When Non-Elitism Outperforms Elitism for Crossing Fitness Valleys

Pietro S. Oliveto
University of Sheffield
Sheffield S1 4DP, United Kingdom

Tiago Paixão
IST Austria
Am Campus 1, 3400
Klosterneuburg, Austria

Jorge Pérez Heredia
University of Sheffield
Sheffield S1 4DP, United Kingdom

Dirk Sudholt
University of Sheffield
Sheffield S1 4DP, United Kingdom

Barbora Trubenová
IST Austria
Am Campus 1, 3400
Klosterneuburg, Austria

ABSTRACT

Crossing fitness valleys is one of the major obstacles to function optimization. In this paper we investigate how the structure of the fitness valley, namely its depth d and length ℓ, influence the runtime of different strategies for crossing these valleys. We present a runtime comparison between the (1+1) EA and two non-elitist nature-inspired algorithms, Strong Selection Weak Mutation (SSWM) and the Metropolis algorithm. While the (1+1) EA has to jump across the valley to a point of higher fitness because it does not accept decreasing moves, the non-elitist algorithms may cross the valley by accepting worsening moves.

We show that while the runtime of the (1+1) EA algorithm depends critically on the length of the valley, the runtimes of the non-elitist algorithms depend crucially only on the depth of the valley. In particular, the expected runtime of both SSWM and Metropolis is polynomial in ℓ and exponential in d while the (1+1) EA is efficient only for valleys of small length. Moreover, we show that both SSWM and Metropolis can also efficiently optimize a rugged function consisting of consecutive valleys.

Keywords

fitness valley; valley path; non-elitism; runtime analysis; natural evolution; theory; strong selection weak mutation regime; Metropolis algorithm; simulated annealing;

1. INTRODUCTION

Randomised search heuristics (RSHs) are general purpose optimisation tools typically used when no good problem specific algorithm is known for the problem at hand.

GECCO '16 July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4206-3/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2908812.2908909


Families of RSHs mainly differ in the way new solutions are generated (i.e. variation operators), how solutions are chosen for the next iterations (i.e. selection) and how many solutions are used by the heuristic in each iteration (i.e. population). Deciding which strategy to use to overcome local optima is crucial in global optimisation on multimodal problems. Two different approaches are commonly used. One strategy is to rely on variation operators such as mutation to produce new solutions of high fitness outside the basin of attraction of the local optimum. Elitist algorithms mainly rely on such strategies when stuck in a local optimum. A different approach is to attempt to escape by accepting solutions of lower fitness in the hope of eventually leaving the basin of attraction of the local optimum. This approach is the main driving force behind non-elitist algorithms. While both approaches may clearly be promising, it is unclear when one should be preferred to the other.

In this paper we investigate this topic by considering general fitness valleys of arbitrary length ℓ and depth d. We define a valley on a Hamming path (a path of Hamming neighbours) to ensure that mutation has the same probability of going forward on the path as going backwards. The valley is composed of a slope of length ℓ_1 descending towards a local minimum from which a slope of increasing fitness of length ℓ_2 can be taken to reach the end of the valley. The steepness of each slope is controlled by parameters d_1 and d_2, respectively indicating the fitness of the two local optima at the extreme left and extreme right of the valley. Our aim is to analyse how the characteristics of the valley impact the performance of elitist versus non-elitist strategies.

We point out that understanding how to cross fitness valleys efficiently is a very important problem also in biology [12]. From a biological perspective, crossing fitness valleys represents one of the major obstacles to the evolution of complex traits. Many of these traits require accumulation of multiple mutations that are individually harmful for their bearers; a fitness advantage is achieved only when all mutations have been acquired—a fitness valley has been crossed.

We consider the simple elitist (1+1) EA and compare its ability to cross fitness valleys with the recently introduced non-elitist Strong Selection Weak Mutation (SSWM) algorithm inspired by a model of biological evolution in the

'strong selection, weak mutation regime' [9]. This regime applies when mutations are rare enough and selection is strong enough that the time between occurrences of new mutations is long compared to the time a new genotype takes to replace its parent genotype, or to be lost entirely [3]. Mutations occur rarely, therefore only one genotype is present in the population most of the time, and the relevant dynamics can be characterized by a stochastic process on one genotype.

Recently, Paixão *et al.* investigated SSWM on $\textsc{Cliff}_d$ [9], a function defined such that non-elitist algorithms have a chance to jump down a "cliff" of height roughly d and to traverse a fitness valley of Hamming distance d to the optimum. The function is a generalised construction of the unitation function introduced by Jägersküpper and Storch to give an example class of functions where a (1,λ) EA outperforms a (1+λ) EA [6]. This analysis revealed that SSWM can cross the fitness valley. However, upon comparison with the (1+1) EA, SSWM achieved only a small speed-up: the expected time of SSWM is at most $n^d/e^{\Omega(d)}$, while the (1+1) EA requires $\Theta(n^d)$ [9].

In this manuscript, we show that greater speed-ups can be achieved by SSWM on fitness valleys. Differently to the work in [9] where global mutations were used, here we only allow SSWM to use local mutations because we are interested in comparing the benefits of escaping local optima by using non-elitism to cross valleys against the benefits of jumping to the other side by large mutations.

After presenting some Preliminaries, we build upon Gambler's Ruin theory [2] in Section 3 to devise a general mathematical framework for the analysis of non-elitist algorithms using local mutations for crossing fitness valleys. We use it to rigorously show that SSWM is able to efficiently perform a random walk across the valley using only local mutations by accepting worse solutions, provided that the valley is not too deep. On the other hand, the (1+1) EA cannot accept worse solutions and therefore relies on global mutations to reach the other side of the valley in a single jump. As a result, the runtime of the (1+1) EA is exponential in the length of the valley while the runtime of SSWM depends crucially only on the depth of the valley. We demonstrate the generality of the presented mathematical tool by using it to prove that the same asymptotic results achieved by SSWM also hold for the well-known Metropolis algorithm. Jansen and Wegener [7] previously compared the performance of the (1+1) EA and Metropolis for a fitness valley encoded as a unitation function where the slopes are symmetric and of the same length. They used their fitness valley as an example where the performance of the two algorithms is asymptotically equivalent. The function class considered herein is more general since it allows for arbitrary slopes.

The framework also allows the analysis for concatenated "paths" of several consecutive valleys, creating a rugged fitness landscape that loosely resembles a "big valley" structure found in many problems from combinatorial optimisation. In particular, in Section 4 we use it to prove that SSWM and Metropolis can cross consecutive paths in expected time that only depends crucially on the depth and number of the valleys.

In this extended abstract many proofs are omitted due to space restrictions.

