
White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Sheffield.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/130705/

Version: Accepted Version

Proceedings Paper:
Nguyen, P.T.H. and Sudholt, D. (2018) Memetic Algorithms Beat Evolutionary Algorithms on the Class of Hurdle Problems. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2018). Genetic and Evolutionary Computation Conference (GECCO 2018), 15-19 Jul 2018, Kyoto, Japan. ACM. ISBN: 978-1-4503-5618-3.

https://doi.org/10.1145/3205455.3205456

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.



Memetic Algorithms Beat Evolutionary Algorithms on the Class of Hurdle Problems *

Phan Trung Hai Nguyen
School of Computer Science
University of Birmingham
p.nguyen@cs.bham.ac.uk

Dirk Sudholt
Department of Computer Science
University of Sheffield
d.sudholt@sheffield.ac.uk

April 17, 2018

Abstract

Memetic algorithms are popular hybrid search heuristics that integrate local search into the search process of an evolutionary algorithm in order to combine the advantages of rapid exploitation and global optimisation. However, these algorithms are not well understood and the field is lacking a solid theoretical foundation that explains when and why memetic algorithms are effective.

We provide a rigorous runtime analysis of a simple memetic algorithm, the (1+1) MA, on the HURDLE problem class, a landscape class of tuneable difficulty that shows a "big valley structure", a characteristic feature of many hard problems from combinatorial optimisation. The only parameter of this class is the hurdle width w, which describes the length of fitness valleys that have to be overcome. We show that the (1+1) EA requires $\Theta(n^w)$ expected function evaluations to find the optimum, whereas the (1+1) MA with best-improvement and first-improvement local search can find the optimum in $\Theta(n^2 + n^3/w^2)$ and $\Theta(n^3/w^2)$ function evaluations, respectively. Surprisingly, while increasing the hurdle width makes the problem harder for evolutionary algorithms, the problem becomes easier for memetic algorithms. We discuss how these findings can explain and illustrate the success of memetic algorithms for problems with big valley structures.

Index terms— Evolutionary algorithms, hybridisation, iterated local search, local search, memetic algorithms, running time analysis, theory

*Preliminary version of this work will appear in the Proceedings of the 2018 Genetic and Evolutionary Computation Conference (GECCO 2018)

1 Introduction

1.1 Motivation

Memetic Algorithms (MAs), also known as evolutionary/genetic local search or global-local search hybrids, are hybrid stochastic search methods that incorporate one or more intensifying local search algorithms into an evolutionary framework. The motivation behind this hybridisation is to create a new algorithm that combines the exploration capabilities of an evolutionary algorithm with the efficiency and exploitation capabilities of local search. There are many examples where this strategy has proven effective; see e. g. [11, 10].

In [21], three advantages for memetic algorithms are pointed out:

1. Local search can quickly find solutions of high quality due to its rapid exploitation.

2. Selection is only performed after local search has had a chance to improve on new offspring; this is beneficial for low-fitness offspring located in the basin of attraction of a high-fitness local optimum, as in a conventional evolutionary algorithm such low-fitness offspring would be removed by selection. This effect is particularly visible for constrained problems, where penalties are used for violated constraints, and local search can act as a repair mechanism [21].

3. Local search can include problem-specific knowledge; this is often possible since local search strategies are typically easy to design, even when it is hard to design a global problem-specific strategy [21].

A challenge when dealing with memetic algorithms and hybrid algorithms, in general, is that the search dynamics can be very hard to understand, in particular due to the interplay of different operators. It is not well understood when and why memetic algorithms perform well, when they do not, and how to design memetic algorithms most effectively for a problem in hand. Most work in this area is empirical and the theory of memetic algorithms is still in its infancy.

There are many different variants of memetic algorithms, from algorithms that only rarely apply local search, with a fixed *local search frequency* to *iterated local search* algorithms where local search is applied in every generation [7]. In the latter scenario, local search turns all search points into local optima, and evolution acts on the sub-space of local optima. The hope is that mutation can lead a memetic algorithm to leave its current local optimum, and to reach the basin of attraction of a better one.

We demonstrate that this strategy works very effectively on a class of problems introduced by Prügel-Bennett [14] 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[1] with an underlying gradient leading towards the global optimum and a number of "hurdles" that have to be overcome. These hurdles consist of a local optimum and a fitness valley that has to be overcome to reach the next local optimum. The distance between local optima is a parameter w called the *hurdle width*, and it can be used to parameterise the width of the fitness valleys. For simple evolutionary algorithms like the (1+1) EA, a larger hurdle width makes the problem harder. This effect was analysed in [14] with non-rigorous arguments based on simplifying assumptions, that led to approximations of the expected time for finding the global optimum.

