *et al.* [46, 47] proved that crossover can speed up finding all-pairs shortest paths. Sudholt [48, 49] and Doerr *et al.* [50] showed that crossover can speed up hill climbing on the simple function OneMax. The former results were later improved by Corus and Oliveto [51] as well as Carvalho Pinto and Doerr [52]. Lengler [53] showed that crossover makes evolutionary algorithms more robust: on monotone functions, a (μ+1) GA with crossover can deal with arbitrary mutation strengths, whereas evolutionary algorithms without crossover fail badly [53, 54]. Moreover, crossover has also been proven to be useful in the context of island models [55] and ant colony optimisers [56].

1.4. Outline

The paper is structured as follows. Section 2 defines the algorithms studied in this work, including (1+1) EA, (1+1) MA, (2+1) MA with crossover as well as the two local search schemes, i.e., FILS and BILS. Section 3 formally describes the class of HURDLE problems and some of their important properties. Our analysis begins in Section 4, where we answer the question of why hybridisation is needed by investigating the expected runtimes of local search algorithms and the (1+1) EA on the HURDLE problems. The efficiency of the (1+1) MA over alternative search methods is then shown in Section 5. In Section 6, we study the effect of crossover in the (2+1) MA and show that it provides a substantial advantage over the mutation-only (1+1) MA. We then discuss the benefits of different search operators and their combinations in Section 7, followed by an empirical study in Section 8 in order to complement the theoretical analyses in previous sections.

2. Preliminaries

Let $\mathcal{X} := \{0,1\}^n$ denote the finite binary search space of dimension n (also called the problem instance size). Each bitstring $x = (x_1, x_2, \ldots, x_n) \in \mathcal{X}$ is called an *individual* (or search point), where $x_i \in \{0,1\}$ for each $i \in \{1, 2, \ldots, n\}$. A *population* is a multiset of individuals. We aim at maximising an *objective function* $f : \mathcal{X} \to \mathbb{R}$, also called *fitness function*. Note that we can easily deal with minimisation problems as well by maximising the function $-f$. Moreover, the ceiling and floor functions are denoted as $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$, respectively. The natural logarithm is denoted as $\ln(\cdot)$, while $\log(\cdot)$ represents the logarithm with base 2.

To focus on the main differences between EAs and MAs, and to facilitate a rigorous theoretical analysis, we consider simple bare-bones algorithms from these two paradigms.

2.1. (1+1) Evolutionary Algorithm

The (1+1) EA, defined in Algorithm 1, is the simplest EA as it operates on a single individual. The algorithm follows an iterative process and starts with an initial individual that is sampled uniformly at random. In each iteration, bits in the bitstring are flipped independently with probability p_m (also called *mutation rate*), and the newly generated offspring replaces the current search point if its fitness is not less than that of the current search point. The default mutation rate is $p_m = 1/n$, but we will also consider a larger mutation rate of $p_m = w/n$.

As a common practice in theoretical investigations, we do not specify a stopping criterion. Instead, we consider the algorithms as infinite processes as we are only interested in analysing the time until the global optimum is found.

Algorithm 1: (1+1) EA

```
1 sample an initial individual $x$ uniformly at random
2 repeat
3     create $y$ by flipping each bit in $x$ independently with probability $p_m$
4     if $f(y) \geq f(x)$ then update $x$ to $y$
5 until *stopping condition is fulfilled.*
```

2.2. Memetic Algorithms

The (1+1) MA, defined in Algorithm 2 [31], is considered the simplest MA, as it works on a single individual and creates an offspring in each iteration by flipping each bit independently with a mutation rate of $1/n$. Unlike the (1+1) EA, the newly generated offspring is refined further using a local search algorithm. We note that any local search algorithm can fit into this framework. Although the (1+1) MA looks quite simple, it captures the fundamental working principles of MAs. Analysing it can reveal insights into how general MAs operate and when and why they are effective.

Algorithm 2: (1+1) MA

```
1 sample an individual $x$ uniformly at random from $\mathcal{X}$
2 repeat
3     create $y$ by flipping each bit in $x$ independently with probability $1/n$
4     create $z$ by calling a local search algorithm on $y$
5     if $f(z) \geq f(x)$ then update $x$ to $z$
6 until *stopping condition is fulfilled.*
```

We note that the (1+1) MA uses mutation as the only genetic operator. We are also interested in crossover and thus introduce the (2+1) MA, defined in Algorithm 3. The algorithm operates on a population of two individuals[2] and uses both mutation and *uniform crossover* to produce an offspring. Recall that in a uniform crossover, each bit from the offspring's bitstring is independently chosen from either parent with equal probability [57]. Then, the two fittest individuals among the parents and the offspring are selected to form the next population. In case of ties, by which we mean that parents and offspring all having the same fitness, the offspring is rejected if it is a duplicate of a parent (if not, ties are broken arbitrarily). This allows different individuals of equal fitness to survive.

2.3. Local Search Algorithms

Local search algorithms employ a *move* operator in order to examine neighbours of the current search point on the search space. Here, the move operator flips a single bit in the bit string (i.e. one-bit mutation). By doing that, it can visit all Hamming neighbours in order to guide the search to a local optimum.

[2]A population of size two is chosen as this is the smallest population size enabling the use of crossover. For our runtime bounds, there is no advantage in using a larger population.

Algorithm 3: (2+1) MA with crossover

```
1 initialise population $\mathcal{P}$ of size two uniformly at random
2 repeat
3     select $x_1$ and $x_2$ independently and uniformly at random from $\mathcal{P}$
4     create $y$ by a crossover of $x_1$ and $x_2$
5     create $z$ by flipping each bit in $y$ independently with probability $1/n$
6     refine $z$ using a local search algorithm
7     let $\mathcal{P}$ contain the two fittest individuals from $\mathcal{P} \cup \{z\}$; if all have the
        same fitness, reject $z$ if it is identical to at least one of its parents
8 until stopping condition is fulfilled.
```

We consider two local search algorithms in the context of the (1+1) MA. Both are common practice and have also been analysed in [31].

Algorithm 4: FILS

```
   input : an individual $x \in \mathcal{X}$
1  $t \leftarrow 0$
2  while $t \leq \delta$ do
3      create a random permutation PER of the set $[n]$
4      set FLAG to 0
5      for $i = 1, 2, \ldots, n$ do
6          create $y$ by flipping bit PER$[i]$ in $x$
7          if $f(y) > f(x)$ then
8              update $x$ to $y$
9              set FLAG to 1
10             increment $t$
11             if $t = \delta$ return $x$
12     if FLAG $= 0$ then return $x$
13 return $x$
```

2.3.1. First-Improvement Local Search

The local search algorithm, defined in Algorithm 4 [31], takes advantage of the first improvement it obtains while searching the Hamming neighbourhood. The algorithm makes at most δ moves, a common strategy in memetic algorithms; the parameter δ is called *local search depth*. The algorithm flips bits according to a random permutation PER of length n. This is done to make sure all bits are treated equally and that the search does not depend on the bit positions. The behaviour of the local search only depends on the number of 0-bits and it is irrelevant *which* bits are set to 0. The fitness function then evaluates new offspring, and the current search point is replaced by the first neighbour discovered with better fitness. The algorithm terminates after either δ moves have been made or after visiting n neighbours of the current search point without finding an improvement.

Algorithm 5: BILS

```
    input: an individual $x \in \mathcal{X}$
 1  for $\delta$ iterations do
 2      CurBestInds $\leftarrow \emptyset$; CurBestFit $\leftarrow f(x)$
 3      for $i = 1, 2, \ldots, n$ do
 4          create $y$ by flipping bit $i$ in $x$
 5          if $f(y) >$ CurBestFit then
 6              CurBestInds $\leftarrow \{y\}$; CurBestFit $\leftarrow f(y)$
 7          else if $f(y) =$ CurBestFit then
 8              CurBestInds $\leftarrow$ CurBestInds $\cup \{y\}$
 9      if CurBestInds $= \emptyset$ then return $x$; otherwise, select $x$ uniformly at
        random from the set CurBestInds
10  return $x$
```

2.3.2. Best-Improvement Local Search

The local search algorithm, defined in Algorithm 5 [31], searches all neighbours (with a Hamming distance of one). The algorithm runs for at most δ iterations, and in each iteration, a neighbour with the largest fitness improvement replaces the current search point. To keep track of the progress so far, it records the best neighbour(s) and their fitness in CurBestInds and CurBestFit, respectively. Whenever the algorithm finds a neighbour with improved fitness compared to CurBestFit, it updates the two variables. At the end of an iteration, if there is more than one neighbour with equal fitness improvements, ties are broken uniformly at random.

3. Class of Hurdle Problems

The Hurdle function class was introduced back in 2004 by Prügel-Bennett [19] as an example class where genetic algorithms with crossover outperform hill climbers.

We start by providing a formal definition and then discuss basic properties of the function that will be used in the subsequent analyses.

The objective of the Hurdle problem is to maximise an objective function $f \colon \mathcal{X} \to \mathbb{R}$ that is formally defined as follows [19].

$$f(x) = -\left\lceil \frac{z(x)}{w} \right\rceil - \frac{\mathrm{rem}(z(x), w)}{w},$$

where $z(x)$ denotes the number of zeros in the bitstring x and $w \in \{2, 3, \ldots, n\}$, called the hurdle width, is the only parameter of the problems. We note that $w = w(n)$ may be a function of n, and $\mathrm{rem}(z(x), w)$ is the remainder of $z(x)$ divided by w.

An initial observation is that the global optimum for the Hurdle problem is the all-ones bitstring, i.e., 1^n, with the maximum fitness of 0 since for every $x \in \mathcal{X}$, we know that $z(x) \geq 0$ and $\mathrm{rem}(z(x), w) \geq 0$, so $f(x) \leq 0$. The equality holds if and only if $z(x) = 0$ and $\mathrm{rem}(z(x), w) = 0$, which correspond to $x = 1^n$.

We also note that $z(x)$ is the Hamming distance $H(x, 1^n)$ between the current search point x and the all-ones optimum. Fig. 1 illustrates the fitness landscapes

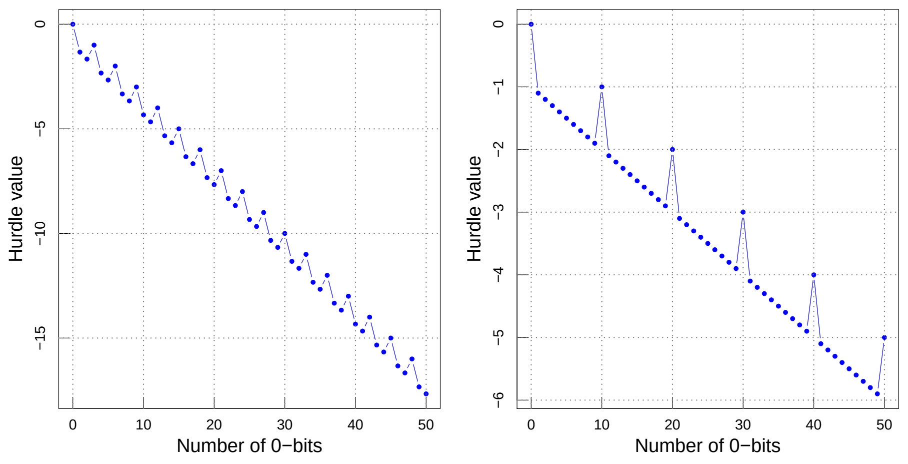

Figure 1: Fitness landscapes for a problem of size $n = 50$ with hurdle width $w = 3$ (left) and $w = 10$ (right).

close to the global optimum, which coincides with the origin where we get $z(1^n) = 0$ and $f(1^n) = 0$.

