Heavy-tailed Mutation Operators in Single-Objective Combinatorial Optimization

Tobias Friedrich[1,2], Andreas Göbel[1], Francesco Quinzan[1], and Markus Wagner[2]

[1] Hasso Plattner Institute, Potsdam, Germany
[2] University of Adelaide, Adelaide, Australia

Abstract. A core feature of evolutionary algorithms is their mutation operator. Recently, much attention has been devoted to the study of mutation operators with dynamic and non-uniform mutation rates. Following up on this line of work, we propose a new mutation operator and analyze its performance on the (1+1) Evolutionary Algorithm (EA). Our analyses show that this mutation operator competes with pre-existing ones, when used by the (1+1) EA on classes of problems for which results on the other mutation operators are available. We present a "jump" function for which the performance of the (1+1) EA using any static uniform mutation and any restart strategy can be worse than the performance of the (1+1) EA using our mutation operator with no restarts. We show that the (1+1) EA using our mutation operator finds a (1/3)-approximation ratio on any non-negative submodular function in polynomial time. This performance matches that of combinatorial local search algorithms specifically designed to solve this problem.
Finally, we evaluate experimentally the performance of the (1+1) EA using our operator, on real-world graphs of different origins with up to ∼37 000 vertices and ∼1.6 million edges. In comparison with uniform mutation and a recently proposed dynamic scheme our operator comes out on top on these instances.

Keywords: Mutation operators, Combinatorial optimization, Restarts

1 Introduction

One of the simplest and most studied *evolutionary algorithm* is the (1+1) EA [5, 22, 26] (see Algorithm 1). A key parameter of the (1+1) EA that affects its performance is the *mutation operator*, i.e., the operator that determines at each step how the potential new solution is generated. In the past several years there has been a huge effort, both from a theoretical and an experimental point of view, towards understanding how this parameter influences the performance of the (1+1) EA and towards deciding which is the optimal way of choosing this parameter (e.g., see [6, 7]).

The most common mutation operator on n-bit strings is the static *uniform mutation* operator. This operator, unif_p, flips each bit of the current solution independently with probability $p(n)$. This probability, $p(n)$ is called static *mutation rate* and remains the same throughout the run of the algorithm. The most common choice for $p(n)$ is $1/n$; thus, mutated solutions differ in expectation in 1-bit from their predecessors. Witt [27] has shown that this choice of $p(n)$ is optimal for all pseudo-Boolean linear functions. Doerr et al. [3] further observe that changing $p(n)$ by a constant factor can lead to large variations of the overall run-time of the (1+1) EA. They also show the existence of functions for which this choice of $p(n)$ is not optimal.

Static mutation rates are not the only ones studied in literature. Jansen et al. [16] propose a mutation rate which at time step t flips each bit independently with probability $2^{(t-1) \mod (\lceil \log_2 n \rceil - 1)}/n$. Doerr et al. [4] observe that this mutation rate is equivalent to a mutation rate of the form α/n, with α drawn uniformly at random from the set $\{2^{(t-1) \mod (\lceil \log_2 n \rceil - 1)} \mid t \in \{1, \ldots, \lceil \log_2 n \rceil\}\}$.

In their pioneering work, Doerr et al. [4] notice that the choice of $p(n) = 1/n$ is a result of over-tailoring the mutation rates to commonly studied simple unimodal problems. They propose a non-static mutation operator fmut_β, which chooses a mutation rate $\alpha \leq 1/2$ from a power-law distribution at every step of the algorithm. Their analysis shows that for a family of "jump" functions introduced below, the run-time of the (1+1) EA is polynomial when using fmut_β and exponential when using the best static mutation rate.

Recently, Friedrich et al. [12] proposed a new mutation operator. Their operator $\mathsf{cMut}(p)$ chooses at each step with constant probability p to flip 1-bit of the solution chosen uniformly at random. With the remaining probability $1-p$, the operator chooses $k \in \{2, \ldots, n\}$ uniformly at random and flips k bits of the solution chosen uniformly at random. This operator performs well in optimizing pseudo-Boolean functions, as well as combinatorial problems such as the minimum vertex cover and the maximum cut. Experiments suggest that this operator outperforms the mutation operator of Doerr et al. [4] when run on functions that exhibit large deceptive basins of attraction, i.e., local optima whose hamming distance from the global optimum is in $\Theta(n)$.