Here we provide a rigorous analysis for the expected optimisation time of the (1+1) EA: we give a tight bound of $\Theta(n^w)$ for the expected optimisation time[2], confirming that the performance degrades very rapidly with increasing hurdle width. For hurdle widths growing with n, $w = \omega(1)$, this expected time is superpolynomial and hence intractable.

In contrast, we show that memetic algorithms perform very effectively on this problem class due to their combination of evolutionary operators and local search. We study a simple iterated local search algorithm called (1+1) MA with two different local searches, *First-Improvement Local Search* (FILS) and *Best-Improvement Local Search* (BILS) [3], and 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. These times are polynomial for all choices of the hurdle width.

Note that the term n^3/w^2 decreases with the hurdle width, hence the surprising conclusion is that larger hurdle widths make the problem much harder for evolutionary algorithms, while making the problem easier for memetic algorithms.

The HURDLE problem, albeit having been defined for a very different purpose [14], turns out to be an ideal example for showcasing the power of memetic algorithms and iterated local search. This finding is particularly significant in the light of "big valley" structures, an important characteristic of many hard problems from combinatorial optimisation [12, 15], 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" [5]. The HURDLE problem is a perfect and very illustrative example of a big valley landscape. By explaining how the (1+1) MA easily solves the HURDLE problem class, we hope to gain insight into how memetic algorithms perform on big valley structures, which may help to explain why state-of-the-art memetic algorithms perform well on problems with big valley structures [8, 16].

[1] A function of unitation is a function that only depends on the number of ones.

[2] See [2, Chapter 3] for a definition of asymptotic notation and symbols $\Theta, O, o, \Omega, \omega$.

1.2 Related Work

There are other examples of functions where memetic algorithms were theoretically proven to perform well (see Sudholt [21] for a more extensive survey). In [18] 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 [19] 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 evolutionary algorithms 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 [17].

Sudholt [20] showed for instances of classical problems from combinatorial optimisation that memetic algorithms with a different kind of local search, *variable-depth local search*, can efficiently cross huge fitness valleys that are nearly impossible to cross with evolutionary algorithms. Witt [25] further analysed the performance of a memetic algorithm, iterated local search, for the Vertex Cover problem. Sudholt and Zarges [22] investigated the use of memetic algorithms for the graph colouring problem. Finally, Wei and Dineen analysed memetic algorithms for solving the Clique problem, investigating the choice of the fitness function [23] as well as the choice of the local search operator [24].

Gießen [4] presented another example function class based on a discretised version of the well-known Rastrigin function. He designed a memetic algorithm 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.

Another line of research is work on *hyperheuristics* that combine different operators. Alanazi and Lehre [1] demonstrated the usefulness of hyperheuristics for artificial functions, and Lissovoi, Oliveto, and Warwicker [6] presented novel, provably efficient hyperheuristic algorithms. The difference to memetic algorithms is that while hyperheuristics typically apply one operator, while learning which operator performs best, memetic algorithms apply different operators, variation and local search, in sequence. The interplay of variation and local search is a major challenge when analysing memetic algorithms.

1.3 Outline

The paper is structured as follows. Section 2 introduces the (1+1) EA, (1+1) MA as well as the two local searches. The class of Hurdle problems

are then formally defined in Section 3, which also includes detailed description about their properties. Section 4 points out the inefficiency of the (1+1) EA and two local search algorithms by investigating their expected optimisation times on the HURDLE problems. Next, tight bounds on the expected optimisation time of the (1+1) MA are derived in Section 5, which reveals the outperformance of the (1+1) MA to the alternative stochastic search methods. Finally, concluding remarks are given in Section 6.

2 Preliminaries

2.1 (1+1) Evolutionary Algorithm

In order to focus on the main differences between evolutionary algorithms and memetic algorithms, and to facilitate a rigorous theoretical analysis, we consider simple bare-bones algorithms from these two paradigms.

The (1+1) EA is the simplest evolutionary algorithm, operating with a population of size one and using only mutation. The mutation operator flips each bit independently with mutation probability p_m, with the default choice being $p_m = 1/n$, where n is the length of the bitstring. The fitness function is defined as $f\colon \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} = \{0,1\}^n$ is the binary search space, and has to be maximised. Algorithm 1 gives a full description of the (1+1) EA. Here MUTATE(x) returns a new bitstring resulting from flipping bits in x independently with probability p_m.

Algorithm 1: (1+1) EA

$x \sim \text{Unif}\{\mathcal{X}\}$
repeat
  $y \leftarrow \text{MUTATE}(x, p_m)$
  if $f(y) \geq f(x)$ then
    $x \leftarrow y$
until *some stopping condition is fulfilled.*

Practical implementations of Evolutionary Algorithms in particular and other search metaheuristics in general require to specify some stopping condition. The simplest is to stop when a fixed number of generations has been exceeded. The theoretical results presented in this paper address the limiting case when the algorithm runs until a global optimum is found. In this case, we are interested in the *expected optimisation time* of the algorithm, defined as the mean of the number of fitness (or function) evaluations performed by the algorithm until a global optimum is found.