The following lemma shows that from any local optimum, the only search points of better fitness are those that have at least w zeros less.

Lemma 1. *Given a* Hurdle *problem with hurdle width w and a local optimum $x \neq 1^n$ as the current search point, the set of search points with fitness larger than $f(x)$ is equal to the set $\{x' \in \mathcal{X} : z(x') \leq z(x) - w\}$.*

Proof. Assume that we are currently at a local optimum $x \neq 1^n$ that contains $z(x) = kw$ zeros, where $k \in \mathbb{N}\backslash\{0\}$. Then $f(x) = -\lceil z(x)/w \rceil - \text{rem}(z(x), w)/w = -\lceil k \rceil - 0/w = -k$.

We now consider a search point x_m with $z(x_m) = kw - m < z(x)$, where $m > 0$. We make a case distinction via m: if $m \geq w$, then $f(x_m) = -\lceil kw - m \rceil - \text{rem}(kw - m, w)/w \geq -(k-1) - \text{rem}(kw - m, w)/w > -k$ as $\text{rem}(kw - m, w)/w < 1$. Otherwise, $\lceil kw - m \rceil = k$ and $\text{rem}(kw - m, w) = w - m > 0$, yielding $f(x_m) = -k - \text{rem}(kw - m, w)/w < f(x)$.

Therefore, when $m \geq w$, we obtain $f(x_m) > f(x)$; otherwise, $f(x_m) < f(x)$. This result implies that search points that have a better fitness than that of a current local optimum $x \neq 1^n$ are those in the set $\{x' \in \mathcal{X} : z(x') \leq z(x) - w\}$. $\square$

We remark that Hurdle functions can be optimised efficiently by non-elitist algorithms. Sudholt [58] recently showed that Randomised Local Search (RLS) can efficiently optimise Hurdle functions in the presence of noise. These results may be viewed in a different light: if we imagine an algorithm that simulates said noise and runs RLS under simulated noise, the results from [58] apply to a non-elitist algorithm in a noise-free setting. In the following, we restrict ourselves to elitist algorithms.

4. Why Hybridisation is Necessary

4.1. Local Search Algorithms

We show that local search algorithms, when applied on their own, are inefficient, even when using appropriate restart strategies. It is obvious that if the number of zeros z in the initial search point is at least w then all considered local search algorithms will get stuck in a suboptimal local optimum forever. If $w \ll n/2$ then $z \geq w$ holds with very high probability. One way to overcome this problem is to employ a *restart* mechanism, which restarts the local search algorithm once a local optimum has been found.

We focus on $w \leq cn$ for some constant $c < 1/2$ as, if $w \geq c'n$ for some constant $c' > 1/2$, with overwhelming probability the initial search point would lie in the basin of attraction of the global optimum and the algorithm would never face any hurdles.

Theorem 2. *The expected runtime of local search algorithms BILS and FILS with $\delta \geq w$, restarting after δ iterations of the local search, on* Hurdle *problems with hurdle width $w \leq cn$ for some constant $c < 1/2$ is $2^{\Omega(n)}$.*

Proof. The local search algorithm flips one bit and only accepts new search points with strictly better fitness value compared to the current one in each iteration; therefore, the initial search point *decides* whether the global optimum can be reached. It is clear that this search point needs to have at most $w-1$ zeros in order for the global optimum to be reached, i.e., lies on the basin of attraction of the global optimum (see Fig. 1). By initialisation, the number of zeros in the initial search point is binomially distributed with n trials and success probability $1/2$. In expectation, there are $n/2$ zeros, and by Chernoff bounds [59, p. 70], a search point with at most $w-1 \leq cn-1 = (1-\delta)(n/2)$ zeros, where $\delta = 1-2c+2/n = 1-2c+o(1) > 0$ as $c < 1/2$, is sampled with probability at most $e^{-((1-2c+o(1))^2/2)\cdot(n/2)} = 2^{-\Omega(n)}$. This means that the expected number of restarts until this event happens is at least $2^{\Omega(n)}$. The proof is completed by noting that every restart leads to at least one function evaluation. □

4.2. (1+1) Evolutionary Algorithm

We show in this section a tight bound $\Theta(n^w)$ on the expected runtime of the (1+1) EA on the Hurdle problems with an arbitrary hurdle width $2 \leq w \leq n/2$. This result implies that the (1+1) EA is not efficient on Hurdle unless the hurdle width w is very small. Our rigorous analysis complements the non-rigorous arguments given in [19].

Theorem 3. *The expected runtime of the (1+1) EA with $p_m = 1/n$ on* Hurdle *with hurdle width $w \geq 2$ is $\mathcal{O}(n^w)$.*

Proof. For a current search point with z zeros, let p_z denote the probability of finding a strictly better search point through mutation. Hence the expected waiting time to find a better search point is $1/p_z$. Owing to elitism, such a step is required at most once as the algorithm will never return to a search point with z zeros. This is the essence of the well-known fitness-level method [60], and it shows that the expected runtime is bounded from above by $\sum_{z=1}^{n-1} 1/p_z$.

We estimate p_z from below. If $\mathrm{rem}(z, w) > 0$ then the fitness can be improved by mutation flipping only a single zero. This happens with probability $p_z \geq z/n \cdot (1-1/n)^{n-1} \geq z/(en)$.

Now assume that $\mathrm{rem}(z, w) = w$ and $z > 0$ (that is, $z = iw$ for some integer i with $1 \leq i \leq \lfloor n/w \rfloor$). Lemma 1 yields that all search points with at most $z - w$ zeros have a strictly larger fitness. For such a mutation we just need to flip at least w zeros simultaneously and maintain the $n - w$ remaining bits. The former event happens with probability $(1/n)^w$, while it is $(1 - 1/n)^{n-w}$ for the latter. Hence, the probability of obtaining a better solution is lower bounded by $p_z \geq \binom{z}{w}(1/n)^w(1 - 1/n)^{n-w} \geq (1/en^w)(z/w)^w$ since $(1 - 1/n)^{n-1} \geq 1/e$ for all $n \geq 1$ [59, p. 435] and $\binom{z}{w} \geq (z/w)^w$ [59, p. 434].

Together, we obtain an upper bound of

$$\begin{aligned}\sum_{z=1}^{n} \frac{1}{p_z} &\leq \sum_{z=1}^{n} \frac{en}{z} + \sum_{i=1}^{\lfloor n/w \rfloor} en^w \left(\frac{w}{iw}\right)^w \\ &= en \sum_{z=1}^{n} \frac{1}{z} + en^w \sum_{i=1}^{\lfloor n/w \rfloor} \frac{1}{i^w} \\ &\leq \mathcal{O}(n \log n) + en^w \sum_{i=1}^{\infty} \frac{1}{i^2} = \mathcal{O}(n^w),\end{aligned}$$

where in the last line we used $\sum_{z=1}^{n} 1/z = \mathcal{O}(\log n)$ and $\sum_{i=1}^{\infty} 1/i^2 = \pi^2/6$. □

To derive the lower bound, we again focus on $w \leq n/2$ for the same line of arguments as in Theorem 2.

Theorem 4. *The expected runtime of the (1+1) EA using $p_m = 1/n$ on* HURDLE *with $2 \leq w \leq n/2$ is $\Omega(n^w)$.*

Proof. The main observation is that, once the (1+1) EA has reached a search point with w zeros, the optimum is the only other search point that will be accepted. The probability of creating the optimum from any search point with w zeros is $(1/n)^w(1 - 1/n)^{n-w} \leq (1/n)^w$ and the expected waiting time for this event is at least n^w.

The claim follows if we can show that a search point with w zeros is reached with probability $\Omega(1)$ as then by the law of total expectation the expected time is $\Omega(1) \cdot n^w = \Omega(n^w)$.

The number of zeros in the initial search point is at least $n/2 \geq w$ with probability at least $1/2$ (by the symmetry of the binomial distribution [61]). We now assume that this event happens and consider the first point in time where the (1+1) EA discovers a search point with at most w zeros (we pessimistically ignore the time it takes to get there). Note that if the (1+1) EA makes a jump from a search point with $i > w$ zeros to a search point with $j < w$ zeros, the algorithm may avoid the hurdle at w zeros altogether. Hence we need to argue that with probability $\Omega(1)$ the (1+1) EA does not "jump over" the hurdle at w zeros.

Let x be the first search point with $z(x) \leq w$ and let $i > w$ denote the number of zeros in the previous search point. Following [62] we use the notation $\mathrm{mut}(n - i, n - j)$ to denote the probability that a mutation of a search point with $n - i$ ones creates an offspring with $n - j$ ones.

Then, using Lemma 9 in [62] in the last step,

$$\Pr\left(z(x) = w \mid z(x) \leq w\right) = \frac{\Pr\left(z(x) = w\right)}{\Pr\left(z(x) \leq w\right)} = \frac{\mathrm{mut}(n - i, n - w)}{\sum_{j=n-w}^{n} \mathrm{mut}(n - i, j)} \geq \frac{1}{2}.$$

This shows that a search point with w zeros is reached with probability at least $\Omega(1)$, completing the proof. $\square$

Theorems 3 and 4 together provide a tight bound on the expected runtime of the (1+1) EA on the Hurdle problem.

Theorem 5. *The expected runtime of the (1+1) EA using $p_m = 1/n$ on* Hurdle *with $2 \leq w \leq n/2$ is $\Theta(n^w)$.*

We remark that increasing the mutation rate can speed up the (1+1) EA. As the above analysis has shown, the expected run time is dominated by the time to locate the global optimum from a search point with Hamming distance w to it (also referred to as the final hurdle). From this starting point, a mutation rate of w/n maximises the probability of flipping exactly w bits. This mutation rate was also proven in [63] to be *optimal* for the Jump_w function, where the same "jump" is required in case of Hurdle.

This insight is not new, and a mutation rate of w/n was already used in [19] for the (1+1) EA on Hurdle. However, even choosing an optimal mutation rate does not help much as we will show in the following. We first show an upper bound for the (1+1) EA with mutation rate $p_m = w/n$.