2. PRELIMINARIES

2.1 Algorithms

In this paper we present a runtime comparison between the (1+1) EA and two non-elitist nature-inspired algorithms, SSWM and Metropolis. While they match the same basic scheme shown in Algorithm 1, they differ in the way they generate new solutions (mutate(x) function), and in the acceptance probability of these new solutions (p_{acc} function).

Algorithm 1 General scheme

Choose $x \in \{0,1\}^n$ uniformly at random
repeat
  $y \leftarrow \text{mutate}(x)$
  $\Delta f = f(y) - f(x)$
  Choose $r \in [0,1]$ uniformly at random
  if $r \leq p_{\text{acc}}(\Delta f)$ **then**
    $x \leftarrow y$
  end if
until stop

The (1+1) EA relies on global mutations to cross the fitness valley and the function mutate(x) flips all bits with uniform probability $1/n$. Conversely, SSWM and Metropolis analysed here use local mutations, hence the function mutate(x) flips a single bit chosen uniformly at random.

Furthermore, the (1+1) EA always accepts a better solution, with ties resolved in favour of the new solution. The probability of acceptance is formally described by

$$p_{\text{acc}}^{\text{EA}}(\Delta f) = \begin{cases} 1 & \text{if } \Delta f \geq 0 \\ 0 & \text{if } \Delta f < 0. \end{cases}$$

SSWM accepts candidate solutions with probability

$$p_{\text{acc}}^{\text{SSWM}}(\Delta f) = p_{\text{fix}}(\Delta f) = \frac{1 - e^{-2\beta\Delta f}}{1 - e^{-2N\beta\Delta f}} \qquad (1)$$

(see Figure 1) where $\Delta f \neq 0$ is the fitness difference between the new and the current solution, $N \geq 1$ is the size of the underlying population and β represents the *selection strength*. For $\Delta f = 0$ we define $p_{\text{acc}}(0) := \lim_{\Delta f \to 0} p_{\text{acc}}(\Delta f) = \frac{1}{N}$. If $N = 1$, this probability is $p_{\text{acc}}(\Delta f) = 1$, meaning that any offspring will be accepted, and if $N \to \infty$, it will only accept solutions for which $\Delta f > 0$. SSWM's acceptance function depends on the absolute difference in fitness between genotypes. It introduces two main differences compared to the (1+1) EA: first, solutions of lower fitness may be accepted with some positive probability, and second, solutions of higher fitness can be rejected. The formula (1), first derived by Kimura [8], represents the probability that a gene that is initially present in one copy in a population of N individuals is eventually present in all individuals (the *probability of fixation*).

The acceptance function p_{acc} is strictly increasing with limits $\lim_{\Delta f \to -\infty} p_{\text{acc}}(\Delta f) = 0$ and $\lim_{\Delta f \to \infty} p_{\text{acc}}(\Delta f) = 1$. The same limits are obtained when β tends to ∞, and thus for large $|\beta\Delta f|$ the probability of acceptance is close to the one in the (1+1) EA, as long as $N > 1$, defeating the purpose of the comparison, with the only difference being the tie-breaking rule: SSWM only accepts the new equally good solution with probability $1/N$ [9].

Finally, the Metropolis algorithm is similar to SSWM in the sense that it is able to accept mutations that decrease

fitness with some probability. However, unlike SSWM, for fitness improvements it behaves like the (1+1) EA in that it accepts *any* fitness improvement. Formally, Metropolis' acceptance function can be described by:

$$p_{\text{acc}}^{\text{MET}}(\Delta f) = \begin{cases} 1 & \text{if } \Delta f \geq 0 \\ e^{\alpha \Delta f} & \text{if} \Delta f < 0 \end{cases}$$

where α is the reciprocal of the "temperature". Temperature in the Metropolis algorithm plays the same role as population size in SSWM: increasing the temperature (decreasing α) increases the probability of accepting fitness decreases. The acceptance functions of all three algorithms are shown in Figure 1.

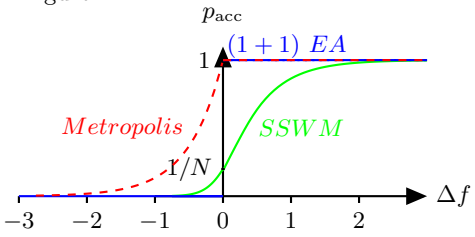

Figure 1: Probability of acceptance. Red - Metropolis, Blue - (1+1) EA, Green - SSWM.

2.2 Long Paths

Previous work on valley crossing [6, 9, 7] used functions of unitation to encode fitness valleys, with 1^n being a global optimum. The drawback of this construction is that the transition probabilities for mutation heavily depend on the current position. The closer an algorithm gets to 1^n, the larger the probability of decreasing the number of ones and moving away from the optimum.

We follow a different approach to avoid this mutational bias, and to ensure that the structure of the fitness valley is independent of its position in the search space. This also allows us to easily concatenate multiple valleys.

We base our construction on so-called *long k-paths*, paths of Hamming neighbors with increasing fitness whose length can be exponential in n [5, 10, 1]. An example is shown in Table 1; for a formal definition we refer to [11, page 2517].

An exponential length implies that the path has to be folded in $\{0,1\}^n$ in a sense that there are $i < j$ such that the i-th and the j-th point on the path have Hamming distance $\mathrm{H}(\cdot,\cdot)$ smaller than $j - i$. Standard bit mutations have a positive probability of jumping from the i-th to the j-th point, hence there is a chance to skip large parts of the path by taking a shortcut. However, long k-paths are constructed in such a way that at least k bits have to flip simultaneously in order to take a shortcut. The probability of such an event is exponentially small if $k = \Theta(\sqrt{n})$, in which case the path still has exponential length.

Long k-paths turn out to be very useful for our purposes. If we consider the first points of a long k-path and assign increasing fitness values to them, we obtain a fitness-increasing path of any desired length.

P_0: 000000000	P_6 : 000111111	P_{12}: 111111000	P_{18}: 111000111
P_1: 000000001	P_7 : 000111011	P_{13}: 111111001	P_{19}: 111000011
P_2: 000000011	P_8 : 000111001	P_{14}: 111111011	P_{20}: 111000001
P_3: 000000111	P_9 : 000111000	P_{15}: 111111111	P_{21}: 111000000
P_4: 000001111	P_{10}: 001111000	P_{16}: 111011111	
P_5: 000011111	P_{11}: 011111000	P_{17}: 111001111	

Table 1: Example of a long k-path with $n = 9, k = 3$.

Given two points $\mathrm{P}_s, \mathrm{P}_{s+i}$ for $i > 0$, P_{s+i} is called the i-th *successor* of P_s and P_s is called a *predecessor* of P_{s+i}. Long k-paths have the following properties.