Inspired by the recent results of Doerr et al. [4] and Friedrich et al. [12] we propose the mutation operator pmut_β that mutates n-bit string solutions as follows. At each step, pmut_β chooses $k \in \{1, \ldots, n\}$ from a power-law distribution. Then k bits of the current solution are chosen uniformly at random and then flipped. During a run of the (1+1) EA using pmut_β, the majority of mutations consist of flipping a small number of bits, but occasionally a large number, of up to n bit flips can be performed. In comparison to the mutations of fmut_β, the mutations of pmut_β have a considerably higher likelihood of performing larger than $(n/2)$-bit jumps. A visualization of these probabilities is shown in Fig. 1. Our results can be summarized as follows.

Run-Time Comparison on Artificial Landscapes. In Section 3.1 we show that the (1+1) EA using pmut_β manages to find the optimum of any function within exponential time. When run on the OneMax function, the (1+1) EA with pmut_β finds the optimum solution in expected polynomial time.

In Section 3.2 we consider the problem of maximizing the n-dimensional jump function, first introduced by Droste et al. [5].

$$\mathsf{Jump}(m,n)(x) = \begin{cases} m + |x|_1 & \text{if } |x|_1 \leq n - m \text{ or } |x|_1 = n; \\ n - |x|_1 & \text{otherwise;} \end{cases}$$

We show that for any value of the parameters m, n with m constant or $n - m$, the expected run time of the (1+1) EA using pmut_β remains polynomial. This is not the case for the (1+1) EA using unif_p, for which Droste et al. [5] showed a run time of $\Theta(n^m + n \log n)$ in expectation. Doerr et al. [4] are able to derive polynomial bounds for the expected run-time of the (1+1) EA using their mutation operator fmut_β, but in their results limit the jump parameter to $m \leq n/2$.

In Section 3.3, we compare the performance of our operator with the performance of the (1+1) EA using unif_p together with a restart strategy. A *restart strategy* for a black-box algorithm is a sequence $(t_1, t_2, \ldots, t_t, \ldots)$ that specifies the algorithm should be run from a uniformly random starting point for t_1 steps, then restarted uniformly at random and run again for t_2 steps and so forth. This definition follows the work of Luby et al. [20], and it does not cover frameworks such as the ones given in de Perthuis de Laillevault et al. [2] and Fischetti and Monaci [9]. Restart strategies are beneficial in a large number of settings such as evolutionary algorithms [15] and SAT solvers [24]. We show that under any restart strategy chosen a priori with a time budget constrain, the (1+1) EA with unif_p, has an exponentially small probability of finding the optimum solution of a jump function within polynomial time.

Optimization of Submodular Functions and Experiments. In Section 4 we examine the performance of the (1+1) EA with pmut_β on submodular functions. Submodular functions arise in the analysis of various optimization problems. Examples include: maximum facility location problems [1], maximum cut and maximum directed cut [13], restricted SAT instances [14]. Submodular functions are also found in AI in connection with probabilistic fault diagnosis problems [17, 18].

Submodular functions exhibit additional properties in some cases, such as *symmetry* and *monotonicity*. These properties can be exploited to derive run time bounds for local randomized search heuristics such as the (1+1) EA. In particular, Friedrich and Neumann [11] give run time bounds for the (1+1) EA and GSEMO on this problem, assuming either monotonicity or symmetry.

We show (Section 4.1) that the (1+1) EA with pmut_β on any non-negative, submodular function gives a 1/3-approximation within polynomial time. This result matches the performance of the local search heuristic of Feige et al. [8] designed to target non-negative, submodular functions in particular. An example of a natural non-negative submodular function that is neither symmetric nor monotone is the utility function of a player in a combinatorial auction (see e.g. [19]).

Algorithm 1: General framework for the (1+1) EA

```
Choose initial solution $x \in \{0,1\}^n$ u.a.r.;
while convergence criterion not met do
    $y \leftarrow$ Mutation($x$) for given mutation operator;
    if $f(y) \geq f(x)$ then
        $x \leftarrow y$;
return x;
```

In Section 4.2 we apply our general upper bound to the maximum directed cut problem. Unlike the results of Friedrich et al. [12] we consider graphs with weighted edges, and our run-time bound does not depend on the maximum outdegree.

Finally, we evaluate the performance of the (1+1) EA on the maximum directed cut problem using pmut_β experimentally, on real-world graphs of different origins, and with up to ~37 000 vertices and ~1.6 million edges. Our experiments show that pmut_β outperforms unif_p and fmut_β on those instances.

2 Algorithms and Setting

2.1 The (1 + 1) Evolutionary Algorithm and Mutation Rates

In this paper we look at the run time of the simple (1+1) Evolutionary Algorithm (EA) under various configurations. This algorithm requires a bit-string of fixed length n as input. An offspring is then generated by the *mutation operator*, an operator that resembles asexual reproduction. The fitness of the solution is then computed and the less desirable result is discarded. This algorithm is *elitist* in the sense that the solution quality never decreases throughout the process. Pseudo-code for the (1+1) EA is given in Algorithm 1.