Theorem 6. *The expected runtime of the (1+1) EA using $p_m = w/n$ on* Hurdle *with hurdle width $2 \leq w \leq n/3$ is $\mathcal{O}(n^w \cdot (e/w)^w)$.*

Proof. We define and estimate p_z as in the proof of Theorem 3. If $\text{rem}(z, w) > 0$ then we again rely on mutations flipping a single zero, which occur with probability at least

$$p_z \geq z \cdot \frac{w}{n} \cdot \left(1 - \frac{w}{n}\right)^{n-1} \geq \frac{zw}{n} \cdot \left(1 - \frac{w}{n}\right)^{2(n-w)} \geq \frac{zwe^{-2w}}{n}$$

From a current local optimum with $z > 0$ zeros, $\text{rem}(z, w) = 0$, the algorithm flips at least w bits (and keeps other bits unchanged) to reach a better fitness value. The probability of this event is $\binom{z}{w}(w/n)^w(1 - w/n)^{n-w} \geq z^w/(e^w n^w)$, and the waiting time until this happens is at most $(en/z)^w$. By the fitness-level method, the expected number of iterations until the global optimum is found is at most

$$e^{2w}(n/w)\sum_{z=1}^{n}\frac{1}{z} + \left(\frac{en}{w}\right)^w \sum_{i=1}^{\lfloor n/w \rfloor}\frac{1}{i^w} = \mathcal{O}(e^{2w}(n\log(n)/w) + (en/w)^w).$$

The proof is completed by arguing that the term $(en/w)^w$ dominates the overall expression. More formally, we claim $(n/(ew))^w = \omega(n \log n)$ for all $2 \leq w \leq n/3$, which implies $\mathcal{O}(e^{2w}(n\log(n)/w) + (en/w)^w) = \mathcal{O}((en/w)^w)$. The derivative of the function $(n/(ew))^w$ of w is $(n/(ew))^w \cdot \ln(n/(e^2 w))$, hence the function is increasing with w in the interval $[2, n/e^2]$ and decreasing in $[n/e^2, n/3]$. The claim $(n/(ew))^w = \omega(n \log n)$ is easily verified for both $w = 2$ and $w = n/3$, hence it holds for all $2 \leq w \leq n/3$. $\square$

For the lower bound, we need to generalise the result of Lemma 9 in [62] for mutation rate $p_m = w/n$. We reuse the notation $\text{mut}(i, i + k)$ for each $0 \leq i \leq n$ as in [62] to denote the probability that a global mutation (with mutation rate p_m) of a search point with i ones creates an offspring with $i+k$ ones. Note that this is the probability of obtaining an offspring with $n-(i+k)$ zeros from an individual with $n-i < n-(i+k)$ zeros via mutation. For our purpose, we focus only on the final hurdle, and the following lemma shows that the probability of ending up at a local optimum with w zeros, given that a search point with at most w zeros is reached, is at least $1/2$ when the hurdle width is chosen appropriately.

Lemma 7. *Consider mutation rate $p_m = w/n$, where $2 \le w \le \sqrt{n} - 1/2$. For all $0 \le i < n-w \le j \le n$,*

$$\frac{\mathrm{mut}(i,j)}{\sum_{k=j}^{n} \mathrm{mut}(i,k)} \ge \frac{1}{2}.$$

Proof. We also note that an offspring with $j+1$ ones is created if and only if there is an integer $\ell \in \mathbb{N}$ such that ℓ ones and $j+1-i+\ell$ zeros are simultaneously flipped. For simplicity, let $\chi(\ell) := (n-j-\ell)/(j-i+1+\ell)$, and note that $\chi(\ell) \le \chi^* := (n-j)/(j-i+1)$ for all $\ell \ge 0$. The probability of the said event is

$$\begin{aligned}
\mathrm{mut}(i,j+1) &= \sum_{\ell=0}^{n} \binom{i}{\ell}\binom{n-i}{j+1-i+\ell} p_m^{j+1-i+2\ell}(1-p_m)^{n-j-1+i-2\ell} \\
&= \sum_{\ell=0}^{n} \binom{i}{\ell}\binom{n-i}{j-i+\ell} \frac{p_m\chi(\ell)}{1-p_m} p_m^{j-i+2\ell}(1-p_m)^{n-j+i-2\ell} \\
&\le \sum_{\ell=0}^{n} \binom{i}{\ell}\binom{n-i}{j-i+\ell} \frac{p_m\chi^*}{1-p_m} p_m^{j-i+2\ell}(1-p_m)^{n-j+i-2\ell} \\
&= \frac{p_m\chi^*}{1-p_m} \sum_{\ell=0}^{n} \binom{i}{\ell}\binom{n-i}{j-i+\ell} p_m^{j-i+2\ell}(1-p_m)^{n-j+i-2\ell} \\
&= \frac{(n-j)p_m}{(j-i+1)(1-p_m)} \cdot \mathrm{mut}(i,j) \\
&\le \frac{w^2}{2(n-w)} \cdot \mathrm{mut}(i,j)
\end{aligned}$$

since $n-j \le w$, $j-i \ge (n-w)-(n-w-1) = 1$ and $p_m = w/n$. Because $w \le \sqrt{n} - 1/2 \le \sqrt{n+1/4} - 1/2$, we then get $w^2 \le n-w$, which is equivalent to $w^2/(2(n-w)) \le 1/2$. Therefore, we get $\mathrm{mut}(i,j+1) \le (1/2)\cdot \mathrm{mut}(i,j)$, and, by induction, $\mathrm{mut}(i,j+k) \le 2^{-k}\cdot \mathrm{mut}(i,j)$ for any $0 \le k \le n-j$. This leads to

$$\begin{aligned}
\frac{\mathrm{mut}(i,j)}{\sum_{m=j}^{n}\mathrm{mut}(i,m)} &= \frac{\mathrm{mut}(i,j)}{\sum_{k=0}^{n-j}\mathrm{mut}(i,j+k)} \\
&\ge \frac{\mathrm{mut}(i,j)}{\sum_{k=0}^{n-j} 2^{-k}\mathrm{mut}(i,j)} = \frac{1}{\sum_{k=0}^{n-j} 2^{-k}} \ge \frac{1}{\sum_{k=0}^{\infty} 2^{-k}} = \frac{1}{2},
\end{aligned}$$

which proves the lemma. $\square$

Theorem 8. *The expected runtime of the (1+1) EA using $p_m = w/n$ on* Hurdle *with $2 \le w \le \sqrt{n} - 1/2$ is $\Omega(n^w \cdot (e/w)^w)$.*

Proof. Following Theorem 4, the probability of ending up with a search point having at least w zeros is at least $1/2$. By Lemma 7 we obtain a lower bound of $1/2$ on the probability of ending up at a local optimum with w zeros, given that a search point with at most w zeros is reached. Thus, with probability $\Omega(1)$ we reach the local optimum with w zeros. Assume that this has actually happened, the global optimum is the only search point with an improved fitness (by Lemma 1) and the probability of creating the optimum by mutation is $(w/n)^w(1-w/n)^{n-w} = \mathcal{O}((w/(en))^w$, where the inequality follows from $(1-w/n)^{n-w} = \left((1-w/n)^{n/w}\right)^w (1-w/n)^{-w} \leq e^{-w}(1-1/\sqrt{n})^{-\sqrt{n}} = \mathcal{O}(e^{-w})$. The expected waiting time for such a mutation is $\Omega((en/w)^w)$ and, by the law of total expectation, the expected runtime is at least $\Omega(1)\cdot\Omega((en/w)^w) = \Omega((en/w)^w)$. □

We are ready to conclude a tight bound on the expected runtime of the (1+1) EA on Hurdle problem with $2 \leq w = \mathcal{O}(\sqrt{n})$. Asymptotically speaking, the optimal mutation rate $p_m = w/n$ does speed up the optimisation process since it results in an improvement by a factor of $(e/w)^w$, which is smaller than one if the hurdle width is $w \geq 3 > e$. However, the expected runtime is still superpolynomial for all $w = \omega(1)$ with $w \leq n/2$.

Theorem 9. *The expected runtime of the (1+1) EA using mutation rate $p_m = w/n$ on* Hurdle *with $2 \leq w \leq \sqrt{n} - 1/2$ is $\Theta(n^w \cdot (e/w)^w)$. For $\sqrt{n} - 1/2 \leq w \leq n/2$ it is $\Omega(e^w)$.*

Proof. The first statement follows from Theorems 6 and 8. For the second statement the probability of creating the optimum by mutation from any non-optimal search point with z zeros is at most $(w/n)^z(1-w/n)^{n-z} \leq (1-w/n)^n \leq e^{-w}$ where the last term holds irrespective of z. Hence the expected waiting time to generate the optimum is at least e^w. Along with the fact that we start in a non-optimal search point with probability $1 - 2^{-n} = \Omega(1)$, this proves a lower bound of $\Omega(e^w)$. □

5. The (1+1) MA is Efficient

We now show that, in contrast to local search on its own and the (1+1) EA, the (1+1) MA can find the global optimum efficiently, for both BILS and FILS. Note in particular we consider BILS and FILS with local search depth $\delta \geq w$ as this is sufficient to run into local optima from anywhere in the search space. Otherwise, local search may finish short of the next local optimum, unless mutation flips many 0-bits to 1. We will discuss this at the end of this section.

We first provide an upper bound on the expected number of iterations needed. This does not include the function evaluations made during local search, which will be bounded separately.

Theorem 10. *The expected number of iterations of the (1+1) MA using BILS or FILS with $\delta \geq w$ on* Hurdle *with any hurdle width $2 \leq w \leq n$ is $\mathcal{O}\left(n^2/w^2\right)$.*

Proof. Assume that an optimum $x \neq 1^n$ is the current search point with z zeros. By Lemma 1, we know that all search points with up to $z - w$ zeros have a better fitness than $f(x)$. Such a search point can be reached if the (1+1) MA flips at least two zeros to jump from x to another search point with at most $z - 2$ zeros with probability $p_z = \binom{z}{2}(1/n)^2(1-1/n)^{n-2} \geq z(z-1)/(2en^2)$, as then a local search algorithm hill-climbs to locate a local optimum with $z-w$ zeros. Hence, the probability of reaching a better local optimum from x is bounded from below by p_z, and the expected waiting time until this happens is at most $1/p_z$. By summing up for all hurdles, that is, $z = iw$ with $1 \leq i \leq \lfloor n/w \rfloor$, the expected number of iterations until the global optimum is found is $\sum_{i=1}^{\lfloor n/w \rfloor} 2en^2/(iw(iw-1))$. By noting the identity $\sum_{i=1}^{\infty} 1/i^2 = \pi^2/6$, we then get

$$\sum_{i=1}^{\infty} \frac{1}{iw(iw-1)} \leq \sum_{i=1}^{\infty} \frac{1}{iw(iw-iw/2)} = \frac{2}{w^2} \sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{3w^2}.$$

Hence, the expected runtime is at most $2en^2\pi^2/(3w^2) = \mathcal{O}\big(n^2/w^2\big)$. □

To bound the number of function evaluations made during local search, we distinguish between local searches that result in a strict improvement over the previous current search point (referred to as *improving local searches*), and those that do not (*non-improving local searches*). This is an example of the *accounting method* [64, Chapter 17.2], where function evaluations are charged to one of two accounts and the total costs are bounded separately for each account. Adding the two bounds will yield an upper bound on the total number of function evaluations made during local search.

We first bound the number of function evaluations spent in any improving local search. The following lemma gives a general bound for the number of evaluations in any local search, regardless of the starting point and whether it is improving or non-improving.

Lemma 11. *The number of function evaluations spent in any local search call on* Hurdle *with hurdle width w and $\delta \geq w$, from every initial search point, is at most wn for BILS and at most $2n$ for FILS.*

Proof. Recall that Hurdle is a function of unitation, that is, all search points with the same number of zeros have the same fitness. Hence if a local move from a search point with i zeros finds a strictly better fitness for a search point with $i-1$ or $i+1$ zeros, local search will never return to a search point with i zeros. This means that local search will either always increase the number of zeros or it will always decrease the number of zeros until a local optimum is found. In both cases, a local optimum will be found after at most $w-1$ iterations and then local search stops after at most n further evaluations.

As BILS makes n function evaluations in every iteration, it makes at most $n(w-1)+n = wn$ function evaluations in total. For FILS, after one iteration of the outer for loop (see Algorithm 4) a local optimum will be found, and it will stop after n further evaluations. □

The expected number of function evaluations for non-improving local searches (particularly BILS) can be bounded more tightly as follows. In contrast to Lemma 11, the following lemma is tailored to the (1+1) MA. It exploits that mutations of a local optimum usually stay close to local optima. Hence if local search runs back into a local optimum of the same quality, local search tends to terminate quickly.

Lemma 12. *Considering the (1+1) MA on* Hurdle, *the expected number of function evaluations spent by BILS and FILS during any non-improving local search is* $\Theta(n)$.

