Theoretical results on bet-and-run as an initialisation strategy

Andrei Lissovoi
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, United Kingdom

Dirk Sudholt
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, United Kingdom

Markus Wagner
School of Computer Science
The University of Adelaide
Adelaide, Australia

Christine Zarges
Department of Computer Science
Aberystwyth University
Aberystwyth, SY23 3DB, United Kingdom

ABSTRACT

Bet-and-run initialisation strategies have been experimentally shown to be beneficial on classical NP-complete problems such as the travelling salesperson problem and minimum vertex cover. We analyse the performance of a bet-and-run restart strategy, where k independent islands run in parallel for t_1 iterations, after which the optimisation process continues on only the best-performing island. We define a family of pseudo-Boolean functions, consisting of a plateau and a slope, as an abstraction of real fitness landscapes with promising and deceptive regions. The plateau shows a high fitness, but does not allow for further progression, whereas the slope has a low fitness initially, but does lead to the global optimum. We show that bet-and-run strategies with non-trivial k and t_1 are necessary to find the global optimum efficiently. We show that the choice of t_1 is linked to properties of the function. Finally, we provide a fixed budget analysis to guide selection of the bet-and-run parameters to maximise expected fitness after $t = k \cdot t_1 + t_2$ fitness evaluations.

CCS CONCEPTS

•Theory of computation → Optimization with randomized search heuristics; •Computing methodologies → Randomized search;

KEYWORDS

initialisation, local search, stochastic search, restarts, optimisation

ACM Reference format:
Andrei Lissovoi, Dirk Sudholt, Markus Wagner, and Christine Zarges. 2017. Theoretical results on bet-and-run as an initialisation strategy. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071329

1 INTRODUCTION

A standard reaction to a malfunctioning desktop PC is to restart it. Nowadays, stochastic search algorithms and randomised search heuristics are frequently restarted as well: If a run does not conclude within a pre-determined limit or if the solution quality is unsatisfactory, we restart the algorithm. This was shown to help avoid heavy-tailed running time distributions [6].

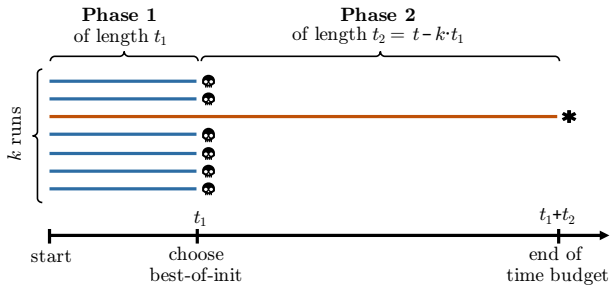

Figure 1: The *bet-and-run* restart strategy starts with k independent runs and total time budget t. After time t_1 all runs except for the best one are terminated (marked with 💀). The best run (marked with ✱) continues for t_2 time steps until the total time budget is used up. (Figure from [5] with approval)

Some theoretical results exist on how to construct optimal restart strategies. For example, Luby et al. [12] showed that, for Las Vegas algorithms with known run time distribution, there is an optimal stopping time in order to minimise the expected running time. They also showed that, if the distribution is unknown, there is a universal sequence of running times which is the optimal restarting strategy up to constant factors. While these results can be used for every problem setting, they only apply to Las Vegas algorithms.

Fewer results are known for the optimisation case. Marti [13] and Lourenço et al. [11] present practical approaches, and a recent theoretical result is presented by Schoenauer et al. [14]. Particularly for the satisfiability problem, several studies make an empirical comparison of a number of restart policies [1, 7].

Many modern optimisation algorithms, even when they work mostly deterministically, have some randomised component, for example by choosing a random starting point. Thus, the initial solution often strongly influences the quality of the outcome. It follows that it is natural to do several runs of the algorithm. Two very typical uses for an algorithm with a total time budget t are to (a) use all of time t for a single run of the algorithm (single-run strategy), or (b) to make a number of k runs of the algorithm, each with running time t/k (multi-run strategy).

Based on these two classical strategies, Fischetti and Monaci [4] investigated the use of the *bet-and-run* strategy described in Algorithm 1 and illustrated in Figure 1. Note that the multi-run strategy

GECCO '17, Berlin, Germany
 978-1-4503-4920-8/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3071178.3071329

of restarting from scratch k times is a special case by choosing $t_1 = t/k$ and $t_2 = 0$, and the single-run strategy corresponds to $k = 1$. Also note that in the first phase the k runs (each using a time budget t_1) do not have to be run in parallel in practice.

Algorithm 1 Bet-and-Run

Phase 1: Perform k runs of the algorithm for some (short) time limit t_1 with $t_1 \leq t/k$.
Phase 2:
Identify best run b among the k, breaking ties at random.
Use remaining time $t_2 = t - k \cdot t_1$ to continue b.

Fischetti and Monaci [4] experimentally studied such a *bet-and-run* strategy for mixed-integer programming. They explicitly introduce diversity in the starting conditions of the used MIP solver (IBM ILOG CPLEX) by directly accessing internal mechanisms. In their experiments with $k = 5$, bet-and-run was typically beneficial.

Recently, Friedrich et al. [5] investigated a comprehensive range of *bet-and-run* strategies on the travelling salesperson problem and the minimum vertex cover problem. Their best strategy was $\textsc{Restarts}^{40}_{1\%}$, which in the first phase does 40 short runs with a time limit that is 1% of the total time budget and then uses the remaining 60% of the total time budget to continue the best run.

From a theoretical point of view, the initialisation can have a small beneficial effect even on very easy functions. Sudholt [15] showed that among all evolutionary algorithms initialising μ solutions uniformly at random and then only using standard bit mutation to generate new solutions, the best algorithms for OneMax and LeadingOnes from this class pick the best from the $\mu > 1$ initial individuals and then run a simple (1+1) EA from there. This strategy decreases the expected running time, compared to the classic (1+1) EA, by an additive term of small order. de Perthuis de Laillevault et al. [2] narrowed down the optimal choice of μ for OneMax and proved a speedup by an additive term of $\Theta(\sqrt{n \log n})$. In relative terms, this speedup is not very significant as the expected running time is of much larger order $\Theta(n \log n)$, hence the relative advantage disappears as the problem size increases. In contrast to this, We consider a function class where much larger improvements are possible.

This article is structured as follows. First, we cover the preliminaries in Section 2 and introduce fundamental properties in Section 3. Then, we show that parallel runs are necessary in Section 4, and dig deeper into the choice of parameters in Section 5. Lastly, we provide a fixed budget analysis to guide parameter selection for practical cases in Section 6.