In the (1+1) EA the offspring generated in each iteration depends on the mutation operator. The standard choice for the $\mathsf{Mutation}(\cdot)$ is to flip each bit of an input string $x = (x_1, \ldots, x_n)$ independently with probability $1/n$. In a slightly more general setting, the mutation operator $\mathsf{unif}_p(\cdot)$ flips each bit of x independently wit probability p/n, where $p \in [0, n/2]$. We refer to the parameter p as *mutation rate*.

Uniform mutations can be further generalized, by sampling the mutation rate $p \in [0, n/2]$ at each step according to a given probability distribution. We assume this distribution to be fixed throughout the optimization process. Among this class of mutation rates, is the *power-law* mutation fmut_β of Doerr et al. [4]. fmut_β chooses the mutation rate according to a power-law distribution on $[0, 1/2]$ with exponent β. More formally, denote with X the r.v. (random variable) that returns the mutation rate at a given step. The power-law operator fmut_β uses a probability distribution $D^\beta_{n/2}$ s.t. $\mathsf{Pr}\left(X = k\right) = H^\beta_{n/2} k^{-\beta}$, where $H^\beta_\ell = \sum_{j=1}^{\ell} \frac{1}{j^\beta}$. The H^β_ℓs are known in the literature as generalized harmonic

Algorithm 2: The mutation operator $\mathsf{pmut}_\beta(x)$

```
$y \leftarrow x$;
choose $k \in [1, \dots, n]$ according to distribution $D_n^\beta$;
flip $k$-bits of $y$ chosen u.a.r. ;
**return** $y$;
```

numbers. Interestingly, generalized harmonic numbers can be approximated with the Riemann Zeta function as $\lim_{\ell \to +\infty} H_\ell^\beta = \zeta(\beta)$, with $\zeta(\cdot)$ the Riemann Zeta function. In particular, harmonic numbers $H_{n/2}^\beta$ are always upper-bounded by a constant, for increasing problem size and for a fixed $\beta > 1$.

2.2 Non-uniform Mutation Rates

In this paper we consider an alternative approach to the non-uniform mutation operators described above. For a given probability distribution $P = [1, \dots, n] \longrightarrow \mathbb{R}$ the proposed mutation operator samples an element $k \in [1, \dots, n]$ according to the distribution P, and flips *exactly* k-many bits in an input string $x = (x_1, \dots x_n)$, chosen uniformly at random among all possibilities. This framework depends on the distribution P, which we always assume fixed throughout the optimization process.

Based on the promising results of Doerr et al. [4], we study a specialization of our non-uniform framework that uses a distribution of the form $P = D_n^\beta$. We refer to this operator as pmut_β, and pseudocode is given in Algorithm 2. This operator uses a power-law distribution on the probability of performing exactly k-bit flips in one iteration. That is, for $x \in \{0,1\}^n$ and all $k \in \{1, \dots, n\}$,

$$\mathsf{Pr}\left(\mathcal{H}\left(x, \mathsf{pmut}_\beta(x)\right) = k\right) = (H_n^\beta)^{-1} k^{-\beta} \tag{1}$$

We remark that with this operator, for any two points $x, y \in \{0,1\}^n$, the probability $\mathsf{Pr}\left(y = \mathsf{pmut}_\beta(x)\right)$ only depends on their hamming distance $\mathcal{H}(x, y)$.

Although both operators, fmut_β and pmut_β, are defined in terms of a power-law distribution their behavior differs. A visualization of this can be seen in Fig. 1). We note that, for any choice of the constant $\beta > 1$ and all $x \in \{0,1\}^n$, $\mathsf{Pr}\left(\mathcal{H}\left(x, \mathsf{fmut}_\beta(x)\right) = 0\right) > 0$, while $\mathsf{Pr}\left(\mathcal{H}\left(x, \mathsf{pmut}_\beta(x)\right) = 0\right) = 0$. We discuss the advantages and disadvantages of these two operators in Sections 3 and C.