Proof. The lower bound $\Omega(n)$ is trivial as both local searches make at least n function evaluations before stopping.

Let i denote the number of zeros in the current search point of the (1+1) MA and j denote the number of zeros in the search point after mutation, from which local search is called.

If $\text{rem}(i, w) = 0$, that is, i is a local optimum, $j < i - 1$ will lead to an improving local search (see Fig. 1), and $j = i - 1$ may either be improving or go back to a search point with i zeros in one iteration. If $j \geq i$ then local search will make at most $j - i$ iterations.

If $\text{rem}(i, w) > 0$, that is, i is not a local optimum, $j < w \cdot \lceil i/w \rceil$ leads to an improving local search, whereas $j \geq w \cdot \lceil i/w \rceil$ will stop after at most $j - i$ iterations.

In all these cases, local search is either improving, or it makes at most $|j - i|$ iterations. Note that a necessary condition of mutating a search point with i zeros into one with j zeros is that at least $|j-i|$ bits flip. The probability for this event is at most $\binom{n}{|j-i|} n^{-|j-i|} \leq 1/(|j - i|!)$. The expected number of iterations in a non-improving local search is thus at most

$$\sum_{j=0}^{n} |j - i| \cdot \frac{1}{|j - i|!} \leq 2 \sum_{d=1}^{\infty} d \cdot \frac{1}{d!} = 2 \sum_{d=0}^{\infty} \frac{1}{d!} = 2e.$$

The number of function evaluations made during a local search that stops after s iterations is at most $(s+1)n$. Hence the expected number of function evaluations is at most $(2e + 1)n$. □

Now we are ready to show an upper bound on the expected runtime of the (1+1) MA on the Hurdle problem.

Theorem 13. *The expected number of function evaluations of the (1+1) MA on* Hurdle *with any hurdle width* $2 \leq w \leq n/2$ *and* $\delta \geq w$ *is* $\mathcal{O}(n^2 + n^3/w^2)$ *when using BILS and* $\mathcal{O}(n^3/w^2)$ *when using FILS.*

Note that if the hurdle width is $w = \Omega(n)$, the expected runtime of the (1+1) MA with FILS is only $\mathcal{O}(n)$. Then the algorithm is as efficient on Hurdle as on the underlying function OneMax without any hurdles.

Proof of Theorem 13. By Theorem 10 the expected number of iterations is bounded by $\mathcal{O}(n^2/w^2)$. The expected number of function evaluations spent in any non-improving local search is $\mathcal{O}(n)$ according to Lemma 12. Together, the number of function evaluations in all non-improving local searches is at most $\mathcal{O}(n^3/w^2)$.

By Lemma 11, the number of function evaluations in any improving local search is at most wn when using BILS and at most $2n$ when using FILS. Since every improving local search ends in a local optimum with better fitness than the current search point of the (1+1) MA, there can only be $\mathcal{O}(n/w)$ improving local searches as this is a bound on the number of fitness levels containing local optima. Hence the effort in all improving local searches is bounded by $\mathcal{O}(n^2)$

when using BILS and $\mathcal{O}(n^2/w)$ when using FILS. Hence the overall number of function evaluations is bounded by $\mathcal{O}(n^2 + n^3/w^2)$ when using BILS and $\mathcal{O}(n^2/w + n^3/w^2) = \mathcal{O}(n^3/w^2)$ when using FILS. □

We also show matching lower bounds for the (1+1) MA with both local searches. We first show a very general lower bound of $\Omega(n^2)$ for the (1+1) MA with BILS. It holds for all functions with a unique global optimum and may be of independent interest.

Theorem 14. *The (1+1) MA using BILS makes at least $\Omega(n^2)$ function evaluations, with probability $1 - 2^{-\Omega(n)}$ and in expectation, on any function with a unique global optimum.*

Proof. It suffices to show the high-probability statement as the expectation is at least $(1 - 2^{-\Omega(n)}) \cdot \Omega(n^2) = \Omega(n^2)$. We show that with probability $1 - 2^{-\Omega(n)}$ one of the following events occurs.

A: The (1+1) MA spends at least $n/12$ iterations before finding the optimum.

B: BILS makes a total of at least $n/6$ iterations before finding the optimum.

Each event implies a lower bound of $\Omega(n^2)$ as each iteration of BILS makes n function evaluations, and each iteration leads to at least one iteration of BILS. For none of these events to occur, the (1+1) MA must find the optimum within $n/12$ iterations, using fewer than $n/6$ iterations of BILS in total. For this to happen, one of the following rare events must occur:

E_1: the (1+1) MA is initialised with a search point that has a Hamming distance less than $n/3$ to the unique optimum or

E_2: the initial search point has a Hamming distance of at least $n/3$ to the optimum, and the algorithm decreases this distance to 0 during the first $n/12$ iterations, using fewer than $n/6$ iterations of BILS.

The reason is that, if none of the events E_1 and E_2 occurs, then this implies $A \cup B$. By contraposition, $\overline{A \cup B} \Rightarrow E_1 \cup E_2$ and $\text{Prob}\left(\overline{A \cup B}\right) \leq \text{Prob}\,(E_1 \cup E_2) \leq \text{Prob}\,(E_1) + \text{Prob}\,(E_2)$ by the union bound (also called Boole's inequality [59, p. 44]). Event E_1 has probability $\text{Prob}\,(E_1) \leq 2^{-\Omega(n)}$ by Chernoff bounds. For E_2, note that each iteration of local search can decrease the Hamming distance to the optimum by at most 1. Hence all iterations of BILS can only decrease the Hamming distance to the optimum by $n/6$ in total, and so the remaining distance of $n/3 - n/6 = n/6$ needs to be covered by mutations. Each flipping bit can decrease the distance to the optimum by at most 1. We have at most $n/12$ mutations, hence the expected number of flipping bits is at most $n/12$. The probability that at least $n/6$ bits flip during at most $n/12$ mutations is $2^{-\Omega(n)}$, which follows from applying Chernoff bounds to i.i.d. indicator variables $X_{i,t} \in \{0, 1\}$ that describe whether the i-th bit is flipped during iteration t or not. Hence $\text{Prob}\,(E_2) \leq 2^{-\Omega(n)}$. Together, we have by the union bound,

$$\begin{aligned}\text{Prob}\left(\overline{A \cup B}\right) &\leq \text{Prob}\,(E_1) + \text{Prob}\,(E_2)\\ &\leq 2^{-\Omega(n)} + 2^{-\Omega(n)} \leq 2^{-\Omega(n)}.\end{aligned}$$

This completes the proof. □

Theorem 15. *The expected number of function evaluations of the (1+1) MA on the* Hurdle *problems with $3 \leq w \leq n/2$ and $\delta \geq w$ is bounded from below by $\Omega(n^2 + n^3/w^2)$ when using BILS and $\Omega(n^3/w^2)$ when using FILS.*

Proof. A bound of $\Omega(n^2)$ for the (1+1) MA with BILS follows from Theorem 14.

We prove lower bounds $\Omega(n^3/w^2)$ for both local searches by considering the remaining time when the (1+1) MA has reached a local optimum with w zeros. The proof of Theorem 4 has revealed that the (1+1) EA reaches such a local optimum with probability $\Omega(1)$, and it is obvious that the same statement also holds for the (1+1) MA as local search can never "jump over" the hurdle with w zeros. Then a lower bound of $\Omega(n^3/w^2)$ follows from showing that the expected number of function evaluations starting with a local optimum having w zeros is $\Omega(n^3/w^2)$.

From such a local optimum, the (1+1) MA with BILS has to flip at least two zeros in one mutation. Otherwise, the offspring will have at least $w - 1$ zeros, and, since $w \geq 3$, BILS will run back into a local optimum with w zeros (or a worse local optimum). The probability for such a mutation is at most $\binom{w}{2} \cdot 1/n^2 = \mathcal{O}\big(w^2/n^2\big)$, and the expected number of iterations, until such a mutation happens, is at least $\Omega(n^2/w^2)$.

The same statement holds for the (1+1) MA with FILS: here it is necessary to either flip at least two zeros as above or to create a search point with $w-1$ zeros and to have FILS find a search point with $w - 2$ zeros as the first improvement. In the latter case, FILS will find the global optimum. The probability of creating a search point with $w - 1$ zeros is at most w/n as it is necessary to flip one of w zeros. In this case, FILS creates a search point with $w - 2$ zeros as first improvement if and only if the first bit to be flipped is a zero. Since there are $w - 1$ zeros, and each bit has the same probability of $1/n$ of being the first bit flipped, the probability of the first improvement decreasing the number of zeros is $(w - 1)/n$. Together, the probability of an iteration creating the global optimum is still $\mathcal{O}\big(w^2/n^2\big)$, and the expected number of iterations is still at least $\Omega(n^2/w^2)$. In every iteration, both BILS and FILS make at least n evaluations. Hence we obtain $\Omega(n^3/w^2)$ as a lower bound on the number of function evaluations. □

Theorems 13 and 15 now give tight bounds on the expected runtime of the (1+1) MA on the Hurdle problem.

Theorem 16. *The expected number of function evaluations of the (1+1) MA on* Hurdle *with any hurdle width $3 \leq w \leq n/2$ and $\delta \geq w$ is $\Theta(n^2 + n^3/w^2)$ when using BILS and $\Theta(n^3/w^2)$ when using FILS.*

Note that the results for the (1+1) MA hold for the standard mutation rate $p_m = 1/n$. For the (1+1) EA, an increased mutation rate of $p_m = w/n$ was shown to yield better performance as it increased the chance to flip w bits in one mutation. For the (1+1) MA on Hurdle this is not necessary; as our analysis has shown, mutation only needs to flip 1 or 2 bits to reach the basin of attraction of the next local optimum, and then local search produces a better local optimum.

So far we always assumed that the local search depth δ is chosen as $\delta \geq w$ for the local search algorithm to run into a local optimum. We remark that this parameter can have a huge impact on the success of the (1+1) MA. More specifically, when $\delta < w$ and we are at a local optimum with iw zeros (where

$i \in \mathbb{N} \setminus \{0\}$), the algorithm has to flip at least $w - \delta$ zeros (while maintaining others unchanged) to obtain a search point with $(i-1)w + \delta$ zeros, from which the next local optimum can be found. Otherwise, the local search will end up at a search point whose fitness is worse than that of the current local optimum, and eventually the move will be rejected.

6. A Proof that Crossover in Memetic Algorithms is Beneficial

In evolutionary computation, there is an ongoing debate started decades ago, about the usefulness of crossover (see, e. g. [65, 66] for recent viewpoints that have been discussed controversially). Since then, many experimental studies showed a substantial benefit of crossover and more effort has been devoted to understand the effect of crossover via rigorous theoretical proofs, including artificially constructed functions [38, 42, 40], colouring problems [44, 45], all-pairs shortest paths [46] and on OneMax as a simple hill-climbing task [50, 49, 51, 52].

Despite these works, to the best of our knowledge, there is no runtime analysis of memetic algorithms that use crossover. This leaves open the question of whether crossover is beneficial in MAs. In this section, we show for the first time that crossover can speed up MAs, leading to a significant speedup on Hurdle. To this end, we consider the (2+1) MA with a uniform crossover as defined in Algorithm 3.

The following theorem gives our main result for this section.