2 PRELIMINARIES

We consider two algorithms augmented by the above bet-and-run strategy, Random Local Search (RLS, Algorithm 2) and the (1+1) evolutionary algorithm ((1+1) EA, which follows the scheme of Algorithm 2, but flips each bit in y independently with probability $1/n$). We analyse the performance of these two algorithms and their augmented variants and are particularly interested in the optimisation time, i. e., the number of fitness evaluations (as opposed to the number of iterations) needed to sample a globally optimal solution. We also consider the expected fitness value after t fitness evaluations.

Algorithm 2 RLS

Choose $x \in \{0,1\}^n$ uniformly at random
repeat
  $y \leftarrow x$. Flip one bit in y chosen uniformly at random
  if $f(y) \geq f(x)$ **then** $x \leftarrow y$
until stop

We consider a fitness function composed of a plateau and a slope, defined by a parameter $\lfloor n/2 \rfloor + 1 < h < n$.

$$f_h(x) = \begin{cases} |x|_1 & \text{if } |x|_1 > n/2 \\ h & \text{otherwise} \end{cases}$$

Note that $\lfloor n/2 \rfloor + 1$ is the lowest fitness value on the slope, hence $\lfloor n/2 \rfloor + 1 < h$ ensures that the Hamming distance between any individual on the plateau and any individual with at least the same fitness on the slope is at least 2. The condition $h < n$ ensures that the global optimum is 1^n, the highest point on the slope.

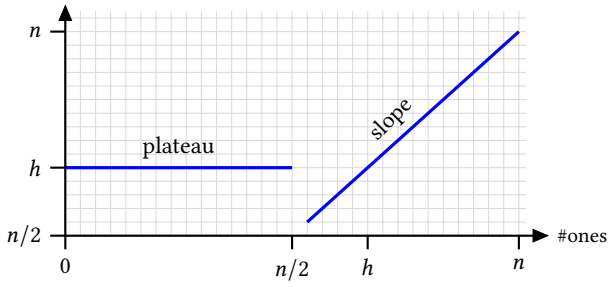

Figure 2: Sketch of the function f_h.

The function is a crude abstraction of more realistic fitness landscapes that contain seemingly promising regions and less promising regions. Here the plateau has a relatively high fitness, compared to the bottom part of the slope, thus it seems to be a promising region. However, no further progress is possible from the plateau (apart from very rare, large mutations), hence the plateau turns out to be deceptive. The slope has a low fitness at first, but it allows individuals to hill-climb to fitness values higher than the plateau, eventually reaching the global optimum.

Deceptivity is one of the features of real-world problems that have been identified in the past as "reasons" for difficulties of evolutionary algorithms—see [16] for an overview on difficulties in optimisation, which furthermore include premature convergence, ruggedness, causality, neutrality, epistasis, and robustness. Weise et al. [16] state that "there are no efficient countermeasures against deceptivity. Using large population sizes, maintaining a very high diversity, and utilizing linkage learning [...] are, maybe, the only approaches which can provide at least a small chance of finding good solutions." In this article, we prove mathematically that *bet-and-run* can be an effective countermeasure.

Note that we designed this function to facilitate a theoretical analysis. Our analyses can be easily generalised to more realistic classes of functions with slopes, valleys, peaks, and so on, however, this is beyond the scope of this article. What matters here is the fitness after t_1 steps.

3 FUNDAMENTAL PROPERTIES

In this section, we prove some useful properties of initialisation using random sampling of individuals from $\{0,1\}^n$, as well as some bounds on the progress made by the RLS and (1+1) EA algorithms on the slope portion of f_h. Note that the slope matches the function $\textsc{OneMax}(x) := \sum_{i=1}^{n} x_i$, which simply counts the number of 1-bits. Some results refer to OneMax for simplicity.

3.1 Initialisation

Search points chosen uniformly at random will have close to $n/2$ bits set to 1. Hence there is a good chance that a population will contain some points on the plateau, and some points on the slope. We make this precise in the following lemmas, which will be used in subsequent theoretical analyses.

We first consider the probability that random initialisation produces an individual on the slope portion of f_h.

Lemma 3.1. *If a point x is sampled uniformly from $\{0,1\}^n$,*

$$P(|x|_1 > n/2) = 1/2 - \Theta(1/\sqrt{n}).$$

Proof. $|x|_1$ is binomially-distributed with parameters n and $p = 1/2$.

If n is odd, the probability of sampling a point on the slope is exactly $1/2$ by symmetry (inverting all bits in any plateau point yields a unique slope point).

If n is even, individuals with $|x|_1 = n/2$ are on the plateau. In this case, the probability of sampling a point on the slope is reduced by $P(|x|_1 = n/2)/2 = \binom{n}{n/2}2^{-1-n} = \frac{n!}{(n/2)!(n/2)!}2^{-1-n} = \Theta(1/\sqrt{n})$. □

With $k > 1$, multiple points are sampled independently to initialise k runs in the bet-and-run strategy. When k is sufficiently large, some key points are sampled with high probability.

Lemma 3.2. *Let k be the number of points sampled uniformly at random from $\{0,1\}^n$, then:*

1. *At least one point with $|x|_1 \leq n/2$ (i.e. on the plateau) is sampled with probability at least $1 - 2^{-k}$,*
2. *At least one point with $|x|_1 \geq n/2 + \Omega(\sqrt{n})$ is sampled with probability at least $1 - (3/4)^k$,*
3. *If $k \leq \text{poly}(n)$, no points with $|x|_1 \geq n/2 + \sqrt{n}\log n$ are sampled with probability $1 - n^{-\Omega(\log n)}$.*

Proof. The first statement follows from the fact that for a search point x chosen uniformly at random, $P(|x|_1 \leq n/2) \geq 1/2$. The probability that this applies to at least one search point is $1 - 2^{-k}$.

For the second statement, Lemma 1 in [10] shows that for a search point x chosen uniformly at random,

$$P\left(|x|_1 \geq n/2 + \pi/(4e)\cdot\sqrt{n}\right) \geq 1/4.$$

The probability that at least one such point is sampled is $1 - (3/4)^k$.