2.2 (1+1) Memetic Algorithm

Algorithm 2 [24] outlines the typical procedure of the (1+1) MA, the simplest memetic algorithm. The algorithm consists of a population of one individual and produces an offspring in each generation by independently flipping each bit in the current search point with mutation probability $p_m = 1/n$. The newly generated offspring is then refined further using a LocalSearch. Any local search algorithms can fit into the scenario. Although the (1+1) MA looks quite simple, it still captures the same working principle as the general MAs. Analysing it can reveal insights into how the general MAs operate and when they can be employed to solve problems.

Algorithm 2: (1+1) MA

$x \sim \text{Unif}\{\mathcal{X}\}$
repeat
  $y \leftarrow \text{Mutate}(x, 1/n)$
  $z \leftarrow \text{LocalSearch}(y)$
  if $f(z) \geq f(x)$ then
    $x \leftarrow z$
until *some stopping condition is fulfilled.*

2.3 Local Searches

We consider the following two local searches in the context of the (1+1) MA. Both local searches are common practice and have also been analysed in [3].

Algorithm 3: FILS

input: a bitstring $x = (x_1, \ldots, x_n) \in \{0,1\}^n$
for δ iterations do
  create a random permutation Per of set $\{1, 2, \ldots, n\}$
  BetterFound $\leftarrow$ *false*
  for $i = 1$ to n do
    $y \leftarrow \text{Flip}\,(x, \text{Per}[i])$
    if $f(y) > f(x)$ then
      $x \leftarrow y$
      BetterFound $\leftarrow$ *true*
  if BetterFound = *false* then
    return x
return x

2.3.1 First Improvement Local Search

(FILS), shown in Algorithm 3, adapted from Wei and Dinneen [3], takes advantage of the first improvement it finds while searching the neighbourhood. The algorithm runs for δ iterations. Bits are flipped according to a random permutation Per of length n (to avoid any search bias due to the choice of bit positions), and newly generated individuals are then scored by the fitness function. Here $\textsc{Flip}(x, i)$ returns a new bitstring resulting from flipping the i-th bit in x. The current search point is replaced by the first neighbour found with a better fitness value. The algorithm stops either after δ iterations of the outer for loop have been performed or after visiting all n neighbours of the current search point without any improvement.

2.3.2 Best Improvement Local Search

(BILS), shown in Algorithm 4, adapted from Wei and Dinneen [3], searches the whole neighbourhood and then picks a search point giving the best improvement.

The algorithm runs for δ iterations, and in each iteration a neighbour with the largest improvement in the fitness among all n neighbours of the current search point is picked to be the next search point. In order to keep track of the progress so far, it stores the best neighbour(s) and best fitness into CurBestInds and CurBestFit, respectively. This means that whenever a neighbour with better fitness compared to CurBestFit has been found, the algorithm performs update on the two variables. At the end of an iteration, if there are more than one neighbours with the same fitness value that is better than $f(x)$, then the next search point is chosen uniformly at random from the set CurBestInds.

3 Class of Hurdle Problems

The Hurdle function class was introduced back in 2004 by Prügel-Bennett [14] as an example class where genetic algorithms with crossover outperform hill climbers. Here we give a formal definition and discuss basic properties of the function that will be used in the subsequent analyses.

The objective is to find a bitstring that maximises the fitness function $f\colon \mathcal{X} \to \mathbb{R}$. The value of the fitness function at a given bitstring $x \in \mathcal{X}$ is [14]

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

In this function, $z(x)$ is the number of zeros in the bitstring x. $w \in \{2, 3, \ldots, n\}$ is called the hurdle width and is the only parameter of the Hurdle problems. Note that $w = w(n)$ may be a function of n. Finally,

Algorithm 4: BILS

input: a bitstring $x = (x_1, \ldots, x_n) \in \{0,1\}^n$
for δ iterations do
  CurBestInds $\leftarrow \emptyset$
  CurBestFit $\leftarrow f(x)$
  for $i = 1$ to n do
    $y \leftarrow$ Flip(x, i)
    if $f(y) >$ CurBestFit then
      CurBestInds $\leftarrow \{y\}$
      CurBestFit $\leftarrow f(y)$
    else if $f(y) =$ CurBestFit then
      CurBestInds $\leftarrow$ CurBestInds $\cup \{y\}$
  if CurBestInds $= \emptyset$ then
    return x
  else
    $x \sim$ Unif{CurBestInds}
return x

$\mathrm{rem}(z(x), w)$ is the remainder of $z(x)$ divided by w, while $\lceil \cdot \rceil$ is the ceiling function.