3 Artificial Landscapes

3.1 General Bounds and the OneMax Function

In this section we derive a general upper-bound on the run time of the (1+1) EA using the mutation operator pmut_β on any fitness function $f \colon \{0,1\}^n \longrightarrow \mathbb{R}$. It is well-known that the (1+1) EA using uniform mutation on any such fitness function has expected run time at most n^n. This upper-bound is tight, in the

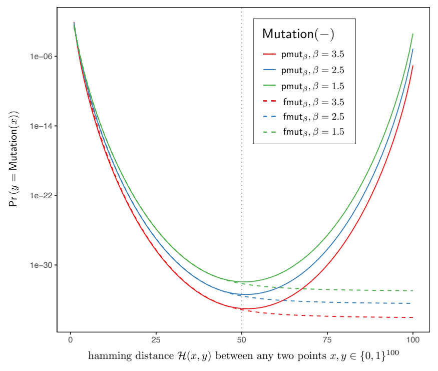

Fig. 1. A visualization of the probability $\Pr(y = \mathsf{Mutation}(x))$, for any two points $x, y \in \{0,1\}^n$ w.r.t. the Hamming distance $\mathcal{H}(x, y)$, for problem size $n = 100$ and for $\beta = 1.5, 2.5, 3.5$. We consider the case $\mathsf{Mutation}(\cdot) = \mathsf{pmut}_\beta$ and $\mathsf{Mutation}(\cdot) = \mathsf{fmut}_\beta$. Note that the y-axis follows a logarithmic scale.

sense that there exists a function f s.t. the expected run time of the (1+1) EA using uniform mutation to find the global optimum of f is $\Omega(n^n)$. For a discussion on these bounds see Droste et al. [5]. Doerr et al. [4] prove that on any fitness function $f\colon \{0,1\}^n \longrightarrow \mathbb{R}$ the (1+1) EA using the mutation operator fmut_β has run time at most $\mathcal{O}\left(H^{\beta}_{n/2} 2^n n^\beta\right)$. Similarly, we derive a general upper-bound on the run time of the (1+1) EA using mutation pmut_β.

Lemma 1. *On any fitness function $f\colon \{0,1\}^n \longrightarrow \mathbb{R}$ the (1+1) EA with mutation pmut_β finds the optimum solution after expected $\mathcal{O}\left(H^{\beta}_{n} e^{n/e} n^\beta\right)$ fitness evaluations, with the constant implicit in the asymptotic notation independent of β.*

Due to space limitations, the proof of this lemma is included in the appendix.

We consider the OneMax function, defined as $\mathsf{OneMax}(x_1, \ldots, x_n) = |x|_1 = \sum_{j=1}^{n} x_j$. This simple linear function of unitation returns the number of ones in a pseudo-Boolean input string. The (1+1) EA with mutation operators unif_p and fmut_β finds the global optimum after $\mathcal{O}(n \log n)$ fitness evaluations (see [22, 5, 4]). It can be easily shown that the (1+1) EA with mutation operator pmut_β achieves similar performance on this instance.

Lemma 2. *The (1+1) EA with mutation pmut_β finds the global optimum of the OneMax after expected $\mathcal{O}\left(H^{\beta}_{n} n \log n\right)$ fitness evaluations, for all $\beta > 1$ and with the constant implicit in the asymptotic notation independent of β.*

Lemma 2 can be proved using the fitness level method outlined in Wegener [26]. The (1+1) EA with mutation pmut_β performs a single chosen bit-flip with probability at least $(H^{\beta}_{n} n)^{-1}$ and the expected time for such an event to occur is $H^{\beta}_{n} n$.

3.2 A Comparison with Static Uniform Mutations

Recall the definition of the *jump* function from the introduction. For $1 < m < n$ this function exhibits a single *local* maximum and a single *global* maximum. The first parameter of $\mathsf{Jump}(m, n)$ determines the hamming distance between the local and the global optimum, while the second parameter denotes the size of the input. We present a general upper-bound on the run time of the (1+1) EA on $\mathsf{Jump}(m, n)$ with mutation operator pmut_β. Then, following the footsteps of Doerr et al. [4], we compare the performance of pmut_β with static mutation operators on jump functions for all $m \leq n/2$.

Lemma 3. *Consider a jump function $f = \mathsf{Jump}(m, n)$ and denote with $T_{\mathsf{pmut}_\beta}(f)$ the expected run time of the (1+1) EA using the mutation pmut_β on the function f. $T_{\mathsf{pmut}_\beta}(f) = H_n^\beta \binom{n}{m} \mathcal{O}\left(m^\beta\right)$, were the constant implicit in the asymptotic notation is independent of m and β.*

Due to space limitations, the proof of this lemma is included in the appendix. Note that the upper-bound on the run time given in Lemma 3 yields polynomial run time on all functions $\mathsf{Jump}(m, n)$ with m constant for increasing problem size and also with $n - m$ constant for increasing problem size.

Following the analysis of Doerr et al. [4], we can compare the run time of the (1+1) EA with mutation pmut_β with the (1+1) EA with uniform mutations, on the jump function $\mathsf{Jump}(m, n)$ for $m \leq n/2$.