LEMMA 1 (LONG PATHS).

1. *For every $i \in \mathbb{N}_0$ and path points P_s and P_{s+i}, if $i < k$ then $H(\mathrm{P}_s, \mathrm{P}_{s+i}) = i$, otherwise $H(\mathrm{P}_s, \mathrm{P}_{s+i}) \geq k$.*
2. *The probability of a standard bit mutation turning P_s into P_{s+i} (or P_{s+i} into P_s) is $1/n^i \cdot (1-1/n)^{n-i}$ for $0 \leq i < k$ and the probability of reaching any P_{s+i} from P_s for $i \geq k$ is at most $1/(k!)$.*

Proof. The first statement was shown in [11, page 2517] (refining a previous analysis in [1, page 73]). The second statement follows from the first one, using that the probability of mutating at least k bits is at most $\binom{n}{k} n^{-k} \leq 1/(k!)$. □

In the following, we fix $k := \sqrt{n}$ such that the probability of taking a shortcut on the path is exponentially small. We assign fitness values such that all points on the path have a higher fitness than those off the path. This fitness difference is made large enough such that the considered algorithms are very unlikely to ever fall off the path. Assuming that we want to use the first m path points $\mathrm{P}_0, \ldots, \mathrm{P}_{m-1}$, then the fitness is given by

$$f(x) := \begin{cases} h(i) & \text{if } x = \mathrm{P}_i, i < m \\ -\infty & \text{otherwise} \end{cases}$$

where $h(i)$ gives the fitness (height) of the i-th path point.

Then, assuming the algorithm is currently on the path, the fitness landscape is a one-dimensional landscape where (except for the two ends) each point has a Hamming neighbour as predecessor and a Hamming neighbour as successor on the path. Local mutations will create each of these with equal probability $1/n$. If we call these steps *relevant* and ignore all other steps, we get a stochastic process where in each relevant step we create a mutant up or down the path with probability 1/2 each (for the two ends we assume a self-loop probability of 1/2). The probability whether such a move is accepted then depends on the fitness difference between these path points.

It then suffices to study the expected number of relevant steps, as we obtain the expected number of function evaluations by multiplying with the expected waiting time $n/2$ for a relevant step.

LEMMA 2. *Let $E(T)$ be the expected number of relevant steps for any Algorithm 1 with local mutations finding a global optimum. Then the respective expected number of function evaluations is $n/2 \cdot E(T)$, unless the algorithm falls off the path.*

In the following, we assume that all algorithms start on P_0. This behaviour can be simulated from random initialisation with high probability by embedding the path into a larger search space and giving hints to find the start of the path within this larger space [11]. As such a construction is cumbersome and does not lead to additional insights, we simply assume that all algorithms start in P_0.

3. CROSSING SIMPLE VALLEYS

In this section we consider paths consisting of two slopes of lengths $\ell_1, \ell_2 \in \{2, 3, ...\}$ respectively. On the first slope starting at point P_0 the fitness decreases from the initial height $d_1 \in \mathbb{R}^+$ until the path point P_{ℓ_1} of minimal fitness. Then the second slope begins with fitness increasing up to the path point $P_{\ell_1+\ell_2}$ of fitness $d_2 \in \mathbb{R}^+$. The total length of the path is $\ell = \ell_1 + \ell_2$. We call such a path VALLEY.

$$h(i)_{\text{VALLEY}} := \begin{cases} d_1 - i \cdot \frac{d_1}{\ell_1} & \text{if } i \leq \ell_1 \\ (i - \ell_1) \cdot \frac{d_2}{\ell_2} & \text{if } \ell_1 < i \leq \ell. \end{cases}$$

Here, $\frac{d_1}{\ell_1}$ and $\frac{d_2}{\ell_2}$ indicate the steepness of the two slopes (see Figure 2).

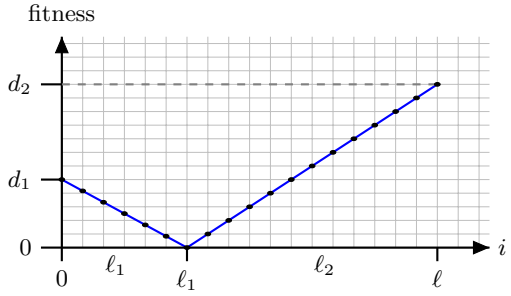

Figure 2: Sketch of the function VALLEY.

3.1 Analysis for the (1+1) EA

We first show that the runtime of the (1+1) EA depends on the length of the valley. Here we restrict parameters to $\ell_1 + \ell_2 \leq \sqrt{n}/4$, as then the probability of the (1+1) EA taking a shortcut is no larger than the probability of jumping by a distance of $\ell_1 + \ell_2$: $\frac{1}{(\sqrt{n})!} \leq n^{-\sqrt{n}/4}$ for $n \geq 4$.

THEOREM 3. *Assume $\ell_1 + \ell_2 \leq \sqrt{n}/4$ and $d_1 \leq d_2$. The expected time for the (1+1) EA starting in P_0 to cross the fitness valley is $\Theta(n^h)$ where $h = \ell_1 + \lceil d_1 \ell_2 / d_2 \rceil$.*

Proof sketch. To cross the fitness valley the (1+1) EA needs a jump of distance at least h, which needs in expectation $\Theta\left(n^h\right)$ iterations to occur. The algorithm might jump back to P_0 but this event is very unlikely. Finally, the time to climb the remaining steps is negligible. □

3.2 A general framework for local search algorithms

We introduce a general framework to analyse the expected number of relevant steps of non-elitist local search algorithms (Algorithm 1 with local mutations) for the VALLEY problem. As explained in Section 2.2, in a relevant step mutation creates a mutant up or down the path with probability 1/2, and this move is accepted with a probability that depends only on the fitness difference. For slopes where the gradient is the same at every position, this resembles a gambler's ruin process.

To apply classical gambler ruin theory [2] two technicalities need to be taken into account. Firstly, two different gambler ruin games need to be considered, one for descending down the first slope and another one for climbing up the second slope. The process may transition between these two ruin games as the extreme ends of each game at the bottom of the valley are not absorbing states. Secondly, the probabilities of winning or losing a dollar (i.e., the probabilities of moving one step up or down a slope) do not necessarily add up to one, but loop probabilities of neither winning or losing a dollar need to be taken into account when estimating expected times (winning probabilities are unaffected by self-loops). In order to simplify the calculations we have developed the following notation.