Lemma 1. *The global optimum for the* Hurdle *problem is* 1^n.

Proof. For every $x \in \{0,1\}^n$, $f(x) \leq 0$ since both $z(x)$ and $\mathrm{rem}(z(x), w)$ cannot be negative. The equality happens if and only if both $z(x)$ and $\mathrm{rem}(z(x), w)$ equal zero, or equivalently $x = 1^n$. □

Note in particular that $z(x)$ can also be viewed as the Hamming distance $H(x, 1^n)$ between the current solution x and the global optimum 1^n. The

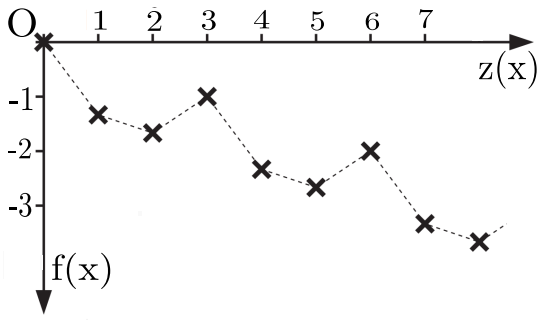

Figure 1: Fitness landscapes of Hurdle with $w = 3$.

fitness landscapes close to the global optimum are shown in Fig. 1 [14]. It can be clearly seen that the global optimum 1^n coincides with the origin where both $z(1^n) = 0$ and $f(1^n) = 0$. In the following lemma, the term *nearest* refers to the scale of $z(x)$, i. e. the most similar number of zeros. Note that this relates to Hamming distances as follows: any search point with $z(x)$ zeros has Hamming distance at least $|i - z(x)|$ to any search point with i zeros. A sufficient condition for a mutation of x having i zeros is flipping $i - z(x)$ zeros and no other bits if $i \geq z(x)$ or flipping $z(x) - i$ ones if $i \leq z(x)$.

Lemma 2. *Given a* HURDLE *problem with hurdle width w and a local optimum $x \neq 1^n$ as the current search point, the nearest search points with fitness larger than $f(x)$ are all search points with $z(x) - w$ zeros: $\{x' \mid z(x') = z(x) - w\}$.*

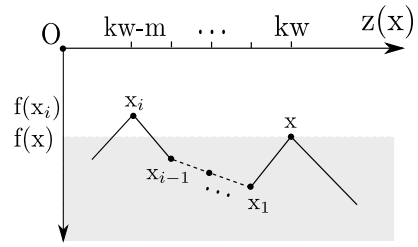

Figure 2: Fitness landscape of HURDLE problem with arbitrary w.

Proof. The current local optimum $x \neq 1^n$contains $z(x) = kw$ zeros where $k \in \mathbb{N} \setminus \{0\}$ (see Fig. 2). Let us consider a search point x_i which is the nearest search point with $f(x_i) \geq f(x)$ and $z(x_i) = kw - m < z(x)$ where $0 < m \leq w$. Here, we exclude the case $m > w$ as the next local optimum corresponds to $m = w$, and its fitness value is already known to be better than $f(x)$.

Now we need to calculate the fitness values for two search points, x and x_i. Note that $z(x)/w = kw/w = k$, and $\mathrm{rem}(z(x), w) = \mathrm{rem}(kw, w) = 0$, then

$$f(x) = -\left\lceil \frac{z(x)}{w} \right\rceil - \frac{\mathrm{rem}(z(x), w)}{w} = -\lceil k \rceil - \frac{0}{w} = -k.$$

On the other hand, $z(x_i)/w = (kw-m)/w = k-m/w$, and $\mathrm{rem}(z(x_i), w) = \mathrm{rem}(kw - m, w) = w - m$ as we can rewrite $z(x_i) = kw - m = (k-1)w + (w - m)$, then

$$f(x_i) = -\left\lceil k - \frac{m}{w} \right\rceil - \frac{w - m}{w} = -\left\lceil k - \frac{m}{w} \right\rceil - 1 + \frac{m}{w}.$$

Now we consider two different cases as follows. If $m = w$, then $m/w = 1$ and $f(x_i) = -(k-1) - 1 + 1 = -k + 1 > f(x)$; otherwise, $m/w < 1$, and $1 - \frac{m}{w} > 0$, then $f(x_i) = -k - \left(1 - \frac{m}{w}\right) < f(x)$.

For all $0 < m \leq w$, we only have $f(x_i) > f(x)$ if and only if $m = w$, and then $z(x_i) = (k-1)w$ where $k \in \mathbb{N} \setminus \{0\}$. This result implies that x_i must be a local optimum. This proof also shows that the difference in the fitness values of two consecutive local optima is exactly one. □

4 Why Is Hybridisation Necessary?

4.1 Local Searches

In this section, we show that local search algorithms in general are unable to optimise the HURDLE problems, unless the initial search point is chosen from a specific regions in the fitness landscape.