Corollary 4. *Consider a jump function $f = \mathsf{Jump}(m, n)$ with $m \leq n/2$ and denote with $T_{\mathsf{pmut}_\beta}(f)$ the run time of the (1+1) EA using the mutation pmut_β on the function f. Similarly, denote with $T_{\text{OPT}}(f)$ the run time of the (1+1) EA using the best possible static uniform mutation on the function f. Then it holds $T_{\mathsf{pmut}_\beta}(f) \leq cm^{\beta-0.5}\, H_n^\beta\, T_{\text{OPT}}(f)$, for a constant c independent of m and β.*

The result above holds because Doerr et al. [4] prove that the best possible optimization time for a static mutation rate a function $f = \mathsf{Jump}(m, n)$ with $m \leq n/2$ is lower-bounded as $1/2\, n^m/m^m\, (n/(n - m))^{n-m} \leq T_{\text{OPT}}(f)$.

3.3 Restart Strategies are not Sufficient

In this section we show that the performance of the (1+1) EA using any static uniform mutation and any restart strategy can be worse than the performance of the (1+1) EA using heavy-tail mutations pmut_β with no restarts. Consider the problem of maximizing the function $\mathsf{Jump}(n-k, n)$, for a constant $k > 0$ and input size n. In this setting, each bit string x with $|x| = k$ is a local optimum with $f(x) = n - 1$ and 1^n is the global optimum, with $f(1^n) = n$. Since $\binom{n}{n-k} \leq n^k$, by applying Lemma 3, we can show that the (1+1) EA with mutation pmut_β, for all $\beta > 1$, finds the global optimum in expected time $\mathcal{O}\left(H_n^\beta k^\beta n^k\right)$.

Now consider the (1+1) EA on $\mathsf{Jump}(n-k, n)$ using a static uniform mutation and an *a priori* restart strategy of the form $(t_1, \ldots, t_\ell)$. We show that, when this algorithm runs on a fixed time budget t_b, the probability of reaching the global optimum is at most $t_b 2^{-\Theta(n)}$. Thus the probability that this algorithm runs in polynomial time is exponentially small.

Theorem 5. *Let $k \in \{0, \dots, n\}$ be a constant and let $f = \mathsf{Jump}(n-k, n)$. Consider the (1+1) EA on f that uses the static uniform mutation operator unif_p, where $0 < p \leq n/2$, and the restart strategy $\mathcal{R} = (t_1, \dots, t_\ell)$. Let $t_b = \sum_{j=1}^{\ell} t_j$ be the total time budget. If A_{OPT} is the event that the above (1+1) EA reaches the global optimum within $t \leq t_b$ evaluations, then $\mathsf{Pr}\left(A_{\text{OPT}}\right) = t_b\, 2^{-\Omega(n)}$.*

Our proof, which appears in the appendix, does not depend at all on the value of f at the global optimum. Thus we can make this value arbitrarily large, showing that the approximation ratio of the (1+1) EA with mutation operator unif_p and restart strategy $\mathcal{R}$ running within polynomial time can be arbitrarily small, with overwhelmingly large probability.

4 Maximizing Submodular Functions

4.1 A General Upper-bound

Consider a finite set V and a function $f \colon 2^V \longrightarrow \mathbb{R}$. We say that f is *submodular* if for all $U, W \subseteq V$, $f(U) + f(W) \geq f(U \cup W) + f(U \cap W)$. We consider the problem of maximizing a non-negative submodular function, with the (1+1) EA using the mutation operator pmut_β. This problem is APX-complete. That is, this problem is NP-hard and does not admit a polynomial time approximation scheme (PTAS), unless $\mathsf{P} = \mathsf{NP}$.

We prove that the (1+1) EA with mutation pmut_β is a $(1/3 - \varepsilon/n)$-approximation algorithm for the problem of maximizing a submodular function. In our analysis we assume neither monotonicity nor symmetry. We approach this problem by searching for $(1+\alpha)$-local optima, which we define below.

Definition 6. *Let $f \colon 2^V \longrightarrow \mathbb{R}_{\geq 0}$ be any submodular function. A set $S \subseteq V$ is a $(1+\alpha)$-local optimum if it holds $(1+\alpha)f(S) \geq f(S \setminus \{u\})$ for all $u \in S$, and $(1+\alpha)f(S) \geq f(S \cup \{v\})$ for all $v \in V \setminus S$, for a constant $\alpha > 0$.*

The definition given above is useful in the analysis because it can be proved that either $(1+\alpha)$-local optima or their complement always yield a good approximation of the global maximum.