For the final statement, we note that the expected number of 1-bits in a randomly sampled point is $\mu = n/2$, and use a Chernoff bound with $\delta = 2\log(n)/\sqrt{n}$, to bound

$$P(|x|_1 \geq (1+\delta)\mu) \leq e^{-\delta^2\mu/3} = e^{-4\log^2 n/6} = n^{-\Omega(\log n)}$$

and then apply a union bound to show that for any $k \leq \text{poly}(n)$, i.e., $k \leq n^{c_3}$ for any constant c_3, no point with at least $n/2 + \sqrt{n}\log n$ 1-bits is sampled with probability at least

$$\left(1 - P(|x|_1 \geq n/2 + \sqrt{n}\log n)\right)^{n^{c_3}} \geq 1 - n^{c_3}\cdot n^{-\Omega(\log n)} = 1 - n^{-\Omega(\log n)}.$$

□

Evolving an individual on the slope, from a population contained on the plateau, requires an exponential amount of time if using standard bit mutation, and is impossible if using RLS.

Lemma 3.3. *Consider RLS or the (1+1) EA with or without bet-and-run initialisation. If all current search points are on the plateau, the expected time to evolve a point of greater or equal fitness is at least $(h - n/2)!$ if using the Standard Bit Mutation operator, and infinite if using Random Local Search.*

Proof. For Random Local Search, we note that it is impossible to escape from the plateau by applying local mutations, as all non-plateau neighbours of any point on the plateau have strictly worse fitness than the plateau.

In case of the (1+1) EA, any number of bits can be flipped in a single iteration. In the best case, flipping $h - n/2$ bits (to go from an individual with $|x|_1 = n/2$ to $|x|_1 = h$) is required; the probability that such a mutation occurs is at most $\binom{n}{h-n/2}n^{-h+n/2} \leq 1/(h-n/2)!$, and thus the expected waiting time for such a mutation to occur is at least $(h - n/2)!$. □

3.2 Progress Estimates

We further provide estimates for the progress on the slope.

Lemma 3.4. *Suppose $|x_0|_1 \geq \lfloor n/2\rfloor + 1$, and let $T_{>h}$ be the first time of RLS on OneMax, starting in x_0, hitting a OneMax-value larger than h, for $1/2 < h/n < 1$ constant. Then*

$$P(T_{>h} \leq n\cdot\ln(n/(2n-2h)) + n^{3/4}) \geq 1 - e^{-\Omega(\sqrt{n})}.$$

Proof. In the following, $H(n) := \sum_{i=1}^{n} 1/i$ denotes the n-th harmonic number. We will show the following inequalities for a suitable value $\delta = \Omega(n^{3/4})$ specified later:

$$\begin{aligned} &P\left(T_{>h} \leq n\cdot\ln(n/(2n-2h)) + n^{3/4}\right) \\ &\geq P\left(T_{>h} \leq \sum_{i=\lfloor n/2\rfloor+1}^{h}\frac{n}{n-i} + \delta\right) \qquad (1) \\ &\geq 1 - \exp\left(-\frac{\delta}{4}\cdot\min\left\{\frac{\delta}{\sum_{i=\lfloor n/2\rfloor+1}^{n-h}\frac{n^2}{(n-i)^2}}, \frac{n}{h}\right\}\right) = 1 - e^{-\Omega(\sqrt{n})}. \qquad (2) \end{aligned}$$

The last equality follows from $\sum_{i=\lfloor n/2\rfloor+1}^{n-h}\frac{n^2}{(n-i)^2} \leq \frac{n^3}{(n-h)^2} = O(n)$ (using $h/n < 1$ constant) and $\delta = \Omega(n^{3/4})$. Inequality (2) follows from the upper tail bound for fitness levels [17, Theorem 2] as on OneMax the probability of improving a fitness of i is $(n-i)/n$.

To show (1), note that

$$\sum_{i=\lfloor n/2\rfloor+1}^{h}\frac{n}{n-i} = \sum_{i=n-h}^{\lceil n/2\rceil-1}\frac{n}{i} \leq n\cdot(H(\lfloor n/2\rfloor) - H(n-h-1)).$$

Using $H(n) = \ln(n) + \gamma + O(1/n)$, for $\gamma \approx 0.5772156649$ the Euler-Mascheroni constant, we get that

$$\begin{aligned} n \cdot (H(\lfloor n/2 \rfloor) - H(n-h-1)) &= n \cdot \left(H(\lfloor n/2 \rfloor) - H(n-h) + \frac{1}{n-h} \right) \\ &\leq n \cdot (\ln(n/2) - \ln(n-h) + O(1/n)) \\ &= n \cdot \ln(n/(2n-2h)) + O(1). \end{aligned}$$

Defining $\delta = n^{3/4} - O(1)$, we get

$$n \cdot \ln(n/(2n-2h)) + n^{3/4} \geq \sum_{i=\lfloor n/2 \rfloor + 1}^{h} \frac{n}{n-i} + \delta$$

which proves (1) and the claim. □

The same method can be applied to the (1+1) EA. The proof is ommited due to space restrictions.

Corollary 3.5. *Suppose $|x_0|_1 \geq \lfloor n/2 \rfloor + 1$, and let $T_{>h}$ be the first time of the (1+1) EA on OneMax, starting in x_0, hitting a OneMax-value larger than h, for $1/2 < h/n < 1$ constant. Then*

$$P(T_{>h} \leq en \cdot \ln(n/(2n-2h)) + n^{3/4}) \geq 1 - e^{-\Omega(\sqrt{n})}$$

Lemma 3.6. *Suppose $\lfloor n/2 \rfloor + 1 \leq |x_0|_1 \leq \lfloor n/2 \rfloor + O(\sqrt{n} \log n)$, and let $T_{\geq h}$ be the first time of RLS on OneMax, starting in x_0, hitting a OneMax-value of at least h, for $1/2 < h/n < 1$ constant. Then*

$$P(T_{\geq h} \geq n \cdot \ln(n/(2n-2h)) - n^{3/4}) \geq 1 - e^{-\Omega(\sqrt{n})}$$

Lemma 3.6 can be proven similarly to Lemma 3.4, using the lower tail bound for fitness levels [17, Theorem 2]. The proof is omitted due to space restrictions.

4 PARALLEL RUNS ARE NECESSARY

We now prove that for $h = 3n/4$, a bet-and-run strategy achieves a polynomial running time on f_h with high probability if t_1 and k are sufficiently large, while simple iterated random sampling (i.e., $t_1 = 1$) is inefficient with high probability. These results hold for both RLS and (1+1) EA.