DEFINITION 1 (FRAMEWORK'S NOTATION). *The VALLEY problem can be considered as a Markov chain with states $\{P_0, P_1, \ldots, P_{\ell_1-1}, P_{\ell_1}, P_{\ell_1+1}, \ldots, P_{\ell_1+\ell_2}\}$. For simplicity we will sometimes refer to these points only with their sub-indices $\{0, 1, \ldots, \ell_1 - 1, \ell_1, \ell_1 + 1, \ldots, \ell_1 + \ell_2\}$.*

We will denote respectively by $p_{i\to j}$ and $E(T_{i\to j})$ the probability and expected time of moving from state i to $j \in \{i-1, i, i+1\}$ in one iteration. Analogously, we will denote by $p^{\text{GR}}_{i\to k}$ the probability of a Gambler's Ruin process starting in i finishing in k before reaching the state $i-1$. And by $E\left(T^{\text{GR}}_{i,k}\right)$ we denote the expected duration of such process.

The following lemmas simplify the runtime analysis of any algorithm that matches the scheme of Algorithm 1 for local mutations and some reasonable conditions.

A common feature of optimisation algorithms is that the selection operator prefers fitness increases over decreases e.g. Randomized Local Search, (1+1) EA or Metropolis. Then, the bottleneck of VALLEY seems to be climbing down the first ℓ_1 steps since several fitness decreasing mutations have to be accepted.

Once in the bottom of the valley P_{ℓ_1} the process may keep moving. It could be the case that the algorithm climbs up again to P_0. But under some mild conditions it will only have to repeat the experiment a constant number of times. Finally, the algorithm will have to climb up to $P_{\ell_1+\ell_2}$. This will take linear time in ℓ_2, provided the probability of accepting an improvement of Δf is by a constant greater than accepting a worsening of the same size.

LEMMA 4. *Consider any algorithm described by Algorithm 1 with local mutations and the following properties on VALLEY with $\ell_1, \ell_2 \in \{2, 3, ...\}$ and $d_1, d_2 \in R^+$*

(i) $p_{\ell_1\to\ell_1-1},\ p_{\ell_1\to\ell_1+1} = \Omega(1)$

(ii) $p^2_{\ell_1\to\ell_1+1} > p_{\ell_1+1\to\ell_1}$

(iii) $p_{\text{acc}}(\Delta f)$ *is non-decreasing*

(iv) $p_{\ell_1\to\ell_1+1} \geq (1+\varepsilon) \cdot p_{\ell_1+1\to\ell_1}$, *for $\varepsilon > 0$ a constant.*

Then the expected number of relevant steps for such process to reach the point $P_{\ell_1+\ell_2}$ starting from P_0 is

$$E(T_{0\to\ell_1+\ell_2}) = \Theta\left(E(T_{1\to\ell_1})\right) + \Theta(\ell_2).$$

In order to prove this we will make use of the following lemma that shows some implications of the conditions from the previous lemma.

LEMMA 5. *In the context of Lemma 4, properties (i) and (ii) imply that*

(i) $p_{\ell_1\to\ell_1-1} + p_{\ell_1\to\ell_1+1} = \frac{1}{c_1}$ *for some constant $c_1 \geq 1$*

(ii) $1 - c_1 \cdot p^{\text{GR}}_{\ell_1+1\to\ell_1} = \frac{1}{c_2}$ *for some constant $c_2 > 1$*

(iii) $1 - c_1 c_2 \cdot p_{\ell_1\to\ell_1-1} = \frac{1}{c_3}$ *for some constant $c_3 > 1$.*

The proof of Lemma 5 is omitted.

Proof of Lemma 4. Since the algorithm only produces points in the Hamming neighbourhood it will have to pass through all the states on the path. We break down the set of states in three sets and expand the total time as the sum of the optimisation time for those three sets:

$$\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right) = \mathrm{E}\left(T_{0\to 1}\right) + \mathrm{E}\left(T_{1\to\ell_1}\right) + \mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right). \quad (2)$$

Note that the lower bound follows directly. Let us now consider the upper bound. We start using a recurrence relation for the last term: once in state ℓ_1, after one iteration, the algorithm can either move to state ℓ_1+1 with probability $p_{\ell_1\to\ell_1+1}$, move to state ℓ_1-1 with probability $p_{\ell_1\to\ell_1-1}$ or stay in state ℓ_1 with the remaining probability (if the mutation is not accepted).

$$\begin{aligned}
&\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right) = 1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}\left(T_{\ell_1+1\to\ell_1+\ell_2}\right)\\
&\quad + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{\ell_1-1\to\ell_1+\ell_2}\right) + p_{\ell_1\to\ell_1}\cdot\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right)\\
&\le 1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}\left(T_{\ell_1+1\to\ell_1+\ell_2}\right)\\
&\quad + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right) + p_{\ell_1\to\ell_1}\cdot\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right).
\end{aligned}$$

Solving the previous expression for $\mathrm{E}\left(T_{1\to\ell_1+\ell_2}\right)$ leads to

$$\begin{aligned}
&\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right)\\
&\le \frac{1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}\left(T_{\ell_1+1\to\ell_1+\ell_2}\right) + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right)}{p_{\ell_1\to\ell_1-1} + p_{\ell_1\to\ell_1+1}}.
\end{aligned}$$

Property (i) of Lemma 4 implies that the denominator is a constant $1/c_1$, then

$$\begin{aligned}
&\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right) \quad (3)\\
&\le c_1\left(1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}\left(T_{\ell_1+1\to\ell_1+\ell_2}\right) + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right)\right).
\end{aligned}$$

Since the acceptance probability is a function of Δf, for both sides of the valley the probabilities of moving to the next or previous state remains constant during each slope and we can cast the behaviour as a Gambler's Ruin problem. Then, when the state is P_{ℓ_1+1} a Gambler's Ruin game (with self-loops) occurs. The two possible outcomes are: (1) the problem is optimised or (2) we are back in P_{ℓ_1}.

$$\mathrm{E}\left(T_{\ell_1+1\to\ell_1+\ell_2}\right) = \mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right) + p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}\cdot\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right). \quad (4)$$

Now we introduce (4) in (3), obtaining

$$\begin{aligned}
&\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right) \le c_1\left(1 + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right)\right)\\
&+ c_1\cdot p_{\ell_1\to\ell_1+1}\cdot\left(\mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right) + p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}\cdot\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right)\right).
\end{aligned}$$

Solving for $\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right)$ yields

$$\begin{aligned}
&\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right)\\
&\le \frac{c_1\left(1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right) + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right)\right)}{1 - c_1\cdot p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}}.
\end{aligned}$$

By Lemma 5, properties (i) and (ii) of Lemma 4 imply that the denominator is a constant $1/c_2$. Hence,

$$\begin{aligned}
&(c_2c_1)^{-1}\cdot\mathrm{E}\left(T_{\ell_1\to\ell_1+\ell_2}\right)\\
&\le 1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right) + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right)\\
&\le 1 + \mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right) + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right).
\end{aligned}$$