Theorem 7. *Consider a non-negative submodular function $f \colon 2^V \longrightarrow \mathbb{R}_{\geq 0}$ over a set of cardinality $|V| = n$ and let S be a $(1+\varepsilon/n^2)$-local optimum. Then either S or $V \setminus S$ is a $(1/3 - \varepsilon/n)$-approximation of the global maximum.*

We remark that Theorem 7 as we present it is implicit in the proof of Theorem 3.4 in Feige et al. [8]. Also, it is possible to construct examples of submodular functions that exhibit $(1+\varepsilon/n^2)$-local optima with arbitrarily bad approximation ratios. Thus, $(1+\varepsilon/n^2)$-local optima alone do not yield any approximation guarantee with respect to the global maximum, unless the valuation oracle is symmetric.

We can use Theorem 7 to estimate the run time of the (1+1) EA using mutation pmut_β to maximize a given submodular function. Intuitively, it is always possible to find a $(1+\varepsilon/n^2)$-local optimum in polynomial time using single bit-flips. It is then possible to compare the approximate local solution S with its complement $V \setminus S$ by flipping all bits in one iteration.

Theorem 8. *Let $f\colon 2^V \longrightarrow \mathbb{R}_{\geq 0}$ be a non-negative submodular function over a set of cardinality $|V| = n$. Then the (1+1) EA with mutation* pmut_β *finds a $(1/3 - \varepsilon/n)$-approximation of the global maximum after expected $\mathcal{O}\left(\frac{1}{\varepsilon} n^3 \log\left(\frac{n}{\varepsilon}\right) + n^\beta\right)$ fitness evaluations.*

Due to space limitations the proof is included in the appendix.

4.2 An Application to the Maximum Directed Cut Problem

Let $G = (V, E)$ be a graph, together with a weight function $w\colon E \longmapsto \mathbb{R}_{\geq 0}$ on the edges. We assume the weights to be non-negative. We consider the problem of finding a subset $U \subseteq V$ of nodes such that the sum of the weights on the outer edges of U is maximal. This problem is the maximum directed cut problem (Max-Di-Cut) and is a known to be NP-complete. In contrast to Friedrich and Neumann [11], our analysis considers both directed and undirected graphs, although it might be possible to obtain improved bounds on undirected graphs. Furthermore, unlike Friedrich et al. [12] our run-time bound does not depend on the size of the maximum cut in G.

We first define the cut function.

Definition 9. *Let $G = (V, E)$ be a graph together with a non-negative weight function $w\colon E \longrightarrow \mathbb{R}_{\geq 0}$. For each subset of nodes $U \subseteq V$, consider the set $\Delta(U) = \{(e_1, e_2) \in E \colon e_1 \in U \text{ and } e_2 \notin U\}$ of all edges leaving U. We define the cut function $f\colon 2^V \longrightarrow \mathbb{R}_{\geq 0}$ as $f(U) = \sum_{e \in \Delta(U)} w(e)$.*

Since we require the weights to be non-negative, the cut function is also non-negative. For any graph $G = (V, E)$ the corresponding cut function is always submodular and, in general, non-monotone (see e.g. [8, 11]). If a graph G is directed, then the corresponding cut function needs not be symmetric. Using Theorem 8, we derive the following upper-bound on the run time.

Corollary 10. *Let $G = (V, E)$ be a graph of order n together with a non-negative weight function $w\colon E \longmapsto \mathbb{R}_{\geq 0}$. Then the (1+1) EA with mutation* pmut_β *is a $(1/3 - \varepsilon/n)$-approximation algorithm for the* Max-Di-Cut *on G. Its expected optimization time is $\mathcal{O}\left(\frac{1}{\varepsilon} n^3 \log\left(\frac{n}{\varepsilon}\right) + n^\beta\right)$.*

4.3 Experiments on Large Real Graphs

For our experimental investigations, we select the 123 large instances used by Wagner et al. [25]. The number of vertices ranges from about 400 to over 6 million and the number of edges ranges from about 1000 to over 56 million. All 123 instances are available online [23].