Theorem 17. *The expected runtime of the (2+1) MA with crossover on the* Hurdle *problem with hurdle width $2 \leq w \leq n/2$ and $\delta \geq w$ is bounded by $\mathcal{O}(n^2 + n^2 \log(n/w)/w)$ when using BILS and $\mathcal{O}\big(n^2 \log(n/w)/w\big)$ when using FILS.*

In comparison to the tight bounds for the (1+1) MA, in the bound for the (2+1) MA with FILS, a factor of n/w is replaced by a factor of $\log(n/w)$. This speedup is non-increasing with w and it is $\Omega(n/\log n)$ for $w = \mathcal{O}(1)$. In the (2+1) MA with BILS the same change applies to the second summand in the runtime bounds. The same speedup compared to the (1+1) MA with BILS applies if $w = \mathcal{O}(\log n)$. For $w = \omega(\log n)$ and $w = o(\sqrt{n})$ the expected time reduces from $\Theta(n^3/w^2) = \omega(n^2)$ to $\mathcal{O}(n^2)$, saving a factor of $\Omega(n/w^2) = \omega(1)$. For $w = \Omega(\sqrt{n})$, the expected time is $\mathcal{O}(n^2)$ for both the (1+1) MA and the (2+1) MA.

Note that the upper bound for the (2+1) MA with BILS can be simplified from $\mathcal{O}(n^2 + n^2 \log(n/w)/w)$ to $\mathcal{O}(n^2 + n^2 \log(n)/w)$ as the second summand only dominates if $w = \mathcal{O}(\log n)$, and then $\log(n/w) = \Theta(\log n)$. We stick to the non-simplified formula for consistency with the (2+1) MA using FILS.

Proof of Theorem 17. It is easy to see that within expected $\mathcal{O}(1)$ iterations, all population members will be local optima. This is because for each individual there is a probability of $1/4 \cdot (1 - 1/n)^n = \Omega(1)$ of selecting it twice as a parent and for mutation not flipping any bits, which implies that local search will run into a local optimum of better fitness. By Lemma 11, the expected number of evaluations in this initial phase is at most $\mathcal{O}(wn)$ with BILS and at most $\mathcal{O}(n)$ with FILS.

Afterwards, since both local searches always stop in a local optimum, the population will always only contain local optima. We, therefore, focus on estimating the number of function evaluations while the population consists of local optima. In other words, every search point has iw zeros for some integer i with $1 \leq i \leq \lfloor n/w \rfloor$.

The main observation is that crossover can speed up the expected time for reaching the basin of attraction of better local optima. However, this crucially depends on the state of the current population. A requirement for crossover to be effective is that the population contains some amount of diversity. If the population contains two copies of the same search point, crossover has no effect and the algorithm effectively just performs mutation and local search as in the (1+1) MA. Likewise, if the population contains local optima of different fitness, it is hard to show how the best fitness might be improved. As we will show, crossover does work provably effectively if the population contains two search points that have the same number of zeros but different genotypes.

In such a population, crossover can make larger jumps than mutation on the scale given by the number of zeros. The ability of larger jumps is a double-edged sword, though. While it can reduce the time needed to jump closer to the optimum and reaching the basin of attraction of a better local optimum, crossover can also lead to larger jumps away from the optimum. This can increase the number of function evaluations required by BILS in a non-improving local search. (This issue does not concern FILS as by Lemma 11 it always terminates in $\mathcal{O}(n)$ function evaluations.) Recall that for the (1+1) MA, in Lemma 12 we argued that the time spent in a non-improving local search is $\Theta(n)$ and the proof revealed that this is because mutation typically only makes small jumps. This argument does not apply when performing a crossover of diverse parents; in this case, we only have an upper bound of wn evaluations during BILS from Lemma 11. On balance, though, crossover leads to the claimed speedup. This discussion shows that careful analysis is needed.

In the following, we use the accounting method again, but different compared to our analysis of the (1+1) MA. We consider the number of function evaluations spent in certain iterations, distinguishing the following cases:

1. the population contains two individuals with different fitness values,
2. the population contains two identical genotypes,
3. the population contains two individuals with the same number of zeros and Hamming distance 2, and
4. the population contains two individuals with the same number of zeros and Hamming distance at least 4.

Note that the Hamming distance of two individuals with the same number of zeros must be an even value. Hence, after the initial phase, every population will fall in one of these cases. So all evaluations are accounted for and the total expected runtime is bounded by the sum of (bounds of) times spent in these four cases (and in the initial phase).

The following lemmas bound the total number of evaluations the algorithm spends in these four cases. Note that in the second and third case there is no diversity or only a small amount of diversity in the population, hence crossover will have no effect or only make small jumps. In the first and the last case, we have no bound on the jump length during crossover, but we will show that in both these cases improvements over the worst parent can be found quickly.

Lemma 18. *In the context of Theorem 17, the expected number of function evaluations spent in all iterations where the population contains two individuals of different fitness values is $\mathcal{O}(n^2)$ with BILS and $\mathcal{O}(n^2/w)$ with FILS.*

Proof. Let iw be the maximum number of zeros in the population and $jw < iw$ be the number of zeros in the other local optimum.

With probability $1/4$, parent selection picks the fitter parent twice. If the subsequent mutation does not flip any bits then another search point with jw zeros is created and the maximum number of zeros in the population decreases for good. The probability of this sequence of events is $1/4 \cdot (1-1/n)^n = \Omega(1)$.

Hence the expected number of iterations for every fixed i is $\mathcal{O}(1)$. There are only $\mathcal{O}(n/w)$ values for i and by Lemma 11 every iteration leads to at most wn evaluations during BILS and at most $2n$ evaluations during FILS. Multiplying the latter with $\mathcal{O}(n/w)$ proves the claim. □

Lemma 19. *In the context of Theorem 17, the expected number of function evaluations spent in all iterations where the population contains two identical search points is $\mathcal{O}(n^2 + (n^2 \log(n/w))/w)$ with BILS and $\mathcal{O}((n^2 \log(n/w))/w)$ with FILS.*

Proof. Let iw be the number of zeros in the two individuals. Note that if a different individual with iw zeros is generated, the algorithm will never return to two identical individuals with iw zeros as the tie-breaking rule in the survival selection will prevent such a duplicate from entering the population. Also note that crossover has no effect since the parents are identical. Hence the algorithm temporarily behaves like the (1+1) MA.

Now, flipping exactly a zero and a one will create a new search point with a different genotype and equal fitness. The probability for such a mutation is at least $iw(n-iw)/(en^2)$ and the expected number of iterations until this happens is at most $en^2/(iw(n-iw))$.

Since the algorithm temporarily behaves like the (1+1) MA, we can apply Lemma 12 to bound the number of evaluations spent in non-improving iterations. Invoking said lemma yields that the expected number of function evaluations in all non-improving local searches, for any fixed i, is $\mathcal{O}(n^3/(iw(n-iw)))$. By noting that

$$\sum_{i=1}^{\lfloor n/w \rfloor} \frac{1}{iw(n-iw)} \le 2 \sum_{i=1}^{\lceil n/(2w) \rceil} \frac{1}{iw(n-iw)} \le \frac{4}{n} \sum_{i=1}^{\lceil n/(2w) \rceil} \frac{1}{iw},$$

the sum over all states i is

$$\begin{aligned}\sum_{i=1}^{\lfloor n/w \rfloor} \frac{\mathcal{O}(n^3)}{iw(n-iw)} &\le \mathcal{O}(n^2) \sum_{i=1}^{\lceil n/(2w) \rceil} \frac{1}{iw} \\ &= \mathcal{O}\left(\frac{n^2}{w}\right) \sum_{i=1}^{\lceil n/(2w) \rceil} \frac{1}{i} = \mathcal{O}\left(\frac{n^2 \log(n/w)}{w}\right).\end{aligned}$$

Considering the number of evaluations in improving local searches, by Lemma 11, adds a further $\mathcal{O}(n/w) \cdot wn = \mathcal{O}(n^2)$ evaluations for BILS and $\mathcal{O}(n/w) \cdot 2n = \mathcal{O}(n^2/w)$ evaluations for FILS. □

Lemma 20. *In the context of Theorem 17, the expected number of function evaluations spent in all iterations where the population contains two individuals with the same number of zeros and Hamming distance 2 is* $\mathcal{O}(n^2 + (n^2 \log(n/w))/w)$ *with BILS and* $\mathcal{O}((n^2 \log(n/w))/w)$ *with FILS.*

Proof. Let iw be the number of zeros in both individuals. If crossover and mutation create an offspring with at most $iw - 2$ zeros, local search will find a better local optimum. We estimate the probability of this event. With probability at least $1/2$, the parents selected for crossover will be different. Then with probability $1/4$, both differing bits will be set to 1 in the crossover. With probability at least $(iw-1)/(en)$ mutation then flips a further zero and no other bits. Hence the probability of crossover and mutation creating an offspring with at most $iw-2$ zeros is at least $(iw-1)/(8en)$. The expected number of iterations until a better local optimum is found is thus at most $8en/(iw-1) = \mathcal{O}(n/(iw))$.

The number of evaluations made during non-improving local searches can be estimated as in Lemma 12; the only difference is that crossover can increase the difference in the number of zeros between the input for local search and the original parent. However, since both parents have Hamming distance 2, this difference can only increase by at most 2, leading to an additional term of at most $2n$ evaluations, compared to the statement of Lemma 12. Thus the expected number of function evaluations in any iteration is still bounded by $\mathcal{O}(n)$ for both local searches.

The total expected number of evaluations across all values of i is at most

$$\sum_{i=1}^{\lfloor n/w \rfloor} \mathcal{O}\left(\frac{n^2}{iw}\right) = \mathcal{O}\left(\frac{n^2}{w}\right) \sum_{i=1}^{\lfloor n/w \rfloor} \frac{1}{i} = \mathcal{O}\left(\frac{n^2 \log(n/w)}{w}\right).$$

Considering the number of evaluations in improving local searches as in Lemma 19 completes the proof. □

Lemma 21. *In the context of Theorem 17, the expected number of function evaluations spent in all iterations where the population contains two individuals with the same number of zeros and Hamming distance at least 4 is* $\mathcal{O}(n^2)$ *with BILS and* $\mathcal{O}(n^2/w)$ *with FILS.*

Proof. Let iw again be the number of zeros in the two individuals. Recall that if crossover and mutation create an offspring with at most $iw - 2$ zeros, local search will find a better local optimum. The remainder of the proof uses and extends arguments from the proof of Theorem 4 in [49]. We denote by $2d \geq 4$ the Hamming distance between the two parents and by X the number of these bits set to 1 during the uniform crossover. Note that X follows a binomial distribution with parameters $2d$ and $1/2$, with an expectation of d. We argue that crossover creates a surplus of at least 2 ones with probability at least $1/16$. By symmetry, $\Pr(X > d) = \Pr(X < d)$ [61] and thus

$$\begin{aligned}\Pr(X \geq d+2) &= \frac{1}{2} \cdot \Pr(X \neq d) - \Pr(X = d+1) \\ &= \frac{1}{2}\left(1 - 2^{-2d}\binom{2d}{d}\right) - 2^{-2d}\binom{2d}{d+1}\end{aligned}$$

Since both $2^{-2d}\binom{2d}{d}$ and $2^{-2d}\binom{2d}{d+1}$ are non-increasing in d, the worst case is attained for $2d = 4$. Plugging this in, we bound $\Pr(X \geq d+2)$ from below by

$$\frac{1}{2}\left(1 - 2^{-4}\binom{4}{2}\right) - 2^{-4}\binom{4}{3} = \frac{1}{2}\left(1 - \frac{6}{16}\right) - \frac{1}{4} = \frac{1}{16}.$$

Mutation does not flip any bits with probability $(1-1/n)^n = \Omega(1)$ and thus the expected number of iterations until a better local optimum is found is $\mathcal{O}(1)$. Since there are $\mathcal{O}(n/w)$ values of i and every local search call (improving or not) requires at most wn evaluations for BILS and at most $2n$ evaluations for FILS by Lemma 12, this implies the claimed bounds. □

Adding up all times from Lemmas 18, 19, 20, 21 as well as the number of evaluations in the initial phase completes the proof of Theorem 17. □

We believe that the upper bound of $\mathcal{O}\big(n^2 \log(n/w)/w\big)$ for the (2+1) MA with FILS is tight and that a lower bound of $\Omega\big(n^2 \log(n/w)/w\big)$ applies when using either FILS or BILS. If the population contains two identical individuals with iw zeros, for an integer $i \in \mathbb{N}$, progress towards higher fitness values can only be made if mutation flips a 0-bit to 1. This has probability at most iw/n and requires at least $n/(iw)$ iterations. This makes $n^2/(iw)$ function evaluations as every local search call makes at least n evaluations. If, with probability $\Omega(1)$, this situation occurs on all hurdles with $iw \leq i_{\max}w$ zeros with $i_{\max}w = \Omega(n^{\varepsilon})$, for any positive constant ε, the claimed lower bound follows. The challenge is showing that this assumption holds, which involves handling populations of non-identical individuals and showing that the population frequently collapses to identical local optima. We also believe that a lower bound of $\Omega(n^2)$ applies to the (2+1) MA with BILS, however, the analysis of Theorem 14 would need to be adapted to handle potential large jumps through crossover. Since we are mainly interested in upper bounds, we leave this for future work.