Let z denote the number of zeros in the initial search point. It is obvious that if $z \geq w$, then the local search algorithm cannot locate the global optimum as it gets stuck at a local one forever. Otherwise, the global optimum can be found with some probability. However, if the local search is allowed to run only once, then for $w \ll n/2$ the chance that it can optimise the HURDLE problems is close to zero since the number of search points with at most $w-1$ zeros is significantly smaller compared to the size of the search space, i.e. 2^n. 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.

Theorem 1. *The expected optimisation time 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 $2^{\Omega(n)}$.*

We focus on $w \leq cn$ for some constant $c < 1/2$ as, otherwise, the majority of search points would lie in the basin of attraction of the global optimum, resembling the function ONEMAX[3].

Proof of Theorem 1. 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 $cn - 1$ zeros in order for the algorithm to be able to optimise the problem (see Fig. 1). By Chernoff bounds [9], this event happens with probability at most $2^{-\Omega(n)}$. The expected number of restarts until this event happens is at least $2^{\Omega(n)}$. Since every restart clearly leads to at least one function evaluation, the expected optimisation time of local search algorithms on HURDLE problems is $2^{\Omega(n)}$. □

[3]The ones-counting problem, i.e. $\text{ONEMAX}(x) := \sum_i x_i$.

4.2 (1+1) Evolutionary Algorithm

In this section, we prove a tight bound $\Theta(n^w)$ on the expected optimisation time 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. Our rigorous analysis complements the non-rigorous arguments given in [14]. Note in particular that starting from an initial search point with at least w zeros, the first generation will end in a local optimum after at most w iterations of the local search with $\delta \geq w$. Since this has a small *additive* contribution to the overall runtime, we can assume that a local optimum is the current search point.

Theorem 2. *The expected optimisation time of the (1+1) EA on the* HURDLE *problem with hurdle width w is $\mathcal{O}(n^w)$.*

Proof. Assume that a local optimum x be the current search point with z zeros. Lemma 2 yields that the nearest search points with a better fitness than $f(x)$ are all local optima with $z - w$ zeros. For such a mutation we just need to flip at least w zeros simultaneously, and keep $n - w$ remaining bits unchanged. The probability of flipping w bits is given by $(1/n)^w$, and keeping $n - w$ bits unchanged is $(1 - 1/n)^{n-w}$. Therefore, the probability of obtaining a better solution is bounded from below by $p_z \geq \binom{z}{w} \left(\frac{1}{n}\right)^w \left(1 - \frac{1}{n}\right)^{n-w} \geq \frac{1}{en^w} \left(\frac{z}{w}\right)^w$ since $\left(1 - \frac{1}{n}\right)^{n-1} \geq 1/e$ and $\binom{z}{w} \geq (z/w)^w$ [2, Appendix C]. The expected number of generations until the global optimum found is now bounded from above by

$$\mathbb{E}[T] \leq \sum_{z=w}^{n} \frac{1}{p_z} \leq en^w w^w \sum_{z=w}^{n} \frac{1}{z^w}. \tag{1}$$

Now we need to calculate the sum $\sum_{z=w}^{n} \frac{1}{z^w}$. Let us consider the function $g(z) = z^{-w}$, $\forall z \in [w, n]$. Since $\nabla_z g = -w/z^{w+1} < 0$, g is a monotonically decreasing function in $[w, n]$. So we can approximate the upper bound of this sum using a method called *approximation by integrals*, i.e. $\sum_{x=a}^{b} g(x) \leq \int_{a-1}^{b} g(x)dx$ [2, Appendix A]. Applying this method yields

$$\begin{aligned}
\sum_{z=w}^{n} \frac{1}{z^w} &= w^{-w} + \sum_{z=w+1}^{n} \frac{1}{z^w} \\
&\leq w^{-w} + \int_{w}^{n} \frac{1}{z^w} dz \\
&\leq w^{-w} + \int_{w}^{\infty} \frac{1}{z^w} dz = w^{-w} + w^{-w} \cdot \frac{w}{w-1} \leq 3w^{-w}.
\end{aligned}$$

Substituting this into (1), we get $\mathbb{E}[T] \leq 3en^w$. □

To derive the lower bound, we again focus on $w \leq n/2$ for the same line of arguments as in Theorem 1.

Theorem 3. *The expected optimisation time of the (1+1)* EA *on* HURDLE *problems with hurdle width* $2 \leq w \leq n/2$ *is* $\Omega(n^w)$.