Theorem 4.1. *If $t_1 = 1$, the expected running time of the $(1+1)$ EA, initialised with the best of k randomly-sampled points, on $f_h(x)$ with $h = 3n/4$ is at least $n^{\Omega(n)}$ for any polynomial choice of k.*

The expected running time of RLS on $f_h(x)$ with $h = 3n/4$ is infinite for any choice of k when $t_1 = 1$. If k is polynomial w.r.t. n, RLS is not able to find the global optimum with probability at least $1 - 2^{-k} - o(1)$.

Proof. The algorithm chooses the best of k points sampled uniformly at random to evolve further. We note that by the symmetry of the binomial distribution, a point on the plateau is among the initial samples with probability at least $1 - 2^{-k}$; unless a slope point with at least $3n/4$ 1-bits is sampled, a plateau point will be chosen if one is sampled.

By Lemma 3.2, with high probability, no points with $|x|_1 \geq n/2 + \sqrt{n} \log n$ are sampled for any polynomial k that is polynomial with respect to n. This means that it is highly unlikely that an individual with at least $3n/4$ 1-bits is sampled.

Combined, the initialisation process produces an individual on the plateau with probability $(1 - 2^{-k})(1 - n^{-\Omega(\log n)})$. When initialised on the plateau, by Lemma 3.3, the expected optimisation time for the (1+1) EA is at least $n^{\Omega(n)}$.

By the law of total expectation, the expected running time is

$$E(T) \geq (1 - 2^{-k})(1 - n^{-\Omega(\log n)}) n^{\Omega(n)} = n^{\Omega(n)}$$

for the (1+1) EA with iterated random sampling, and infinite for RLS with iterated random sampling, where any polynomial number of independent uniform samples k are used to initialise the algorithm. □

As an interesting consequence, we note that for $h = 3n/4$, $k = 1$ would perform better than any other polynomial choice of k when $t_1 = 1$: increasing the number of samples performed during initialisation increases the probability that a plateau individual will be sampled and accepted.

On the other hand, if, in addition to sampling a modest number of initial search points, those samples were allowed to evolve in parallel for a sufficient number of iterations, it would be possible for islands initialised on the slope to climb it and discover individuals of higher fitness than the plateau.

Theorem 4.2. *A bet-and-run strategy with $k \geq c \log n$, where c is an appropriately-chosen constant, and $t_1 = en \ln(2) + n^{3/4}$, using either RLS or the (1+1) EA on the islands, is able to construct the optimum of $f_{3n/4}$ in time $O(n \log n + kn)$ with high probability.*

Proof. Using Lemma 3.2, we note that there exists a constant c such that with high probability, when $k = c \log n$, at least one island is initialised with $|x|_1 \geq n/2 + \Omega(\sqrt{n})$ one bits. We focus on the progress made on this island during the first t_1 iterations.

For RLS, it is impossible for the island to construct a plateau individual when initialised with $|x|_1 > \lfloor n/2 \rfloor + 1$, so the island remains on the slope for t_1 iterations, and follows the climbing behaviour analysed in Lemma 3.4. Setting $t_1 \geq n \ln(2) + n^{3/4}$ is therefore sufficient for the island to construct an individual with higher-than-plateau fitness in t_1 iterations with high probability.

If the islands are running the (1+1) EA, per Corollary 3.5, after $t_1 = en \ln(2) + n^{3/4}$ iterations, an island which is initialised on the slope, and does not revert to the plateau, will with high probability have a best-so-far individual with fitness higher than $3n/4$. Unlike RLS, it is possible for (1+1) EA islands to revert to the plateau at any point prior to reaching an individual with fitness greater than h, by flipping sufficiently many 1-bits bits in a single mutation. For an island initialised with $|x|_1 > n/2 + \Omega(\sqrt{n})$, constructing a point on the plateau will require flipping at least $\Omega(\sqrt{n})$ 1-bits; while at any time when such a mutation would be accepted, there are no more than $3n/4$ 1-bits in the current individual. Thus, the probability that such a mutation occurs in one iteration and is accepted is at most $(3/4)^{\Omega(\sqrt{n})}$. Taking a union bound over the entire first phase of the bet-and-run strategy, the probability that the island we are focusing on remains on the slope is at least $(1 - (3/4)^{\Omega(\sqrt{n})})^{t_1} \geq 1 - O(n) \cdot (3/4)^{\Omega(\sqrt{n})} = 1 - 2^{-\Omega(\sqrt{n})}$.

Thus, with appropriate choices of k and t_1, a better-than-plateau individual will exist on at least one island after t_1 iterations with high probability. This individual is chosen as the winner by the bet-and-run strategy, and given that reverting to the plateau is no longer possible for either algorithm, the 1^n optimum is found in at most an additional $O(n \log n)$ iterations with high probability, following well-known results for OneMax [17]. Thus, by setting $k = c \log n$, where c is a sufficiently large constant, and $t_1 = en \ln(2) + n^{3/4}$,

after combined $O(kt_1 + n \log n) = O(n \log n)$ fitness evaluations, the global optimum will have been constructed with high probability. □

5 ON THE CHOICE OF PARAMETERS

How to choose parameter values such as t_1? The answer depends on the function at hand: t_1 should be large enough to allow the algorithm to detect whether a search point was initialised in a promising region of the search space. For our function class f_h this depends on the parameter h, which determines both the fitness of the plateau and the distance the algorithm has to travel up the slope from the typical initialisation around $n/2$ ones.

We show that if t_1 is too small, i. e., smaller than the expected time to climb up a distance of h then the algorithm is inefficient. For simplicity we focus on RLS only, however the same effect also occurs for the (1+1) EA.

Theorem 5.1. *Consider RLS with a bet-and-run strategy using parameters $k \leq \text{poly}(n)$ on f_h. If $t_1 \leq (1-\varepsilon)n \ln(n/(2n-2h))$ for any constant $\varepsilon > 0$, then, with probability $1 - 2^{-k} - e^{-\Omega(\sqrt{n})}$, the algorithm fails to find a global optimum.*

Proof. Note that once an island reaches the plateau, RLS can never escape. So we only have to consider islands that initialise on the slope. For n large enough (otherwise the claim is trivial),

$$t_1 \leq (1-\varepsilon)n \ln(n/(2n-2h)) < n \ln(n/(2n-2h)) - n^{3/4}.$$

By Lemma 3.6, the probability that any island initialised on the slope reaches a fitness of at least h in t_1 generations is $e^{-\Omega(\sqrt{n})}$. Taking the union bound over at most k islands on the slope, the probability is still only of order $k \cdot e^{-\Omega(\sqrt{n})}$. As $k \leq \text{poly}(n) = n^{O(1)} = e^{O(\log n)}$ this simplifies to $e^{O(\log n)} \cdot e^{-\Omega(\sqrt{n})} = e^{-\Omega(\sqrt{n})}$.