7. All Operators are Necessary for Optimal Performance

The success of the (2+1) MA is down to the combination of mutation, crossover and local search. We argue that, when considering elitist $(\mu+\lambda)$-type algorithms, every other combination of operators performs worse.

We have rigorously proved that mutation alone in the (1+1) EA[3] and local search alone are inefficient, and while mutation and local search is a lot more efficient, the expected runtimes are generally larger than the times stated in Theorem 17 for the (2+1) MA. Crossover by itself can get stuck if a bit emerges that is different from the optimum in all members of the population or if the population collapses to copies of the same genotype. There is a positive

[3]We do not have a negative result for arbitrary $(\mu+\lambda)$ EAs. As observed in [25, p. 2520], populations can help to escape from local optima if mutation creates an offspring that is worse than a locally optimal parent and the population contains a search point that is even worse than the offspring, allowing the offspring to survive. For reasonable values of w and μ it is, however, unlikely that this effect decreases the expected runtime as the population quickly tends to gather around the best local optimum [25, p. 2520]. For $w = \omega(\log n)$ we get a superpolynomial expected runtime if a population where all search points have w zeros is reached with a polynomially small probability.

probability that this already happens during initialisation, which establishes an infinite expected runtime[4]. The same holds for crossover and local search if the population collapses to copies of the same local optimum as then crossover and local search have no effect.

We believe that mutation and uniform crossover without local search cannot match the bounds from Theorem 17 either, at least not for arbitrary hurdle widths. In the seminal paper that introduced Hurdle, Prügel-Bennett empirically showed an advantage for a Genetic Algorithm with mutation and uniform crossover for $w = 2$ [19]. However, for larger w we believe that the performance will deteriorate quickly with growing w.

For instance, if the whole population in a $(\mu{+}\lambda)$ Genetic Algorithm consists of search points with w zeros[5], crossover and mutation need to generate the global optimum 1^n as any other search point will be rejected. The best-case scenario, in this case, is that two parents are chosen for a crossover that do not share a zero, that is, all zeros are in mutually different positions (cf. Lemma 2 in [40] for the equivalent scenario on $\textsc{Jump}_w$). The probability that these $2w$ zeros will be set to 1 in the crossover is only $2^{-2w} = 4^{-w}$. Thus, the expected remaining time to find an optimum from a population where all search points have w zeros is at least 4^w, which is superpolynomial for $w = \omega(\log n)$. If the probability of reaching such a population is $\Omega(1)$, or polynomially small, the overall expected time is superpolynomial.

Figure 2 illustrates all possible combinations of operators and the expected runtimes for the respective algorithms, for Hurdle problems with hurdle width $w = \omega(\log n)$. We conclude that on Hurdle a combination of mutation, crossover and local search is vital to achieving the best possible performance.

8. Experiments

We provide additional experiments to investigate the runtime for realistic problem sizes and to get further insights into the constant factors and lower-order terms hidden in our asymptotic expressions. Although we have been looking at the *expected runtime* of the algorithms (denoted as $\mathbb{E}[T]$), there is no way to estimate it from experimental results. However, we can approximate it by averaging runtimes of a large number of independent runs of the algorithm (also called *empirical runtime*, denoted by $\overline{T}$).

Note that although our theoretical results also cover the local search algorithms by themselves, we do not empirically study them here because their limitations on Hurdle are obvious. We only consider the (1+1) EA, the (1+1) MA (using mutation only) and the (2+1) MA with uniform crossover.

[4]We remark that, despite an infinite expectation, there is still a possibility that an algorithm will succeed in polynomial time with high probability if the population size is large enough. Such a result was shown on OneMax by Prugel-Bennett, Rowe and Shapiro [67].

[5]We do not have a formal analysis of the probability that this happens. If w is not very small, we believe that it is quite high as then it is very likely that at least one individual will eventually reach a local optimum with w zeros and then quickly take over the whole population (cf. Lemma 2 in [49]).

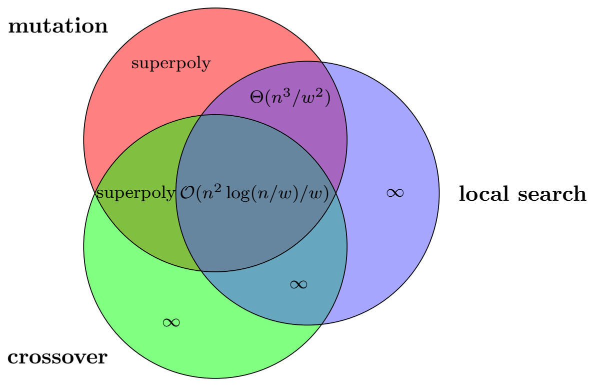

Figure 2: Overview of expected runtimes for all combinations of operators on Hurdle problems with hurdle width $w = \omega(\log n)$. For local search we show results for FILS as the better performing operator; results for BILS include an additional term of n^2. The lower bounds for mutation and that for mutation and crossover assume that a population where all search points have w zeros is reached with at least polynomially small probability.

8.1. (1+1) EA

For the (1+1) EA, we are interested in how the empirical runtime grows with problem instance size n. To do this, we run the algorithm with mutation rate $1/n$ and $w \in \{2, 3, 4\}$ for $n \in \{10, 15, \ldots, 50\}$. For each value of n, the algorithm is run 100 times. The empirical runtimes and their best-fit models are described in Fig. 3.

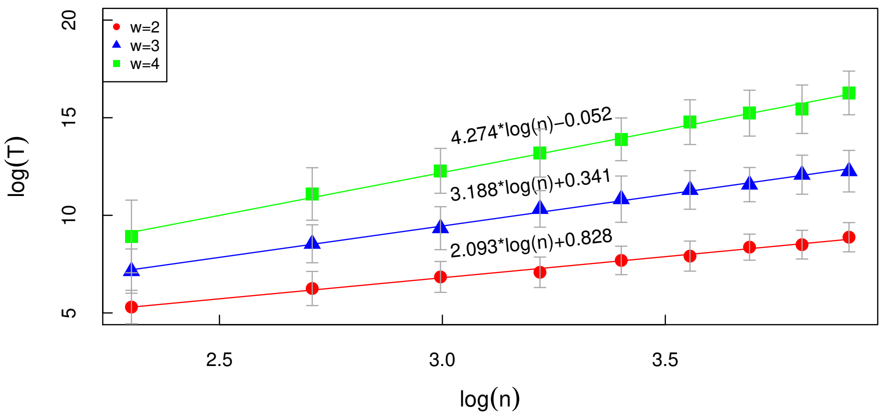

Figure 3: Empirical runtime ($\log \overline{T}$ against $\log n$) of the (1+1) EA with $p_m = 1/n$ on Hurdle. The best-fit model for each hurdle width $w \in \{2, 3, 4\}$ and the standard deviation for each data point are also plotted.

We have rigorously proven an expected runtime of $\Theta(n^w)$ for the (1+1) EA on the Hurdle problem in Theorem 5. On a logarithmic scale, a runtime of $c \cdot n^w$ (for some constant $c > 0$) becomes $w \cdot \log(n) + \log(c)$. Thus, we fit a function of $a \cdot \log(n) + b$ to the empirical data to check in particular in how far the empirical results correctly reveal the exponent $a = w$ (and also the multiplicative constant $c > 0$ satisfying $b = \log(c)$). Figure 3 plots the logarithm of the empirical runtime against the logarithm of the problem instance size n. We observe that the linear models (w.r.t. $\log(n)$) fit well to the empirical data. Especially, for the smallest hurdle width $w = 2$, the slope a is very close to the hurdle width, i.e., $a \approx 2.093$. For other values of $w \in \{3, 4\}$, the values of a are still reasonably close to the true values of 3 and 4, respectively. We believe that a better fit could be obtained by increasing the number of independent runs of the algorithm and by increasing the range of problem instance sizes to reduce the possible impact of small-order terms. This highlights the limitations of empirical research and the benefits of rigorous theorems that hold for arbitrary problem sizes and hurdle widths.

8.2. (1+1) MA

We first examine how different mutation rates affect the empirical runtime of the (1+1) MA on Hurdle. To do this, we fix $n = 50$ and set $p_m = c/n$, where $c \in \{1, 2, \ldots, 7\}$ on Hurdle with $w = 4$. For each value of c, we average the runtimes from 100 independent runs of the algorithm. The result is described in the left plot in Fig. 4. Note that we also consider the (1+1) EA and observe that the mutation rate w/n yields the best empirical runtime, which matches our analyses in Theorem 9. For the (1+1) MA with FILS the optimal mutation rate is $1/n$ because to leave a local optimum the algorithm only needs to flip a zero and the mutation rate $1/n$ maximises this probability. On the other hand, the mutation rate $2/n$ yields the best empirical runtime for the (1+1) MA with BILS since this mutation rate maximises the probability of flipping two zeros to leave a local optimum.

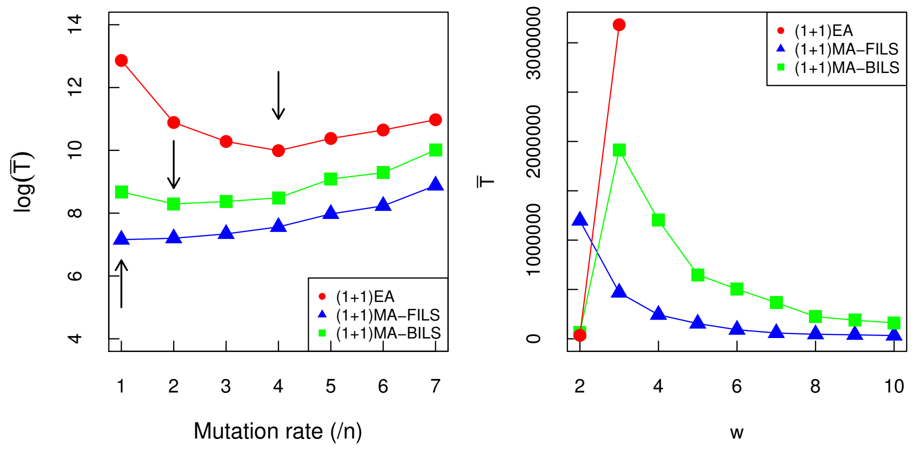

Figure 4: Empirical runtime of the (1+1) EA and (1+1) MA on Hurdle. Left: different mutation rates c/n for $c \in \{1, 2, \ldots, 7\}$, $w = 4$ and $n = 50$. The arrows show the value of the mutation rate (in the chosen range) where the empirical runtime is minimised. Right: different hurdle width $w \in \{2, 3, \ldots, 10\}$, $n = 100$ and mutation rate $1/n$.