We introduce this into (2), leading to

$$\begin{aligned}
&\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right) \le \mathrm{E}\left(T_{0\to 1}\right) + \mathrm{E}\left(T_{1\to\ell_1}\right)\\
&\quad + c_2c_1\left(1 + \mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right) + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right)\right).
\end{aligned}$$

Solving for $\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right)$ leads to

$$\begin{aligned}
&\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right)\\
&\le \frac{\mathrm{E}\left(T_{0\to 1}\right) + \mathrm{E}\left(T_{1\to\ell_1}\right) + c_2c_1 + c_1c_2\cdot\mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right)}{1 - c_1c_2\cdot p_{\ell_1\to\ell_1-1}}.
\end{aligned}$$

Again by Lemma 5, properties (i) and (ii) of Lemma 4 imply that the denominator is a constant $1/c_3$. Hence,

$$\begin{aligned}
&\mathrm{E}\left(T_{0\to\ell_1+\ell_2}\right) \quad (5)\\
&\le c_3\left(\mathrm{E}\left(T_{0\to 1}\right) + \mathrm{E}\left(T_{1\to\ell_1}\right) + c_2c_1 + c_1c_2\cdot\mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right)\right).
\end{aligned}$$

Now we consider the last term. Due to property (iv) of Lemma 4 once in ℓ_1+1 there is a constant probability of moving towards the optimum, then $\mathrm{E}\left(T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right) = \Theta(\ell_2)$. Plugging this into (5) proves the claimed upper bound. □

Now we estimate the time to move from P_0 to P_{ℓ_1}. As in the previous proof, the main arguments are a recurrence relation and a Gambler's Ruin game.

LEMMA 6. *Consider any algorithm described by Algorithm 1 with local mutations on* VALLEY *with* $\ell_1, \ell_2 \in \{2, 3, ...\}$ *and* $d_1, d_2 \in R^+$. *Then the number of relevant steps to go from the state* P_1 *to* P_{ℓ_1} *is*

$$E\left(T_{1\to\ell_1}\right) = \frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\cdot\left(E\left(T^{\mathrm{GR}}_{1,\ell_1}\right) + \frac{p^{\mathrm{GR}}_{1\to 0}}{p_{0\to 1}}\right).$$

Proof. At the state P_1 a Gambler's Ruin game (with self-loops) occurs. The two possible outcomes are: (1) we have reached the valley P_{ℓ_1} or (2) we are back to P_0.

$$\begin{aligned}
\mathrm{E}\left(T_{1\to\ell_1}\right) &= \mathrm{E}\left(T^{\mathrm{GR}}_{1,\ell_1}\right) + p^{\mathrm{GR}}_{1\to 0}\cdot\mathrm{E}\left(T_{0\to\ell_1}\right)\\
&= \mathrm{E}\left(T^{\mathrm{GR}}_{1,\ell_1}\right) + p^{\mathrm{GR}}_{1\to 0}\cdot\left(\mathrm{E}\left(T_{0\to 1}\right) + \mathrm{E}\left(T_{1\to\ell_1}\right)\right).
\end{aligned}$$

Solving for $\mathrm{E}\left(T_{1\to\ell_1}\right)$ leads to

$$\mathrm{E}\left(T_{1\to\ell_1}\right) = \frac{\mathrm{E}\left(T^{\mathrm{GR}}_{1,\ell_1}\right) + p^{\mathrm{GR}}_{1\to 0}\cdot\mathrm{E}\left(T_{0\to 1}\right)}{1 - p^{\mathrm{GR}}_{1\to 0}}$$

Which using $1 - p^{\mathrm{GR}}_{1\to 0} = p^{\mathrm{GR}}_{1\to\ell_1}$ simplifies to

$$\mathrm{E}\left(T_{1\to\ell_1}\right) = \frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\cdot\left(\mathrm{E}\left(T^{\mathrm{GR}}_{1,\ell_1}\right) + \frac{p^{\mathrm{GR}}_{1\to 0}}{p_{0\to 1}}\right).$$ □

3.3 Application to SSWM

In this subsection we make use of the previous framework to analyse the SSWM for the VALLEY problem. To apply this framework we need to know how a Gambler's Ruin with the fixation probabilities of the SSWM behaves. When dealing with these probabilities the ratio between symmetric fitness variations appears often. The next lemma will be very helpful to simplify this ratio.

LEMMA 7. *For every* $\beta \in \mathbb{R}^+$, $\Delta f \in \mathbb{R}$ *and* $N \in \mathbb{N}^+$ $\frac{p_{\mathrm{fix}}(-\Delta f)}{p_{\mathrm{fix}}(+\Delta f)} = e^{-2(N-1)\beta\Delta f}$.

Proof. The proof follows from the definition of p_{fix} and applying the relation $e^x = (e^x - 1)/(1 - e^{-x})$. □

The following lemma contains bounds on the expected duration of the game and winning probabilities for SSWM. Although VALLEY has slopes of d_1/ℓ_1 and d_2/ℓ_2, SSWM through the action of the parameter β sees an effective gradient of $\beta \cdot d_1/\ell_1$ and $\beta \cdot d_2/\ell_2$. Varying this parameter allows the algorithm to accommodate the slope to a comfortable value. We have set this effective gradient to $\beta|\Delta f| = \Omega(1)$ so that the probability of accepting an improvement is $\Omega(1)$.

LEMMA 8 (SSWM GAMBLER'S RUIN). *Consider a Gambler's Ruin problem where in each iteration player one wins a dollar with probability $p_1 = \frac{1}{2} \cdot p_{\text{fix}}(\Delta f)$ and player two with probability $p_2 = \frac{1}{2} \cdot p_{\text{fix}}(-\Delta f)$.*

Let player one start with $n_1 = 1$ dollar and player two start with $n_2 = \ell - 1$ dollars, $\Delta f < 0$ and $(N-1)\beta|\Delta f| = \Omega(1)$. Then the winning probability of player one can be bounded as follows

$$\frac{-2(N-1)\beta\Delta f}{e^{-2(N-1)\beta k\Delta f}} \leq P_1 \leq \frac{e^{-2(N-1)\beta\Delta f}}{e^{-2(N-1)\beta(n_1+n_2)\Delta f} - 1}$$

and the expected number of evaluations until the end of the game is $E(T_f) = O(1)$.

Proof. The main difference with the classical Gambler's Ruin is that we have self-loop probabilities (because in general $\frac{1}{2} \cdot p_{\text{fix}}(\Delta f) + \frac{1}{2} \cdot p_{\text{fix}}(-\Delta f) < 1$).