If no island reaches fitness h at time t_1, an individual on the plateau will survive, and all other islands will be removed. By Lemma 3.2, the probability of initialising on the plateau is at least $1 - 2^{-k}$. Taking a union bound over failure probabilities $e^{-\Omega(\sqrt{n})}$ and 2^{-k} proves the claim. □

If t_1 is large enough, we can guarantee that the global optimum is found with high probability:

Theorem 5.2. *Consider RLS with a bet-and-run strategy using parameters $k \leq \text{poly}(n)$ on f_h. If $t_1 \geq (1+\varepsilon)n \ln(n/(2n-2h))$ for any constant $\varepsilon > 0$ then with probability at least $1 - (3/4)^k - O(1/n)$ the algorithm finds a global optimum in time $O(kn \log n)$.*

Proof. By Lemma 3.2 with probability $1 - (3/4)^k$ there is at least one island initialised on the slope, with a sufficient distance to the plateau such that it can never reach the plateau. Fix such an island, then for n large enough (otherwise the claim is trivial),

$$t_1 \geq (1+\varepsilon)n \ln(n/(2n-2h)) \geq n \cdot \ln(n/(2n-2h)) + n^{3/4}.$$

By Lemma 3.4 the island has reached a fitness larger than h after t_1 steps, with probability $1 - e^{-\Omega(\sqrt{n})}$. This means that this, or another island on the slope will survive after time t_1. The time bound follows from the fact that RLS optimises OneMax and hence the slope in expected time $O(n \log n)$ with high probability $1 - O(1/(kn))$ [3] (the failure probability can be as small as an inverse polynomial of arbitrarily large degree), and using a union bound over k islands again, so at most $O(kn \log n)$ function evaluations are needed with probability at least $1 - O(1/n)$. Adding all failure probabilities and absorbing $e^{-\Omega(\sqrt{n})}$ in the term $O(1/n)$ proves the claim. □

6 EXPECTED FITNESS

As seen in the previous section, if t_1 is too small, RLS with a bet-and-run strategy is trapped on the plateau with high probability, resulting in a fitness of h. If t_1 is large enough to ensure that the best island after t_1 steps is on the slope, RLS with a bet-and-run strategy optimises f_h in time $O(kn \log n)$. However, if one is given an overall budget of t steps with the goal of maximising the expected fitness after t steps, choosing too large t_1 will waste computation time in Phase 1 of the algorithm. It is therefore interesting to consider the expected fitness for some budget $t = k \cdot t_1 + t_2$ – such analysis is known as fixed budget analysis [9]. We will use this perspective to show that too large values for t_1 lead to a decrease in the expected final fitness.

6.1 A Single Lineage of RLS

Let us first consider a single lineage of RLS on $f_h(x)$ in the fixed budget setting. We consider three different cases and denote the initial search point by x_0.

(1) RLS is initialised on the plateau, i. e., $|x_0|_1 \leq n/2$. This happens with probability $\Theta(1/\sqrt{n})$ (following similar arguments as in Lemma 3.1). In this case, RLS will stay on the plateau forever.
(2) RLS is initialised on the left-most point of the slope, i. e., $|x_0|_1 = \lfloor n/2 \rfloor + 1$. This happens with probability $\Theta(1/\sqrt{n})$ (following similar arguments as in Lemma 3.1). In this case, the first iteration will determine if we continue on the slope or reach the plateau: Flipping a 1-bit will lead to a point on the plateau; flipping a 0-bit ensures that RLS can climb up the slope.
(3) RLS is initialised further up on the slope, i. e., $|x_0|_1 > \lfloor n/2 \rfloor + 1$. This happens with probability $\Theta(1/\sqrt{n})$ (following similar arguments as in Lemma 3.1). In this case, RLS will climb up the slope and eventually reach the optimum.

We consider Case (1) and (3) in more detail in Lemma 6.1 and 6.3 and use these two results to derive a statement for Case (2) in Lemma 6.4. For the sake of simplicity we assume that n is even. The case for odd n can be proved similarly.

Lemma 6.1 (Case 1). *Let x_t be the search point after t iterations of a single lineage of RLS. The expected fitness value of x_t conditional on initialising on the plateau is:*

$$E(f(x_t) \mid |x_0|_1 \leq n/2) = h$$

for all $t \geq 0$ and $n/2 + 1 < h < n$.

Proof. For all $x \in \{0,1\}^n$ with $|x|_1 \leq n/2$, we have $f_h(x) = h > n/2 + 1$. For $y \in \{0,1\}^n$ with $|y|_1 = n/2 + 1$ (a leftmost point on the slope), we have $f(y) = n/2 + 1 < h = f_h(x)$. Since RLS only performs 1-bit flips, it will be trapped on the plateau forever. □

Before examining the other two cases, we consider the expected fitness conditional on initialising with exactly i bits safely on the slope, i. e., $i > n/2 + 1$. Note, that i needs to be an integer, however, for the sake of readability we omit $\lfloor \ldots \rfloor$ in the following.

Lemma 6.2. *The expected fitness of one lineage of RLS starting in x_0 with $i = n/2 + g(n) \leq n$ 1-bits for some $g(n)\colon \mathbb{N}^+ \to \mathbb{R}^+$ and $g(n) \geq 2$ for all n, after t iterations, is*

$$E(f(x_t) \mid |x_0| = i) = n - (n/2 - g(n)) \cdot (1 - 1/n)^t$$

Proof. As discussed before $f_h(x)$ is equivalent to the well-known OneMax problem on the slope. We therefore follow the line of thought of Theorem 5 in [9] and observe that

$$E(f(x_t) \mid |x_0| = i) = i + (n - i)\left(1 - (1 - 1/n)^t\right)$$

holds as a bit initially set to 0 has the value 1 at time t if and only if there is a point of time when this specific bit is flipped. Using $i = n/2 + g(n)$ a straightforward calculation leads to

$$\begin{aligned}
&E(f(x_t) \mid |x_0| = i)\\
&= n/2 + g(n) + (n - (n/2 + g(n)))\left(1 - (1 - 1/n)^t\right)\\
&= n/2 + g(n) + (n/2 - g(n))\left(1 - (1 - 1/n)^t\right)\\
&= n/2 + g(n) + (n/2 - g(n)) - (n/2 - g(n)) \cdot (1 - 1/n)^t\\
&= n - (n/2 - g(n)) \cdot (1 - 1/n)^t
\end{aligned}$$

□

Using Lemma 6.2 we can now derive bounds on the expected fitness assuming that RLS is initialised in some point that high enough on the slope.