The instances vary widely in their origin. For example, we include 14 collaboration networks (ca-*, from various sources such as Citeseer and also Hollywood

Table 1. Average ranks (based on mean cut size) at $t = 10\,000$ and $t = 100\,000$ iterations (lower is better). Our pmut_β approaches perform best at both budgets. unif_1 or $\mathsf{fmut}_{1.5}$ have the worst average rank.

mutation	t=10k	t=100k
$\mathsf{fmut}_{1.5}$	3.4	6.8
$\mathsf{fmut}_{2.5}$	4.9	5.1
$\mathsf{fmut}_{3.5}$	5.8	4.6
$\mathsf{pmut}_{1.5}$	1.6	3.1
$\mathsf{pmut}_{2.5}$	2.2	1.9
$\mathsf{pmut}_{3.5}$	3.3	1.1
unif_1	6.8	4.9

productions), 14 web graphs (web-*, showing the state of various subsets of the internet at particular points in time), five infrastructure networks (inf-*), six interaction networks (ia-*, e.g. about email exchange), 21 general social networks (soc-*, e.g., Delicious, LastFM, Youtube) and 44 subnets of Facebook (socfb-*, mostly from different American universities). We take these graphs and run Algorithm 1 with different mutation operators: fmut_β and pmut_β with $\beta \in \{1.5, 2.5, 3.5\}$ and unif_1. The solution representation is based on vertices and we initialize uniformly at random. Each edge has a weight of 1.

We perform 100 independent runs (100 000 evaluations each) with an overall computation budget of 72 hours per mutation-instance pair. Out of the initial 123 instances 67 finish their 100 repetitions per instance within this time limit.[3] We will report on these 67 in the following, and we will use the average cut size achieved in the 100 runs as the basis for our analyses.

First, we rank the seven approaches based on the average cut size achieved in 100 independent runs (best rank is 1, worst rank is 7). Table 1 shows the average rank achieved by the seven different mutation approaches across the 68 instances. It is obvious that unif_1 is among the worst. pmut_β clearly performs best, however, while pmut_β with $\beta = 1.5$ performs best at 10 000 iterations, pmut_β with $\beta = 3.5$ performs best when the budget is 100 000 iterations.

Across the 67 instances, the achieved cut sizes vary significantly (see Fig. 2 and Table 3). For example, the average gap between the worst and the best approach is 46% at 10 000 iterations and it still is 8.1% at 100 000 iterations. Also, when we compare the best fmut_β and pmut_β configurations (as per Table 3), then we can see that (i) pmut_β is better or equal to fmut_β, and (ii) the performance advantage of pmut_β over fmut_β is 2.2% and 1.3% on average, with a maximum of 4.8% and 6.4% (i.e., for 10 000 and 100 000 evaluations).

5 Discussion

In the pursuit of optimizers for complex landscapes that arise in industrial problems, we have identified a new mutation operator. This operator allows for good

[3] Source categories of the 67 instances: 2x bio-*, 6x ca-*, 5x ia-*, 2x inf-*, 1x soc-*, 40x socfb-*, 4x tech-*, 7x web-*. The largest graph is socfb-Texas84 with 36 364 vertices and 1 590 651 edges.

mutation	average rank $t = 10,000$	$t = 100,000$
fmut$_{1.5}$	3.4	6.8
fmut$_{2.5}$	4.9	5.1
fmut$_{3.5}$	5.8	4.6
pmut$_{1.5}$	1.6	3.1
pmut$_{2.5}$	2.2	1.9
pmut$_{3.5}$	3.3	1.1
unif$_1$	6.8	4.9

Table 2. Average ranks (based on mean cut size) of at $t = 10\,000$ and $t = 100\,000$ iterations (lower is better).

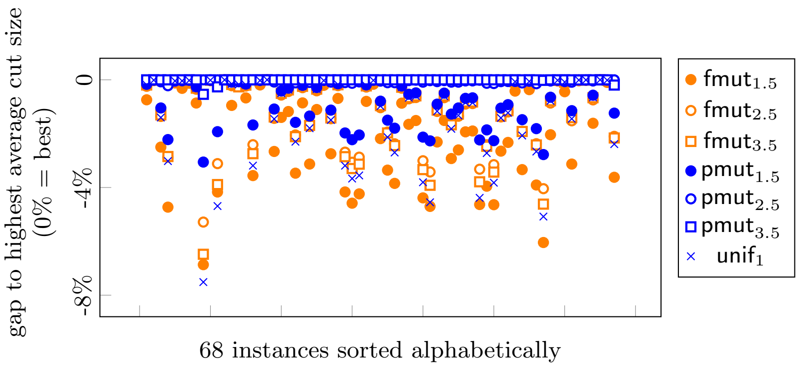

Fig. 2. Distance of average cut size to best average of the seven approaches.

performance of the classical (1+1) EA when optimizing not only simple artificial test functions, but the whole class of non-negative submodular functions. As submodular functions find applications in a variety of natural settings, it is interesting to consider the potential utility of our operator as a building block for optimizers of more complex landscapes, where submodularity can be identified in parts of these landscapes.