Obviously, this only affects the duration of the game but not the winning probabilities. Using Lemma 7 and the well-known Gambler's Ruin results [2] we get:

$$P_1 = \frac{1 - \left(\frac{p_2}{p_1}\right)^{n_1}}{1 - \left(\frac{p_2}{p_1}\right)^{n_1+n_2}} = \frac{1 - \left(\frac{p_{\text{fix}}(-\Delta f)}{p_{\text{fix}}(\Delta f)}\right)^{n_1}}{1 - \left(\frac{p_{\text{fix}}(-\Delta f)}{p_{\text{fix}}(\Delta f)}\right)^{n_1+n_2}} = \frac{1 - e^{-2(N-1)\beta n_1 \Delta f}}{1 - e^{-2(N-1)\beta(n_1+n_2)\Delta f}}.$$

Notice that this is the same expression as the fixation probability if we change β for $(N-1)\beta$ and N for ℓ. Then we can apply the bounds for the original fixation probabilities from Lemma 1 in [9] to obtain the inequalities of the theorem's statement.

For the expected duration of the game, we use the result of a classic Gambler's Ruin [2] and divide by the probability of having a relevant step $p_1 + p_2$.

$$\begin{aligned} \mathrm{E}(T_f) = & \frac{1}{p_2 + p_1} \cdot \frac{n_1 - (n_1 + n_2) \cdot P_1}{p_2 - p_1} \\ \leq & \frac{1 - \ell \cdot P_1}{p_2^2 - p_1^2} \leq \frac{1}{p_2^2 - p_1^2}. \end{aligned}$$

Using Lemma 7 with $(N-1)\beta|\Delta f| = \Omega(1)$ leads to $p_2 = \Omega(1)$. Finally, using $p_2 > p_1$ the constant upper bound follows, $\mathrm{E}(T_f) = O(1)$. □

While the optimisation time of the (1+1) EA grows exponentially with the length of the valley, the following theorem shows that for the SSWM the growth is exponential in the depth of the valley. This means that the SSWM can effectively cross long valleys with moderate gradients. On the other hand even if the length is short it cannot efficiently optimise cliffs where the depth is large.

THEOREM 9. *The expected number of function evaluations T_f for SSWM with local mutations to reach $\mathrm{P}_{\ell_1+\ell_2}$ from P_0 on* VALLEY *with $\ell_1, \ell_2 \in \{2, 3, ...\}$ and $d_1, d_2 \in R^+$ is*

$$\begin{aligned} E(T_f) = & O\left(n \cdot e^{2N\beta d_1(\ell_1+1)/\ell_1}\right) + \Theta(n \cdot \ell_2) \text{ and} \\ E(T_f) = & \Omega\left(n \cdot e^{2(N-1)\beta d_1(\ell_1-1)/\ell_1}\right) + \Theta(n \cdot \ell_2) \end{aligned}$$

provided $\beta d_1/\ell_1,\ \beta d_2/\ell_2,\ N = \Omega(1)$.

Proof. The first part of the proof consists of estimating $\mathrm{E}(T_{1\to\ell_1})$ by using the statement of Lemma 6. Then we will check that the conditions from Lemma 4 are met and we will add the $\Theta(\ell_2)$ term. Finally, we will take into account the time needed for a relevant step in the long path to obtain the n factor in the bounds.

As just described above we start considering $\mathrm{E}(T_{1\to\ell_1})$ by using Lemma 6. Let us start with the upper bound.

$$\mathrm{E}(T_{1\to\ell_1}) = O\left(\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}} \cdot \left(\mathrm{E}\left(T^{\mathrm{GR}}_{1,\ell_1}\right) + \frac{1}{p_{0\to1}}\right)\right)$$

using Lemma 8 we bound $p^{\mathrm{GR}}_{1\to\ell_1}$ yielding

$$\mathrm{E}(T_{1\to\ell_1}) = O\left(\frac{e^{2(N-1)\beta d_1}}{2(N-1)\beta d_1/\ell_1} \cdot \left(O(1) + \frac{1}{p_{0\to1}}\right)\right).$$

Since p_{fix} for $\Delta f < 0$ decreases when the parameters N, β and $|\Delta f|$ increase and $N\beta d_1/\ell_1 = \Omega(1)$, we get $p^{-1}_{0\to1} = \Omega(1)$ and $O(1) + \frac{1}{p_{0\to1}} = O\left(\frac{1}{p_{0\to1}}\right)$. Hence,

$$\mathrm{E}(T_{1\to\ell_1}) = O\left(\frac{e^{2(N-1)\beta d_1}}{2(N-1)\beta d_1/\ell_1} \cdot \frac{1}{p_{0\to1}}\right)$$

Using Lemma 1 in [9] to lower bound $p_{0\to1}$ we get

$$\mathrm{E}(T_{1\to\ell_1}) = O\left(\frac{e^{2(N-1)\beta d_1}}{2(N-1)\beta d_1/\ell_1} \cdot \frac{e^{2N\beta d_1/\ell_1}}{2\beta\frac{d_1}{\ell_1}}\right).$$

Using $(N-1)\beta d_1/\ell_1 = \Omega(1)$ and $\beta d_1/\ell_1 = \Omega(1)$ leads to

$$\mathrm{E}(T_{1\to\ell_1}) = O\left(e^{2N\beta d_1(\ell_1+1)/\ell_1}\right).$$

We now consider the lower bound. Starting again from Lemmas 4 and 6 and bounding $p^{\mathrm{GR}}_{1\to\ell_1}$ with Lemma 8

$$\begin{aligned} \mathrm{E}(T_{1\to\ell_1}) &= \Omega\left(\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\right) = \Omega\left(\frac{e^{2(N-1)\beta d_1} - 1}{e^{2(N-1)\beta d_1/\ell_1}}\right) \\ &= \Omega\left(e^{2(N-1)\beta d_1 \frac{\ell_1-1}{\ell_1}} - \frac{1}{e^{2(N-1)\beta d_1/\ell_1}}\right) \\ &= \Omega\left(e^{2(N-1)\beta d_1 \frac{\ell_1-1}{\ell_1}}\right). \end{aligned}$$

Now we need to apply Lemma 4 to add the $\Theta(\ell_2)$ term in both bounds. We start checking that all the conditions are satisfied. Firstly, since p_{fix} for $\Delta f > 0$ increases when the parameters (N, β and Δf) increase, then $N\beta d_2/\ell_2 = \Omega(1)$ implies $p_{\ell_1\to\ell_1+1} = \Omega(1)$. Analogously for $p_{\ell_1\to\ell_1-1}$ with $N\beta d_1/\ell_1 = \Omega(1)$ satisfying the first property.