Proof. We first show that a search point with w zeros is reached with probability $\Omega(1)$. The initial number of zeros follows a binomial distribution with parameters n and $1/2$. As $w \leq n/2$, the probability that the initial search point will have at least w zeros is at least $1/2$ (by symmetry of the binomial distribution). It may be possible to "jump over" search points with w zeros, i.e. to make a transition from $i > w$ zeros to $j < w$ zeros, so long as the HURDLE value with i zeros is less or equal to that of j zeros. However, Lemma 9 in [13] states that the conditional probability of standard bit mutation reaching a search point with w zeros, given that a search point with *at most* w zeros is reached, is at least $1/2$.

Once the (1+1) EA has reached such a search point with w zeros, the expected remaining optimisation time is $n^w(1-1/n)^{-n+w} = \Omega(n^w)$ as the optimum is the only search point with a higher fitness (see Lemma 2) and the probability of jumping to the optimum from any such search point is $n^{-w}(1-1/n)^{n-w}$. By the law of total expectation, the expected optimisation time is at least $\Omega(1) \cdot \Omega(n^w) = \Omega(n^w)$. □

We remark that the (1+1) EA can be slightly sped up by increasing the mutation rate. As the above analysis has shown, the expected optimisation time is dominated by the time to locate the global optimum from a search point with Hamming distance w to it. From this starting point, a mutation rate of w/n maximises the probability of flipping exactly w bits. In fact, this choice was already used in [14] for the (1+1) EA. However, even choosing an optimal mutation rate does not help much as, even if a mutation does flip exactly w bits, the mutation needs to select the right bits to flip, and the chance of choosing exactly the w bits that differ from the optimum is $1/\binom{n}{w}$, leading to an expected time of at least $\binom{n}{w} \geq n^w/w^w$ [2] from such a local optimum (and $\Omega(\binom{n}{w}) \geq \Omega(n^w/w^w)$ from random initialisation). This still results in a superpolynomial expected time if $w = \omega(1)$, that is, w grows with n.

5 Memetic Algorithms Are Efficient

We now show that, in contrast to local search on its own, and evolutionary algorithms, the (1+1) MA can find the global optimum efficiently, for both BILS and FILS. The main result of this section is as follows. 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.

Theorem 4. *The expected number of function evaluations of the (1+1) MA on* HURDLE *with any hurdle width* $2 \leq w \leq n/2$ *is* $\Theta(n^2 + n^3/w^2)$ *for* BILS *and* $\Theta(n^3/w^2)$ *for* FILS*, both using* $\delta \geq w$.

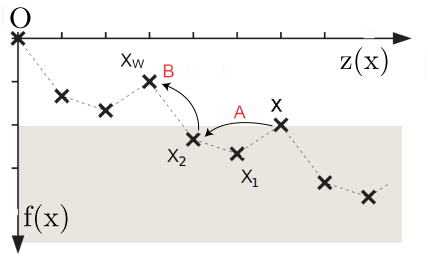

Figure 3: (1+1) MA on HURDLE problems

In order to prove Theorem 4, we first prove upper bounds of $\mathcal{O}(n^2 + n^3/w^2)$ and $\mathcal{O}(n^3/w^2)$, respectively, and then we prove lower bounds of $\Omega(n^2 + n^3/w^2)$ and $\Omega(n^3/w^2)$, respectively, showing that the upper bounds are asymptotically tight.

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

Theorem 5. *The expected number of generations 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* $\mathcal{O}(n^2/w^2)$.

Proof. Assume a local optimum x is the current search point with z zeros in the bitstring. The fitness landscape around x is illustrated in Fig. 3. Lemma 2 yields that the nearest search points with a better fitness than $f(x)$ are the local optima with exactly $z - w$ zeros. Let us consider the situation when the (1+1) MA flips at least two zeros to move from x to x_2 (say event A), and then any local search will locate the next local optimum x_w by performing a sequence of one-step jumping: x_2 to x_3,..., x_{w-1} to x_w (say event B). Clearly, event A happens with probability

$$p(A) = \binom{z}{2}\left(\frac{1}{n}\right)^2\left(1-\frac{1}{n}\right)^{n-2} \geq \frac{z(z-1)}{2en(n-1)}.$$

Given event A, event B occurs with unity probability, i. e. $p(B \mid A) = 1$, as the local search always locates the next local optimum. Hence, the probability of reaching x_w from x is bounded from below by $p(A)$, and the expected number of generations is at most $1/p(A)$. The expected number of generations until the global optimum is found is

$$\mathbb{E}[T] \leq \sum_{z \in \{w, 2w, \ldots\}} \frac{2en(n-1)}{z(z-1)}. \tag{2}$$

We have, using $\sum_{i=1}^{\infty} \frac{1}{i^2} = \pi^2/6$,

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

Substituting into (2) yields $\mathbb{E}[T] = \mathcal{O}(n^2/w^2)$. □

In order 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, and those that don't. This is an example of the *accounting method* [2, 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.