Moreover, we discover that increasing the hurdle width makes the problem more difficult for the (1+1) EA but easier for the (1+1) MA. To verify this, we run the algorithms under different hurdle widths $w \in \{2, 3, \ldots, 10\}$, $n = 100$ and fixed mutation rate $1/n$. The empirical runtime is described in the right

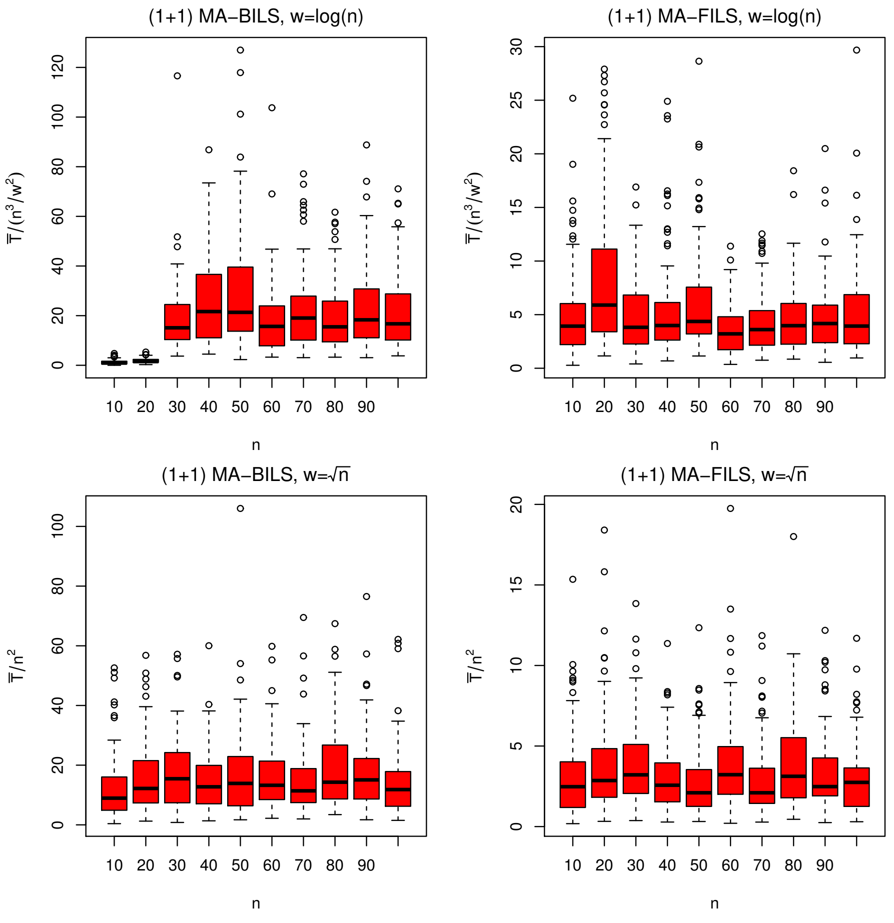

Figure 5: Boxplots of the scaled runtime of the (1+1) MA with different local search schemes and mutation rate $p_m = 1/n$ on Hurdle with $w = \log n$ (small) and $w = \sqrt{n}$ (large).

plot in Fig. 4. We observe that the empirical runtime of the (1+1) EA grows exponentially fast even for small hurdle width $w \in \{2, 3\}$[6], whereas the runtimes of the (1+1) MA with FILS and BILS decrease when w becomes larger. Note that the (1+1) MA with BILS is surprisingly efficient for $w = 2$, which is not covered by Theorem 16. The reason is that with $w = 2$, if mutation of a local optimum decreases the number of zeros by 1, BILS will run into the next local optimum (in contrast to $w \geq 3$ where it will return to the previous local optimum). The (1+1) MA with FILS is also more likely to run back into the previous local optimum if the number of zeros is small, as then all Hamming neighbours have better fitness and the first bit-flip will be accepted. If the number of zeros is small, it is likely to increase during the first step of FILS. This explains why

[6]We only plot the runtime of the (1+1) EA for $w \in \{2, 3\}$ because the runtime grows exponentially fast in the hurdle width w. For example, when $w = 4$, the average number of function evaluations of 10 independent runs of the (1+1) EA was larger than $3.9 \cdot 10^8$. Adding such a point to the graph would make it harder to visually follow the change in the runtime of the MAs for different hurdle widths.

BILS is a better choice for $w = 2$.

Finally, in Section 5, we showed that the (1+1) MA requires expected runtimes of $\Theta(n^2 + n^3/w^2)$ with BILS and $\Theta(n^3/w^2)$ with FILS on Hurdle. We are interested in the effect of the n^2 and n^3/w^2 terms. It is clear that if $w = o(\sqrt{n})$, the latter dominates the former, resulting in an expected runtime of $\Theta(n^3/w^2)$; otherwise, the runtime will be $\Theta(n^2)$. We now consider two regimes of the hurdle width: $w = \log n \in o(\sqrt{n})$ (small) and $w = \sqrt{n}$ (large). For other parameters, we use mutation rate $1/n$ and $n \in \{10, 20, \ldots, 100\}$. For each value of n, we run the algorithm 100 times independently.

We note that, for a small hurdle width $w = \log n$, Theorem 16 shows that the algorithm with either FILS or BILS takes an expected runtime of $\mathbb{E}[T] = \Theta(n^3/w^2)$. This means that $\mathbb{E}[T]/(n^3/w^2) = \Theta(1)$ for sufficiently large n. The relationship between $\overline{T}/(n^3/w^2)$ against n is described in the top two plots in Fig. 5, where we obtain horizontal lines, which match very well with the theoretical expected runtime. Furthermore, for larger hurdle width $w = \sqrt{n}$, Theorem 16 shows that the (1+1) MA with either FILS or BILS requires an expected runtime $\mathbb{E}[T] = \Theta(n^2)$ on Hurdle, which is equivalent to $\mathbb{E}[T]/n^2 = \Theta(1)$. Similarly, the relationship between $\overline{T}/n^2$ against n is also described in the two remaining plots in Fig. 5. Again, we obtain horizontal lines, well matching our theoretical results.

8.3. (2+1) MA

We also carry out experiments to examine the advantage of (uniform) crossover for the (2+1) MA. We fix $n = 100$, mutation rate of $1/n$ and run the algorithms on Hurdle instances with $w \in \{2, 3, \ldots, 10\}$. For each value of n, we run the algorithm 100 times and calculate the average runtime. The results are described in Fig. 6. The figure shows that the (2+1) MA with any local search scheme outperforms the (1+1) MA. Furthermore, the empirical runtimes for the (2+1) MA decrease as the hurdle width increases, i.e., the problem becomes easier. This matches the observation obtained from our theoretical analyses in Section 6.

9. Conclusions

We have provided a rigorous runtime analysis, comparing the simple (1+1) EA with the (1+1) MA using two local search algorithms, FILS and BILS, on the class of Hurdle problems. Our main results are tight bounds of $\Theta(n^2 + n^3/w^2)$ on the expected number of function evaluations of the (1+1) MA using BILS and $\Theta(n^3/w^2)$ for the (1+1) MA using FILS. On the other hand, the (1+1) EA and local search algorithms on their own take expected time $\Theta(n^w)$ and $2^{\Omega(n)}$, respectively. For $w = \omega(1)$ the latter times are superpolynomial, whereas the expected number of function evaluations for the (1+1) MA is always polynomial, regardless of the hurdle width w. This still holds when considering the (1+1) EA with the optimal mutation rate w/n, where the expected runtime is $\Theta(n^w \cdot (e/w)^w)$ for $2 \le w \le \sqrt{n} - 1/2$.

Another major contribution was investigating the effect of crossover and giving the first rigorous proof that crossover can have a substantial advantage over using mutation only. More specifically, the (2+1) MA optimises the Hurdle problem within expected runtimes of $\mathcal{O}\big(n^2 + (n^2 \log(n/w))/w\big)$ using BILS and

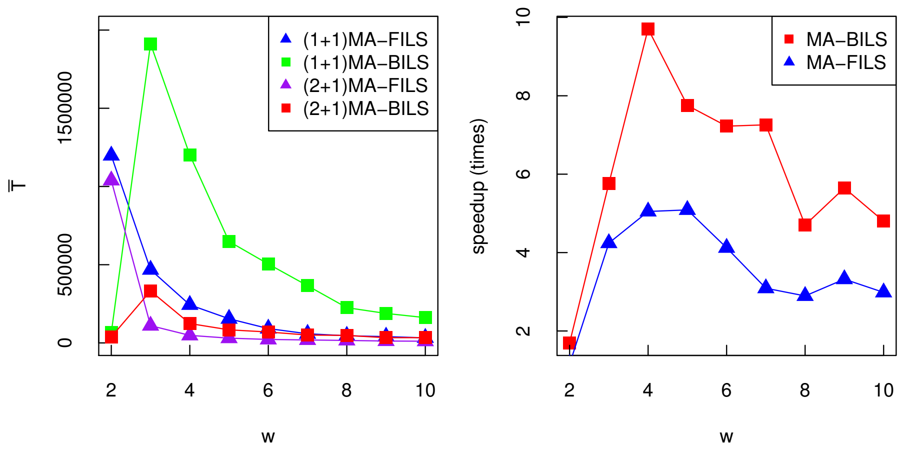

Figure 6: Empirical runtime of the (1+1) MA and the (2+1) MA with different local search schemes for mutation rate $p_m = 1/n$ and $n = 100$ on Hurdle with $w \in \{2, 3, \dots, 10\}$. The right-hand side shows the speedup of the (2+1) MA over the (1+1) MA as the ratio of the respective average runtimes.

of $\mathcal{O}\big((n^2 \log(n/w))/w\big)$ using FILS. We observe that crossover speeds up the optimisation process by replacing the term n/w on the expected runtimes of the (1+1) MA by $\log(n/w)$, yielding speedups of a factor of $\Omega(n/\log n)$ for constant hurdle widths, $w = \mathcal{O}(1)$.

A surprising conclusion is also that the Hurdle problem class becomes easier for the (1+1) MA as the hurdle width w grows. The reason is that while the (1+1) EA has to jump to a better local optimum by mutation, for the (1+1) MA it suffices to jump into the basin of attraction of a better local optimum. Increasing the hurdle width w makes it harder for the (1+1) EA to make this jump, but it also increases the size of the basin of attraction of every local optimum, effectively giving the (1+1) MA a bigger target to jump to. Crossover is useful in this setting as it increases the chances to jump into the basin of attraction of a better local optimum if the population contains a small amount of diversity.

For Hurdle we found that using mutation, crossover and local search together gives provably better runtime guarantees than using any proper subset of these operators. This demonstrates the benefit of hybridisation in memetic algorithms.

Acknowledgements

The authors thank all anonymous reviewers for their many constructive comments that helped to improve the manuscript.

References

[1] P. T. H. Nguyen, D. Sudholt, Memetic algorithms beat evolutionary algorithms on the class of hurdle problems, in: Proc. GECCO, ACM Press, 2018, pp. 1071–1078.