Secondly, property (ii) is satisfied if

$$p^2_{\ell_1\to\ell_1-1} > p_{\ell_1+1\to\ell_1} \Leftrightarrow p^2_{\text{fix}}(d_2/\ell_2) > 4 \cdot p_{\text{fix}}(-d_2/\ell_2).$$

Using Lemma 7 leads to

$$p^2_{\ell_1\to\ell_1-1} > p_{\ell_1+1\to\ell_1} \Leftrightarrow p_{\text{fix}}(d_2/\ell_2)\cdot e^{2(N-1)\beta d_2/\ell_2} > 4$$
$$\Leftrightarrow \frac{1-e^{-2\beta d_2/\ell_2}}{1-e^{-2N\beta d_2/\ell_2}}\cdot e^{2(N-1)\beta d_2/\ell_2} > 4$$
$$\Leftarrow e^{2(N-1)\beta d_2/\ell_2} - e^{2\beta(N-2)d_2/\ell_2} > 4.$$

Since all the parameters are positive the condition holds for N being a large enough constant.

The third property is satisfied since for $N > 1$ the fixation probability is strictly increasing with Δf. The proof for the fourth condition follows directly from Lemma 7 and the condition $N\beta d_2/\ell_2 = \Omega(1)$. Considering the time for a relevant step from Lemma 2 completes the proof. □

3.4 Application to Metropolis

We now apply the framework from Section 3.2 to the Metropolis algorithm. The analysis follows very closely the one of SSWM and we omit the proofs for the sake of brevity. We first cast Metropolis on Valley as a Gambler's Ruin problem. Like SSWM, Metropolis can make use of its parameter α to accommodate the gradient of Valley.

Lemma 10 (Metropolis Gambler's Ruin downhill). *Consider a Gambler's Ruin problem where in each iteration player one loses a dollar with probability $p_2 = \frac{1}{2}$ and player two with $p_1 = \frac{1}{2}\cdot e^{-\alpha\Delta f}$, where $\Delta f < 0$. Let player one start with 1 dollar, player two with $\ell-1$ dollars and $\alpha|\Delta f| = \Omega(1)$. Then, the probability that player one wins can be bounded by:*

$$\frac{-\alpha\Delta f}{e^{-\alpha\ell\Delta f}} < P_1^{\text{GR-Met}} < \frac{e^{-\alpha\Delta f}}{e^{-\alpha\ell\Delta f}-1}$$

and the expected number of function evalutations of the game is $E(T_f) = O(1)$.

Lastly, we make use of the previous lemma and the framework presented in Section 3.2 to determine bounds on the runtime of Metropolis on Valley.

Theorem 11. *The expected number of function evaluations for Metropolis to reach $P_{\ell_1+\ell_2}$ from P_0 on* Valley *with $\ell_1, \ell_2 \in \{2, 3, ...\}$ and $d_1, d_2 \in R^+$ is $O\left(n\cdot e^{\alpha d_1(1+1/\ell_1)}\right) + \Theta(n\cdot\ell_2)$ and $\Omega\left(n\cdot e^{\alpha d_1(1-1/\ell_1)}\right)+\Theta(n\cdot\ell_2)$ provided $\alpha d_1/\ell_1$, $\alpha d_2/\ell_2 = \Omega(1)$.*

4. CROSSING CONCATENATED VALLEYS

We now define a class of functions called ValleyPath consisting of m consecutive valleys of the same size. Each of the consecutive valleys is shifted such that the fitness at the beginning of each valley is the same as that at the end of the previous valley. Fitness values from one to the next valley increase by an amount of $d_2 - d_1 > 0$. Formally:

$$h(i,j)_{\text{ValleyPath}} := \begin{cases} j\cdot(d_2-d_1) + d_1 - i\cdot\frac{d_1}{\ell_1} & \text{if } i \le \ell_1 \\ j\cdot(d_2-d_1) + (i-\ell_1)\cdot\frac{d_2}{\ell_2} & \text{if } \ell_1 < i \le \ell. \end{cases}$$

Here $0 < j \le m$ indicates a valley while $0 \le i \le \ell_1 + \ell_2 = \ell$ indicates the position in the given valley. Hence, the global optimum is the path point $\mathrm{P}_{m\cdot\ell}$.

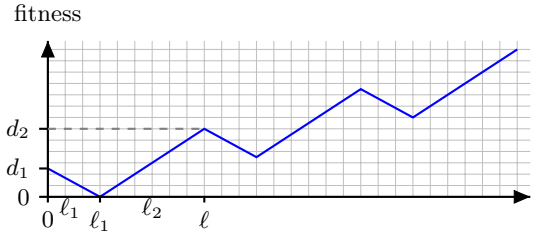

Figure 3: Sketch of the function ValleyPath.

ValleyPath represents a rugged fitness landscape with many valleys and many local optima (peaks). It loosely resembles a "big valley" structure found in many real-world problems: from a high-level view the concatenation of valleys indicates a "global" gradient, i. e. the direction towards valleys at higher indices. The difficulty for optimisation algorithms is to overcome these many local optima and to still be able to identify the underlying gradient. We show here that both SSWM and Metropolis are able to exploit this global gradient and find the global optimum efficiently. Note that ValleyPath is a very broad function class in that it allows for many shapes to emerge, from few deep valleys to many shallow ones. Our results hold for all valley paths with $d_1 < d_2$.

As in the analysis for Valley, instead of considering the whole Markov chain underlying ValleyPath we take a high-level view and consider the Markov chain that describes transitions between neighbouring peaks. Since the peaks have increasing fitness this chain is quite simple and we will use drift arguments. The idea of the next theorem is simple: if we can find constant bounds for the drift we will only need to repeat the Valley experiment for as many peaks as there are in ValleyPath.

Theorem 12. *Consider any Algorithm 1 with local mutations on* ValleyPath. *Consider the points in time where the algorithm is on a peak and focus on transitions between different peaks. Let X_t be a random variable describing the number of peaks to the right of the current valley at the t-th time a different peak is reached. If the drift over peaks can be bounded by positive constants such that*

$$0 < c_1 \le E(X_t - X_{t+1} \mid X_t = i > 0) \le c_2$$

then the expected number of function evaluations for the algorithm is

$$E(T_f) = O\left(m\cdot E\left(T^O_{\text{Valley}}\right)\right) \text{ and } \Omega\left(m\cdot E\left(T^\Omega_{\text{Valley}}\right)\right)$$

where $E\left(T^O_{\text{Valley}}\right)$ and $E\left(T^\Omega_{\text{Valley}}\right)$ are the upper and lower bounds for Valley *respectively.*

Proof. Follows directly from the application of the standard additive drift theorem [4]. □

We need a net positive drift towards the optimum to be able to apply Theorem 12. The following lemma (proof omitted) shows some conditions on the acceptance probability $p_{\text{acc}}(\Delta f)$ that are sufficient to obtain this constant drift. First, the acceptance probability for deleterious mutations must decrease rapidly. Also the probability of accepting an improvement of Δf must be much bigger than accepting a worsening of the same size with $-\Delta f$. Finally, $p_{\text{acc}}(\Delta f)$ must be a non-decreasing function of Δf.