Lemma 6.3 (Case 3). *Let x_t be the search point after t iterations of a single lineage of RLS. The expected fitness value of x_t for all $t \geq 0$ conditional on initialising safely on the slope is:*

$$\begin{aligned}
n - (n/2 - 2) \cdot (1 - 1/n)^t &\leq E(f(x_t) \mid |x_0|_1 > n/2 + 1)\\
&\leq n - \left(n/2 - \sqrt{n}\log n\right) \cdot (1 - 1/n)^t + n \cdot n^{-\Omega(\log n)}
\end{aligned}$$

Proof. We need to consider the conditional expectation for all $i > n/2 + 1$. For the lower bound we assume that $|x_0|_1 = n/2 + 2$, the lowest point on the slope that ensures that RLS cannot move onto the plateau. Using Lemma 6.2 with $g(n) = 2$ we obtain:

$$\begin{aligned}
&E(f(x_t) \mid |x_0|_1 > n/2 + 1) \geq E(f(x_t) \mid |x_0|_1 = n/2 + 2)\\
&= n - (n/2 - 2) \cdot (1 - 1/n)^t
\end{aligned}$$

From the proof of Lemma 3.2 we know that $|x_0|_1 < n/2 + \sqrt{n}\log n$ with probability $1 - n^{-\Omega(\log n)}$. Using Lemma 6.2 with $g(n) = \sqrt{n}\log n$ we get

$$\begin{aligned}
&E(f(x_t) \mid |x_0|_1 > n/2 + 1)\\
&= \left(1 - n^{-\Omega(\log n)}\right) \cdot E\left(f(x_t) \mid n/2 + 1 < |x_0|_1 \leq n/2 + \sqrt{n}\log n\right)\\
&\quad + n^{-\Omega(\log n)} \cdot E\left(f(x_t) \mid |x_0|_1 > n/2 + \sqrt{n}\log n\right)\\
&\leq E\left(f(x_t) \mid |x_0|_1 = n/2 + \sqrt{n}\log n\right) + n \cdot n^{-\Omega(\log n)}\\
&= n - \left(n/2 - \sqrt{n}\log n\right) \cdot (1 - 1/n)^t + n \cdot n^{-\Omega(\log n)}
\end{aligned}$$

□

Finally, we consider the borderline case where $|x_0|_1 = n/2 + 1$.

Lemma 6.4 (Case 2). *Let x_t be the search point after t iterations of a single lineage of RLS. The expected fitness value of x_t conditional on initialising on the leftmost point on the slope is:*

$$\begin{aligned}
n/2 + h/2 - (n/4 - 1) \cdot (1 - 1/n)^{t-1} - 1/2&\\
\leq E(f(x_t) \mid |x_0|_1 = n/2 + 1) &\leq\\
n/2 + h/2 - (n/4 - 2) \cdot (1 - 1/n)^{t-1}&
\end{aligned}$$

Proof. For $|x_0|_1 = n/2 + 1$, the probability to move onto the plateau in the next step is $(n/2+1)/n = 1/2+1/n$ and the probability to walk up the slope is $(n/2 - 1)/n = 1/2 - 1/n$. Thus, we have

$$\begin{aligned}
&E(f(x_t) \mid |x_0|_1 = n/2 + 1)\\
&= (1/2 + 1/n) \cdot E(f(x_t) \mid |x_1|_1 = n/2)\\
&\quad + (1/2 - 1/n) \cdot E(f(x_t) \mid |x_1|_1 = n/2 + 2)\\
&= (1/2 + 1/n) \cdot h + (1/2 - 1/n) \cdot \left(n - (n/2 - 2) \cdot (1 - 1/n)^{t-1}\right)\\
&= n/2 + h/2 - (n/4 - 3/2 + 2/n) \cdot (1 - 1/n)^{t-1} + h/n - 1
\end{aligned}$$

This can easily be bounded above and below by the terms stated in the lemma. □

We can now use the above results to derive the desired fixed budget result.

Theorem 6.5. *Let x_0 be uniformly at random from $\{0,1\}^n$.*

$$\begin{aligned}
\frac{n}{2} + \frac{h}{2} - \left(\frac{n}{4} - 1\right) \cdot \left(1 - \frac{1}{n}\right)^t &\leq E(f(x_t))\\
&\leq \frac{n}{2} + \frac{h}{2} - \left(\frac{n}{4} - \frac{1}{2}\sqrt{n}\log n\right) \cdot \left(1 - \frac{1}{n}\right)^t + \Theta(\sqrt{n})
\end{aligned}$$

Proof. The result can easily be obtained by combining the results from Lemma 6.1, 6.3 and 6.4. The calculations are omitted due to space restrictions. □

One could argue that the result in Theorem 6.5 is somewhat misleading as it combines two very different types of RLS runs on $f_h(x)$: If RLS is initialised on the plateau, the observed fitness does not change at all, but our bounds on the expected fitness increase with increasing t (until a certain point) due to the influence of the behaviour on the slope. Similarly, the relatively high value on the plateau has a significant influence on the initial fitness value and thus, if RLS is initialised on the slope the observed fitness values may be much lower than the above expectation. Finally, if the plateau is not very high, the expected fitness is always much smaller than the optimal fitness value. We therefore argue that the results as presented in Lemmas 6.1, 6.3 and 6.4 together with the probabilities for the corresponding events may be a more appropriate way to extend fixed budget analysis to multimodal problems as it makes a more precise statement about the expected fitness.

We remark that this is somewhat similar to the discussion of limitations of running time analysis: An expected exponential optimisation time (or in the case for RLS an infinite optimisation time) for some function can be misleading if the worst case is very unlikely and thus, the optimisation time is in fact polynomial with high probability. In such cases, we usually use more meaningful statements such as "the optimisation time is T with probability p" instead of the expected optimisation time [8]. We argue that a similar approach should be taken in the context of fixed budget analysis where appropriate.

6.2 Bet-And-Run Strategy

We now use the results of the previous section to derive bounds on the expected fitness for RLS with a bet-and-run strategy using parameters $t = k \cdot t_1 + t_2$. We restrict our investigations to the case where t_1 is large enough to optimise f_h in polynomial time with high probability and demonstrate how the expected fitness decreases with increasing t_1 once t_1 grows beyond a certain threshold.