[2] S. Wolf, P. Merz, Evolutionary local search for the super-peer selection problem and the p-hub median problem, in: T. Bartz-Beielstein, M. J.

Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, M. Sampels (Eds.), Hybrid Metaheuristics, Springer Berlin Heidelberg, 2007, pp. 1–15.

[3] R. Dorne, J.-K. Hao, A new genetic local search algorithm for graph coloring, in: A. E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature — PPSN V, 1998, pp. 745–754.

[4] B. Freisleben, P. Merz, A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems, in: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 616–621.

[5] F. Neri, N. Kotilainen, M. Vapa, An adaptive global-local memetic algorithm to discover resources in p2p networks, in: M. Giacobini (Ed.), Applications of Evolutionary Computing, 2007, pp. 61–70.

[6] U. Benlic, J.-K. Hao, Memetic search for the quadratic assignment problem, Expert Syst. Appl. 42 (1) (2015) 584–595.

[7] Y. Mei, K. Tang, X. Yao, Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem, IEEE Trans. Evol. Comput. 15 (2) (2011) 151–165.

[8] R. Menca, M. R. Sierra, C. Menca, R. Varela, Memetic algorithms for the job shop scheduling problem with operators, Appl. Soft Comput. 34 (2015) 94–105.

[9] G. Fraser, A. Arcuri, P. McMinn, A memetic algorithm for whole test suite generation, J. Syst. Software 103 (2015) 311–327.

[10] F. Neri, C. Cotta, P. Moscato (Eds.), Handbook of Memetic Algorithms, Vol. 379 of Studies in Computational Intelligence, Springer, 2012.

[11] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: A literature review, Swarm Evol. Comput. 2 (2012) 1–14.

[12] X. Chen, Y. Ong, M. Lim, K. C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (5) (2011) 591–607.

[13] D. Sudholt, Memetic evolutionary algorithms, in: A. Auger, B. Doerr (Eds.), Theory of Randomized Search Heuristics – Foundations and Recent Developments, no. 1 in Series on Theor. Comput. Sci., World Scientific, 2011, pp. 141–169.

[14] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search, in: Handbook of Metaheuristics, Vol. 57 of International Series in Oper. Res. & Manag. Sci., Kluwer Academic Publishers, Norwell, MA, 2002, pp. 321–353.

[15] J. Lin, Y. Chen, Analysis on the collaboration between global search and local search in memetic computation, IEEE Trans. Evol. Comput. 15 (5) (2011) 608–623.

[16] G. Ochoa, N. Veerapen, Deconstructing the big valley search space hypothesis, in: Proc. EvoCOP, Springer, 2016, pp. 58–73.

[17] C. R. Reeves, Landscapes, operators and heuristic search, Ann. Oper. Res. 86 (0) (1999) 473–490.

[18] T. Friedrich, F. Neumann, What's hot in evolutionary computation, in: Conference on Artificial Intelligence (AAAI), 2017, pp. 5064–5066.

[19] A. Prügel-Bennett, When a genetic algorithm outperforms hill-climbing, Theor. Comput. Sci. 320 (1) (2004) 135 – 153.

[20] M. J. Dinneen, K. Wei, On the analysis of a (1+1) adaptive memetic algorithm, in: IEEE Workshop on Memetic Comput. (MC), 2013, pp. 24–31.

[21] D. R. Hains, D. L. Whitley, A. E. Howe, Revisiting the big valley search space structure in the TSP, J. Oper. Res. Soc. 62 (2) (2011) 305–312.

[22] P. Merz, B. Freisleben, Memetic algorithms and the fitness landscape of the graph bi-partitioning problem, in: Parallel Problem Solving From Nature V, Springer Berlin Heidelberg, 1998, pp. 765–774.

[23] J. Shi, Q. Zhang, E. Tsang, EB-GLS: an improved guided local search based on the big valley structure, Memetic Comput. 10 (3) (2018) 333–350.

[24] D. Sudholt, On the analysis of the (1+1) memetic algorithm, in: Proc. GECCO, ACM Press, 2006, pp. 493–500.

[25] D. Sudholt, The impact of parametrization in memetic evolutionary algorithms, Theor. Comput. Sci. 410 (26) (2009) 2511–2528.

[26] D. Sudholt, Local search in evolutionary algorithms: the impact of the local search frequency, in: Proc. ISAAC, Vol. 4288 of LNCS, Springer, 2006, pp. 359–368.

[27] D. Sudholt, Hybridizing evolutionary algorithms with variable-depth search to overcome local optima, Algorithmica 59 (3) (2011) 343–368.

[28] C. Witt, Analysis of an iterated local search algorithm for vertex cover in sparse random graphs, Theor. Comput. Sci. 425 (0) (2012) 117–125.

[29] D. Sudholt, C. Zarges, Analysis of an iterated local search algorithm for vertex coloring, in: Proc. ISAAC, Vol. 6506 of LNCS, Springer, 2010, pp. 340–352.

[30] K. Wei, M. J. Dinneen, Runtime analysis comparison of two fitness functions on a memetic algorithm for the clique problem, in: Proc. CEC, IEEE, 2014, pp. 133–140.

[31] K. Wei, M. J. Dinneen, Runtime analysis to compare best-improvement and first-improvement in memetic algorithms, in: Proc. GECCO, ACM Press, 2014, pp. 1439–1446.

[32] C. Gießen, Hybridizing evolutionary algorithms with opportunistic local search, in: Proc. GECCO, ACM Press, 2013, pp. 797–804.

[33] Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Transactions on Systems, Man, and Cybernetics, Part B 36 (1) (2006) 141–152.

[34] F. Alanazi, P. K. Lehre, Runtime analysis of selection hyper-heuristics with classical learning mechanisms, in: Proc. CEC, IEEE, 2014, pp. 2515–2523.

[35] A. Lissovoi, P. S. Oliveto, J. A. Warwicker, On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation, in: Proc. AAAI, AAAI Press, 2019, pp. 2322–2329.

[36] A. Lissovoi, P. S. Oliveto, J. A. Warwicker, On the runtime analysis of generalised selection hyper-heuristics for pseudo-boolean optimisation, in: Proc. GECCO, ACM Press, 2017, pp. 849–856.

[37] B. Doerr, A. Lissovoi, P. S. Oliveto, J. A. Warwicker, On the runtime analysis of selection hyper-heuristics with adaptive learning periods, in: Proc. GECCO, ACM Press, 2018, pp. 1015–1022.

[38] T. Jansen, I. Wegener, On the analysis of evolutionary algorithms—a proof that crossover really can help, Algorithmica 34 (1) (2002) 47–66.

[39] T. Kötzing, D. Sudholt, M. Theile, How crossover helps in pseudo-Boolean optimization, in: Proc. GECCO, ACM Press, 2011, pp. 989–996.

[40] D.-C. Dang, T. Friedrich, T. Kötzing, M. S. Krejca, P. K. Lehre, P. S. Oliveto, D. Sudholt, A. M. Sutton, Escaping local optima using crossover with emergent diversity, IEEE Trans. Evol. Comput. 22 (3) (2018) 484–497.

[41] D.-C. Dang, T. Friedrich, M. S. Krejca, T. Kötzing, P. K. Lehre, P. S. Oliveto, D. Sudholt, A. M. Sutton, Escaping Local Optima with Diversity-Mechanisms and Crossover, in: Proc. GECCO, ACM Press, pp. 645–652.

[42] T. Jansen, I. Wegener, Real royal road functions—where crossover provably is essential, Discrete Appl. Math. 149 (2005) 111–125.

[43] T. Storch, I. Wegener, Real royal road functions for constant population size, Theor. Comput. Sci. 320 (2004) 123–134.

[44] S. Fischer, I. Wegener, The one-dimensional Ising model: Mutation versus recombination, Theor. Comput. Sci. 344 (2–3) (2005) 208–225.

[45] D. Sudholt, Crossover is provably essential for the Ising model on trees, in: Proc. GECCO, ACM Press, 2005, pp. 1161–1167.

[46] B. Doerr, E. Happ, C. Klein, Crossover can provably be useful in evolutionary computation, Theor. Comput. Sci. 425 (0) (2012) 17–33.

[47] B. Doerr, D. Johannsen, T. Ktzing, F. Neumann, M. Theile, More effective crossover operators for the all-pairs shortest path problem, Theoretical Computer Science 471 (2013) 12–26.

[48] D. Sudholt, Crossover speeds up building-block assembly, in: Proc. GECCO, ACM Press, 2012, pp. 689–696.

[49] D. Sudholt, How crossover speeds up building-block assembly in genetic algorithms, Evol. Comput. 25 (2) 237–274.

[50] B. Doerr, C. Doerr, F. Ebel, From Black-Box Complexity to Designing New Genetic Algorithms, Theor. Comput. Sci. 567 (0) (2015) 87–104.

[51] D. Corus, P. S. Oliveto, Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms, IEEE Trans. Evol. Comput. 22 (5) (2018) 720–732.

[52] E. C. Pinto, C. Doerr, A simple proof for the usefulness of crossover in black-box optimization, in: A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, D. Whitley (Eds.), Parallel Problem Solving From Nature, Vol. 11102 of LNCS, Springer, 2018, pp. 29–41.

[53] J. Lengler, A general dichotomy of evolutionary algorithms on monotone functions, IEEE Trans. Evol. Comput. To appear.

[54] J. Lengler, X. Zou, Exponential slowdown for larger populations: The (μ+1)-EA on monotone functions, in: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA '19), ACM, 2019, pp. 87–101.

[55] F. Neumann, P. S. Oliveto, G. Rudolph, D. Sudholt, On the effectiveness of crossover for migration in parallel evolutionary algorithms, in: Proc. GECCO, ACM Press, 2011, pp. 1587–1594.

[56] D. Sudholt, C. Thyssen, Running time analysis of ant colony optimization for shortest path problems, J. Discrete Algor. 10 (2012) 165–180.

[57] A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing, 2nd Edition, Springer, 2015.

[58] D. Sudholt, On the robustness of evolutionary algorithms to noise: Refined results and an example where noise helps, in: Proc. GECCO, ACM Press, 2018, pp. 1523–1530.

[59] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.

[60] I. Wegener, Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions, in: R. Sarker, X. Yao, M. Mohammadian (Eds.), Evol. Optim., Kluwer, 2002, pp. 349–369.

[61] R. Kaas, J. Buhrman, Mean, median and mode in binomial distributions, Statistica Neerlandica 34 (1) (1980) 13–18.

[62] T. Paixão, J. Pérez Heredia, D. Sudholt, B. Trubenová, Towards a runtime comparison of natural and artificial evolution, Algorithmica 78 (2) (2017) 681–713.

[63] B. Doerr, H. P. Le, R. Makhmara, T. D. Nguyen, Fast genetic algorithms, in: Proc. GECCO, ACM Press, 2017, pp. 777–784.

[64] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, The MIT Press, 2009.

[65] K. M. Burjorjee, Hypomixability elimination in evolutionary systems, in: Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms (FOGA '15), ACM, 2015, pp. 163–175.

[66] A. Livnat, C. Papadimitriou, Sex as an algorithm: The theory of evolution under the lens of computation, Commun. ACM 59 (11) (2016) 84–93.

[67] A. Prügel-Bennett, J. Rowe, J. Shapiro, Run-time analysis of population-based evolutionary algorithm in noisy environments, in: Proceedings of the 2015 ACM Press Conference on Foundations of Genetic Algorithms (FOGA 2015), 2015, pp. 69–75.