LEMMA 13. *Consider any algorithm described by Algorithm 1 with local mutations on* VALLEYPATH *with* $\ell_2 \geq 2$ *and an acceptance probability such that for* $\delta, \alpha, \beta \geq 1$

(i) $p_{\text{acc}}(\Delta f) \geq e^{\alpha\delta} \cdot p_{\text{acc}}(\Delta f - \delta)$ *for* $\Delta f < 0$

(ii) $\frac{p_{\text{acc}}(\Delta f)}{p_{\text{acc}}(-\Delta f)} \geq e^{\beta\delta}$

(iii) $p_{\text{acc}}(\Delta f)$ *is non-decreasing.*

Let X_t *be a random variable describing the number of peaks at the right of the current valley (see Theorem 12). Then the drift between peaks* $E(X_t - X_{t+1} \mid X_t = i > 0)$ *can be lower bounded by* $2/c$*, for a constant* $c > 0$*.*

4.1 SSWM

We analyse the runtime of SSWM on VALLEYPATH. The following lemma shows that the fixation probability decreases exponentially for deleterious mutations. This will be needed to obtain a lower bound for the drift between peaks in order to apply Theorem 12.

LEMMA 14. *Let* $\delta, \beta \in \mathbb{R}^+$ *and* $\Delta f \leq 0$*. Then* $p_{\text{fix}}(\Delta f) \geq e^{\delta} \cdot p_{\text{fix}}(\Delta f - \delta)$*. Provided* $2\beta(N-1) \geq 1 + 2 \cdot \max\left(1, \frac{1}{\delta}\right)$*.*

The previous lemma together with Lemma 13 allow to show constant bounds for the drift between peaks. Then it is straightforward to obtain the runtime of SSWM on VALLEYPATH.

THEOREM 15. *The expected number of function evaluations for SSWM on* VALLEYPATH *with* $\delta = d_2/\ell_2 - d_1/\ell_1 \geq 1$*,* $\ell_2 \geq 2$ *and* $2\beta(N-1) \geq 3$ *is*

$$E(T_f) = O\left(m \cdot n \cdot \left(e^{2N\beta d_1 (l_1+1)/l_1} + \Theta(l_2)\right)\right) \text{ and}$$
$$E(T_f) = \Omega\left(m \cdot n \cdot \left(e^{2(N-1)\beta d_1 (l_1-1)/l_1} + \Theta(l_2)\right)\right).$$

Proof. Due to Lemmas 14 and 7, SSWM fits in Lemma 13 and we can apply Theorem 12 taking into account the optimisation time for VALLEY. □

4.2 Metropolis

An equivalent result to that of the SSWM for VALLEYPATH is shown for Metropolis in the following theorem.

THEOREM 16. *The expected number of function evaluations for Metropolis on* VALLEYPATH *with* $\delta = d_2/\ell_2 - d_1/\ell_1 \geq 1$*,* $\ell_2 \geq 2$ *and* $\delta\alpha > 1$ *is*

$$E(T_f) = O\left(m \cdot n \cdot \left(e^{\alpha d_1 (l_1+1)/l_1} + \Theta(l_2)\right)\right) \text{ and}$$
$$E(T_f) = \Omega\left(m \cdot n \cdot \left(e^{\alpha d_1 (l_1-1)/l_1} + \Theta(l_2)\right)\right).$$

Proof. Due to the exponential decrease of the acceptance probability for deleterious mutations, Metropolis fits in Lemma 13 and we can apply Theorem 12 taking into account the optimisation time for VALLEY. □

Note that our approach can be extended to concatenations of valleys of different sizes, assuming $d_1 < d_2$ for each valley.

5. CONCLUSIONS

We presented an analysis of randomised search heuristics for crossing fitness valleys where no mutational bias exists and thus the probability for moving forwards or backwards on the path depends only on the fitness difference between neighbouring search points. Our focus was to highlight characteristics of valleys where an elitist selection strategy should be preferred to a non-elitist one and vice versa. To achieve our goals we presented a mathematical framework to allow the analysis of non-elitist algorithms on valleys and paths of concatenated valleys. We rigorously proved that while the (1+1) EA is efficient for valleys and valley paths up to moderate lengths, both SSWM and Metropolis are efficient when the valleys and valley paths are not too deep. A natural direction for future work is to extend the mathematical framework to allow the analysis of SSWM with global mutations, thus highlighting the benefits of combining both non-elitism and global mutations for overcoming local optima.

Acknowledgments: The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 618091 (SAGE) and from the EPSRC under grant agreement no EP/M004252/1.

6. REFERENCES

[1] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[2] W. Feller. *An introduction to probability theory and its applications*. Wiley, 1968.

[3] J. H. Gillespie. Molecular evolution over the mutational landscape. *Evolution*, 38(5):1116–1129, 1984.

[4] J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

[5] J. Horn, D. E. Goldberg, and K. Deb. Long path problems. In *Parallel Problem Solving from Nature (PPSN III)*, volume 866 of *LNCS*, pages 149–158, 1994.

[6] J. Jägersküpper and T. Storch. When the plus strategy outperforms the comma strategy and when not. In *Proc. of IEEE FOCI '07*, pages 25–32. IEEE, 2007.

[7] T. Jansen and I. Wegener. A comparison of simulated annealing with a simple evolutionary algorithm on pseudo-Boolean functions of unitation. *Theoretical Computer Science*, 386(1-2):73–93, 2007.

[8] M. Kimura. On the probability of fixation of mutant genes in a population. *Genetics*, 47(6):713–719, 1962.

[9] T. Paixao, J. Pérez Heredia, D. Sudholt, and B. Trubenová. First steps towards a runtime comparison of natural and artificial evolution. In *Proc. of GECCO '15*, pages 1455–1462. ACM, 2015.

[10] G. Rudolph. How mutation and selection solve long-path problems in polynomial expected time. *Evolutionary Computation*, 4(2):195–205, 1997.

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

[12] M. C. Whitlock, P. C. Phillips, F. B.-G. Moore, and S. J. Tonsor. Multiple Fitness Peaks and Epistasis. *Annual Review of Ecology and Systematics*, 26:601–629, 1995.