Lemma 3. *Call a local search* improving *if it terminates with a search point that has a strictly better fitness than the current search point of the (1+1) MA; otherwise, the local search is called* non-improving. *The number of function evaluations spent in any improving local search call on* Hurdle *with hurdle width w and $\delta \geq w$ is at most wn for* BILS *and at most $2n$ for* FILS.

Proof. As Hurdle is a function of unitation, while no local optimum is found, local search will either decrease the number of ones in each iteration, or increase the number of ones in every iteration. In both cases a local optimum will be found after at most $w-1$ iterations. Once a local optimum has been reached, at most n further evaluations are needed before local search stops.

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 3) a local optimum will be found, and then n further evaluations are needed before it stops. □

The expected number of function evaluations for non-improving local searches can be bounded as follows.

Lemma 4. *In the setting of Theorem 4, 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 ones 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 ones into one with j ones 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

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

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 $(2e + 1)n$. □

Putting the previous results together, we are now prepared to prove the upper bounds claimed in Theorem 4.

Proof of the upper bounds from Theorem 4. By Theorem 5 the expected number of generations 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 4. Together, the number of function evaluations in all non-improving local searches is at most $\mathcal{O}(n^3/w^2)$.

By Lemma 3, the number of function evaluations in any improving local search is at most wn for BILS and at most $2n \leq wn$ for FILS. Since every improving local search ends in a local optimum with a 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 $\mathcal{O}(n^2)$, and the overall number of function evaluations is bounded by $\mathcal{O}(n^2 + n^3/w^2)$. □

To prove the lower bounds from Theorem 4, 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 6. *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 $n/12$ generations 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 generation leads to at least one iteration of BILS.

In order for none of these events to occur, the (1+1) MA must find the optimum within $n/12$ generations, 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 $n/12$ generations, using fewer than $n/6$ iterations of BILS.

The reason is that, if none of the events E_1 and E_2 occur, then this implies $A \cup B$. By contraposition, $\overline{A \cup B} \Rightarrow E_1 \cup E_2$ and $\text{Prob}\left(\overline{A \cup B}\right) \leq \text{Prob}\left(E_1 \cup E_2\right) \leq \text{Prob}\left(E_1\right) + \text{Prob}\left(E_2\right)$ by the union bound.

Event E_1 has probability $\text{Prob}\left(E_1\right) \leq 2^{-\Omega(n)}$ 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 iid indicator variables $X_{i,t} \in \{0, 1\}$ that describe whether the i-th bit is flipped during generation t or not. Hence $\text{Prob}\left(E_2\right) \leq 2^{-\Omega(n)}$.

Together, we have by the union bound,

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

This completes the proof. □

Proof of the lower bounds from Theorem 4. A bound of $\Omega(n^2)$ for the (1+1) MA with BILS follows from Theorem 6.

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. Theorem 3 reveals 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. 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 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 $\binom{w}{2} \cdot 1/n^2 = \mathcal{O}(w^2/n^2)$, and the expected number of generations 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 $w-2$ ones 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 a generation creating the global optimum is still $\mathcal{O}(w^2/n^2)$, and the expected number of generations is still at least $\Omega(n^2/w^2)$.

In every generation, 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. □

6 Conclusions and Future Work

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 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.

The Hurdle problem hence represents an illustrative problem where

a hybrid algorithm drastically outperforms both of the individual search algorithms it contains, when these are run on their own.

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 the global optimum by mutation, for the (1+1) MA it suffices to jump into the basin of attraction of the global 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 the global optimum, effectively giving the (1+1) MA a bigger target to jump to.

More specifically, our analysis has shown that the (1+1) MA can efficiently reach a better local optimum by flipping two 0-bits during mutation, as then the resulting mutant is located in the basin of attraction of a better local optimum. The expected optimisation time is dominated by the time spent in the last local optimum, that is, when the current search point contains w zeros. From here, a mutation flipping two 0-bits has probability $\Theta(w^2/n^2)$, where the factor of w^2 results from $\binom{w}{2}$ choices for the two flipping 0-bits. The larger the hurdle width, the larger the probability of making such a mutation, and the lower the term of order n^3/w^2 becomes. Note that the (1+1) MA is otherwise agnostic to the width of the fitness valley, as local search will efficiently locate a better local optimum, regardless of the distance between local optima. This is in sharp contrast to the (1+1) EA, which has to flip exactly w bits to jump to the optimum, leading to an expected time of $\Theta(n^w)$.

Amongst problems with a "big valley" structure, HURDLE has a favourable landscape for the (1+1) MA as local optima have a very small Hamming distance to search points in the basin of attraction of better local optima. This makes it easy to transition from one local optimum to another by mutation and local search. A promising avenue for future work would be to analyse the performance of the (1+1) MA for other classes of problems with big valley structures where larger jumps need to be made to transition to better local optima. This may require increasing the mutation rate, as commonly done in iterated local search algorithms [7].