Theorem 6.6. *Consider RLS with a bet-and-run strategy using parameters $c \log n \leq k \leq \text{poly}(n)$ on f_h for a large enough constant $c > 0$ and $t_1 \geq (1+\varepsilon)n\ln(n/(2n-2h))$ for any constant $\varepsilon > 0$. Then, the expected fitness after $t = k \cdot t_1 + t_2$ steps is*

$$n - \left(\frac{n}{2} - d\sqrt{n}\right) \cdot \left(1 - \frac{1}{n}\right)^{t-(k-1)\cdot t_1} - \left(\frac{3}{4}\right)^k \cdot n \leq E(f(x_t))$$
$$\leq (1+\delta) \cdot \left(n - \left(\frac{n}{2} - \sqrt{n}\log n\right) \cdot \left(1 - \frac{1}{n}\right)^{t-(k-1)\cdot t_1}\right) + o(1)$$

for all $t \geq 0$ and $d, \delta > 0$ constant.

Proof. We first observe that for an overall budget of $t = k \cdot t_1 + t_2$ a single island will get $t_1 + t_2$ steps and thus, we need to bound the progress that can be made in this many steps.

According to Lemma 3.4 and Theorem 5.2, the best island after $t_1 \geq (1+\varepsilon)n\ln(n/(2n-2h))$ steps will be on the slope with probability $1 - e^{-\Omega(\sqrt{n})}$. In this case, the expected fitness after t steps is at least $E(f(x_{t_1+t_2}) \mid |x_0|_1 > n/2 + 1)$, i. e., the expected fitness of an individual initialised on the slope after $t_1 + t_2$ steps. Otherwise, the fitness will be h.

According to Lemma 3.2, there will be at least one point with $|x_0|_1 \geq n/2 + d\sqrt{n}$ ($d > 0$ some appropriately chosen constant) with probability $1 - (3/4)^k$. Thus, following the line of thought of Lemma 6.3, we get $n - \left(n/2 - d\sqrt{n}\right) \cdot (1-1/n)^{t_1+t_2} - (3/4)^k \cdot n$ as lower bound on the expected fitness after $t_1 + t_2$ steps. Note, that the other failure cases are dominated by the $(3/4)^k \cdot n$ term.

For the upper bound, we need to consider the island with the best fitness after t_1 steps. According to Lemma 6.3 the expected fitness for an island on the slope after t_1 steps is at most $n - \left(n/2 - \sqrt{n}\log n\right) \cdot (1-1/n)^{t_1} + n \cdot n^{-\Omega(\log n)}$. The probability to deviate from this by more than a factor of $(1+\delta)$, $\delta > 0$ an arbitrary constant, in k independent runs is $e^{-\Omega(\sqrt{n})}$ using Lemma 3.6 and similar arguments as in the proof of Theorem 5.1 (details omitted). To get an upper bound for the complete budget, we use the same argument for $t_1 + t_2$ steps. Failure cases can be accounted for in a similar way as for the lower bound.

Using $t_1 + t_2 = t_1 + (t - k \cdot t_1) = t - (k-1) \cdot t_1$ yields the result. □

To make the result more tangible we visualise the two bounds from Theorem 6.6 in Figure 3 using $n = 100$, $t = 1000$ and $k = 10$. We see that the expected fitness after t steps decreases with increasing t_1, showing that too large t_1 can waste fitness evaluations.

6.3 Experimental Supplements

As our proofs do not reveal which *bet-and-run* configuration is the one that performs best, we conduct experiments to show the effect the plateau height h has on the performance landscape (see

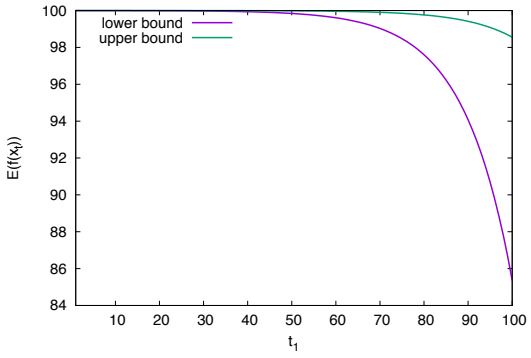

Figure 3: Visualisation of bounds from Theorem 6.6.

Figure 4). Note that in our setup with $n = 100$ in the experiments requires $h \geq 52$ so that we have a local minimum that cannot be crossed by RLS. We include $h = 50$ nevertheless to show the situation when no such local optima exist.

We can observe significant qualitative changes. First, when $h = 52$, the best choice is to do several short runs in the first phase. This recommendation still holds as the total budget increases, and in fact many more *bet-and-run* configurations turn out to perform similarly well. Second, when $h = 62.5$, there are three regions of best performing configurations (the two visible "bumps" and a thin ridge along $k = 1$), of which two merge to a major ridge as the total computation budget increases. For $h = 75$ and $h = 87.5$, there is the thin ridge along $k = 1$, and only one major region that increases in size with increasing total budget.

Interestingly, when the total budget is only n, the best *bet-and-run* configuration for the three rightmost cases is the one where only a single run is performed; small ridges along $k = 1$ are present and just visible in the plots. When $h = 52$, the best strategy is to have several short runs and to proceed with Phase 2 from the best of these short runs. This recommendation does not hold for the other three plateau heights, where performing several short runs in Phase 1 often results in the worst performance. Instead, either several long runs should be performed when the total budget increases ($h = 87.5$ or $h = 75$) or even a wide range of *bet-and-run* configurations performs well ($h = 62.5$).

In summary, we can see that the choice of the best-performing k-t_1-combination depends on the problem and the overall budget.

7 CONCLUSIONS

In this article, we have proven mathematically that *bet-and-run* can be an effective countermeasure when a problems with promising and deceptive regions are encountered. We conjecture that a similar result holds for functions where the deceptive area is not a plateau, but a slope towards a local optimum. However, in this case parameterisation and benefit will depend not only on the offset h, but also on the gradient of the slope.

We also show that the choice of t_1 is linked to properties of the function, and we provide a fixed budget analysis to guide the selection of parameters. A natural next step is to extend our analyses from bimodal functions to multi-modal ones. In addition, we plan to characterise the progress variance of individual runs (in Phase 1) theoretically, so that this can be exploited in proofs and in practice.