Another avenue for future work could be to rigorously analyse the expected running time of genetic algorithms with crossover on the HURDLE problem class, to investigate how their performance compares against that of the (1+1) MA. Experimental results in [14] suggest that crossover provides a substantial advantage over the (1+1) EA, however no rigorous analysis has been done.

References

[1] Alanazi, F. and Lehre, P. K. [2014], Runtime analysis of selection hyper-heuristics with classical learning mechanisms, *in* 'IEEE Congress on Evolutionary Computation (CEC 2014)', pp. 2515–2523.

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. [2009], *Introduction to Algorithms*, 3rd edn.

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

[4] Gießen, C. [2013], Hybridizing evolutionary algorithms with opportunistic local search, *in* 'Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)', pp. 797–804.

[5] Hains, D. R., Whitley, D. L. and Howe, A. E. [2011], 'Revisiting the big valley search space structure in the tsp', *Journal of the Operational Research Society* **62**(2), 305–312.

[6] Lissovoi, A., Oliveto, P. S. and Warwicker, J. A. [2017], On the runtime analysis of generalised selection hyper-heuristics for pseudo-boolean optimisation, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)', pp. 849–856.

[7] Lourenço, H. R., Martin, O. C. and Stützle, T. [2002], Iterated local search, *in* 'Handbook of Metaheuristics', Vol. 57 of *International Series in Operations Research & Management Science*, pp. 321–353.

[8] Merz, P. and Freisleben, B. [1998], Memetic algorithms and the fitness landscape of the graph bi-partitioning problem, *in* 'Parallel Problem Solving from Nature (PPSN V)', pp. 765–774.

[9] Motwani, R. and Raghavan, P. [1995], *Randomized Algorithms*.

[10] Neri, F. and Cotta, C. [2012], 'Memetic algorithms and memetic computing optimization: A literature review', *Swarm and Evolutionary Computation* **2**, 1–14.

[11] Neri, F., Cotta, C. and Moscato, P., eds [2012], *Handbook of Memetic Algorithms*, Vol. 379 of *Studies in Computational Intelligence*.

[12] Ochoa, G. and Veerapen, N. [2016], Deconstructing the big valley search space hypothesis, *in* 'Proceedings of the 16th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2016)', pp. 58–73.

[13] Paixão, T., Pérez Heredia, J., Sudholt, D. and Trubenová, B. [2017], 'Towards a runtime comparison of natural and artificial evolution', *Algorithmica* **78**(2), 681–713.

[14] Prügel-Bennett, A. [2004], 'When a genetic algorithm outperforms hill-climbing', *Theoretical Computer Science* **320**(1), 135 – 153.

[15] Reeves, C. R. [1999], 'Landscapes, operators and heuristic search', *Annals of Operations Research* **86**(0), 473–490.

[16] Shi, J., Zhang, Q. and Tsang, E. P. K. [2017], 'EB-GLS: an improved guided local search based on the big valley structure', *Memetic Computing* **abs/1709.07576**.

[17] Sudholt, D. [2006*a*], Local search in evolutionary algorithms: the impact of the local search frequency, *in* 'Proceedings of the 17th International Symposium on Algorithms and Computation (ISAAC '06)', Vol. 4288 of *LNCS*, pp. 359–368.

[18] Sudholt, D. [2006*b*], On the analysis of the (1+1) memetic algorithm, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '06)', pp. 493–500.

[19] Sudholt, D. [2009], 'The impact of parametrization in memetic evolutionary algorithms', *Theoretical Computer Science* **410**(26), 2511–2528.

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

[21] Sudholt, D. [2011*b*], Memetic evolutionary algorithms, *in* A. Auger and B. Doerr, eds, 'Theory of Randomized Search Heuristics – Foundations and Recent Developments', number 1 *in* 'Series on Theoretical Computer Science', pp. 141–169.

[22] Sudholt, D. and Zarges, C. [2010], Analysis of an iterated local search algorithm for vertex coloring, *in* '21st International Symposium on Algorithms and Computation (ISAAC 2010)', Vol. 6506 of *LNCS*, pp. 340–352.

[23] Wei, K. and Dinneen, M. J. [2014*a*], Runtime analysis comparison of two fitness functions on a memetic algorithm for the clique problem, *in* 'IEEE Congress on Evolutionary Computation (CEC '14)', pp. 133–140.

[24] Wei, K. and Dinneen, M. J. [2014*b*], Runtime analysis to compare best-improvement and first-improvement in memetic algorithms, *in* 'Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14)', pp. 1439–1446.

[25] Witt, C. [2012], 'Analysis of an iterated local search algorithm for vertex cover in sparse random graphs', *Theoretical Computer Science* **425**(0), 117–125.