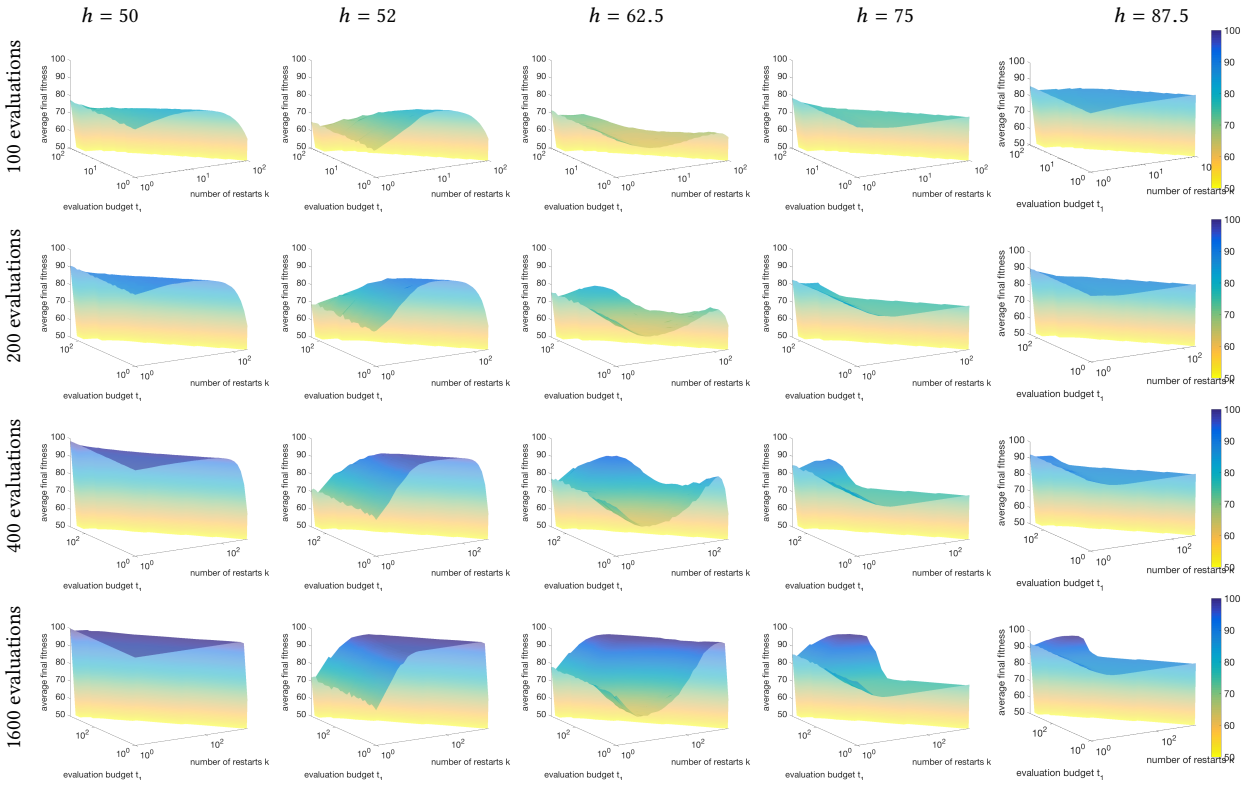

Figure 4: RLS: effect of k and t_1 on average fitness achieved on $f_h(x)$ given various computation budgets and various plateau heights. $n = 100$, resulting in minimal fitness of 50 and maximum fitness of 100. We consider 30 values of k and t_1 each, spaced out logarithmically. Shown are the averages of 1000 independent runs. As we can see, the choice of the best-performing k-t_1-combination depends on the problem and the overall budget. Note that the yellow plummeting face is a plotting artefact.

ACKNOWLEDGMENTS

This work has been supported by the ARC Discovery Early Career Researcher Award DE160100850, by a Priority Partner Grant of the University of Adelaide, and by the EPSRC under Grant n. EP/M004252/1. 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). Thanks to Jorge Pérez Heredia, Martin Krejca, and Aneta Neumann for initial discussions and to Wil Ward and Wenwen Li for their technical support. This article is based upon work from COST Action CA15140 'Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO)' supported by COST (European Cooperation in Science and Technology).

REFERENCES

[1] Armin Biere. Adaptive restart strategies for conflict driven SAT solvers. In *Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 28–33, 2008.
[2] Axel de Perthuis de Laillevault, Benjamin Doerr, and Carola Doerr. Money for nothing: Speeding up evolutionary algorithms through better initialization. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 815–822, 2015.
[3] Benjamin Doerr and Leslie Ann Goldberg. Drift analysis with tail bounds. In *Parallel Problem Solving from Nature (PPSN XI)*, pages 174–183. Springer, 2010.
[4] Matteo Fischetti and Michele Monaci. Exploiting erraticism in search. *Operations Research*, 62(1):114–122, 2014.
[5] Tobias Friedrich, Timo Kötzing, and Markus Wagner. A generic bet-and-run strategy for speeding up stochastic local search. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 801–807, 2017.
[6] Carla P. Gomes, Bart Selman, Nuno Crato, and Henry A. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24(1):67–100, 2000.
[7] Jinbo Huang. The effect of restarts on the efficiency of clause learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI)*, pages 2318–2323, 2007.
[8] Thomas Jansen. *Analyzing Evolutionary Algorithms. The Computer Science Perspective*. Springer, 2013.
[9] Thomas Jansen and Christine Zarges. Performance analysis of randomised search heuristics operating with a fixed budget. *Theor. Comput. Sci.*, 545:39–58, 2014.
[10] Timo Kötzing, Dirk Sudholt, and Madeleine Theile. How crossover helps in pseudo-Boolean optimization. In *Proceedings of the 13th Genetic and Evolutionary Computation Conference (GECCO 2011)*, pages 989–996. ACM Press, 2011.
[11] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. Iterated local search: Framework and applications. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, pages 363–397. Springer US, 2010.
[12] Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47(4):173–180, 1993.
[13] Rafael Marti. Multi-start methods. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 355–368. 2003.
[14] Marc Schoenauer, Fabien Teytaud, and Olivier Teytaud. A rigorous runtime analysis for quasi-random restarts and decreasing stepsize. In *Artificial Evolution*, pages 37–48. Springer, 2012.
[15] Dirk Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Trans. on Evol. Computation*, 17(3):418–435, 2013.
[16] Thomas Weise, Michael Zapf, Raymond Chiong, and Antonio J. Nebro. Why is optimization difficult? In Raymond Chiong, editor, *Nature-Inspired Algorithms for Optimisation*, pages 1–50. Springer, 2009.
[17] Carsten Witt. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Information Processessing Letters*, 114(1-2):38–41, 2014.