References

1. A. A. Ageev and M. Sviridenko. An 0.828-approximation algorithm for the uncapacitated facility location problem. *Discrete Applied Mathematics*, 93(2-3):149–156, 1999.
2. Axel de Perthuis de Laillevault, Benjamin Doerr, and Carola Doerr. Money for nothing: Speeding up evolutionary algorithms through better initialization. In *GECCO*, pages 815–822, 2015.
3. B. Doerr, T. Jansen, D. Sudholt, C. Winzen, and C. Zarges. Mutation rate matters even when optimizing monotonic functions. *Evolutionary Computation*, 21(1):1–27, 2013.
4. B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen. Fast genetic algorithms. In *GECCO*, pages 777–784, 2017.

	$t = 10k$		$t = 100k$	
	total	$\mathsf{pmut}_{1.5}$ vs $\mathsf{fmut}_{1.5}$	total	$\mathsf{pmut}_{3.5}$ vs $\mathsf{fmut}_{3.5}$
min gap	0.3%	0.3%	0.0%	0.0%
mean gap	12.2%	2.2%	2.1%	1.3%
max gap	46.0%	4.8%	8.1%	6.4%

Table 3. Summary of cut-size differences. "total" refers to the gap between the best and worst performing mutation out of all seven. The two highlighted pairs compare the best fmut_β and pmut_β values listed in Table 1.

5. S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.
6. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.
7. A. E. Eiben and J. E. Smith. *Introduction to evolutionary computation.* Natural Computing Series. Springer, 2003.
8. U. Feige, V. S. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal of Computing*, 40(4):1133–1153, 2011.
9. M. Fischetti and M. Monaci. Exploiting erraticism in search. *Operations Research*, 62(1):114–122, 2014.
10. T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.
11. T. Friedrich and F. Neumann. Maximizing submodular functions under matroid constraints by evolutionary algorithms. *Evolutionary Computation*, 23(4):543–558, 2015.
12. T. Friedrich, F. Quinzan, and M. Wagner. Escaping large deceptive basins of attraction with heavy mutation operators. In *GECCO*, 2018. Accepted for publication, available from https://hpi.de/friedrich/docs/paper/GECCO18.pdf.
13. M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6), 1995.
14. J. Håstad. Some optimal inapproximability results. *Jornal of the ACM*, 48(4):798–859, 2001.
15. T. Jansen. On the analysis of dynamic restart strategies for evolutionary algorithms. In *PPSN*, pages 33–43, 2002.
16. T. Jansen and I. Wegener. Real royal road functions–where crossover provably is essential. *Discrete Applied Mathematics*, 149(1-3):111–125, 2005.
17. A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, pages 1650–1654, 2007.
18. J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *STOC*, pages 323–332, 2009.
19. B. Lehmann, D. J. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.
20. M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of las vegas algorithms. *Information Processing Letters*, 47(4):173–180, 1993.
21. M. Mitzenmacher and E. Upfal. *Probability and computing - randomized algorithms and probabilistic analysis.* Cambridge University Press, 2005.

22. H. Mühlenbein. How genetic algorithms really work: Mutation and hillclimbing. In *PPSN*, pages 15–26, 1992.
23. R. A. Rossi and N. K. Ahmed. The Network Data Repository with Interactive Graph Analytics and Visualization (Website). http://networkrepository.com, 2015.
24. V. Ryvchin and O. Strichman. Local restarts. In *SAT*, pages 271–276, Berlin, Heidelberg, 2008. Springer-Verlag.
25. M. Wagner, T. Friedrich, and M. Lindauer. Improving local search in a minimum vertex cover solver for classes of networks. In *CEC*, pages 1704–1711, 2017.
26. I. Wegener. Theoretical aspects of evolutionary algorithms. In *ICALP*, pages 64–78, 2001.
27. C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *STACS*, pages 44–56, 2005.

A Omitted Proofs of Section 3

Proof of Lemma 1. Without loss of generality we assume n to be even. We proceed by identifying a general lower bound on the probability of reaching any point from any other point. To this end, let $x, y \in \{0,1\}^n$ be any two points and let $k = \mathcal{H}(x,y)$ be their Hamming distance. Then the probability of reaching the point y in one iteration from x is

$$\mathsf{Pr}\left(y = \mathsf{pmut}_\beta(x)\right) = \binom{n}{k}^{-1} \mathsf{Pr}\left(\mathcal{H}\left(x, \mathsf{pmut}_\beta(x)\right) = k\right).$$

From (1) we have that it holds

$$\mathsf{Pr}\left(\mathcal{H}\left(x, \mathsf{pmut}_\beta(x)\right) = k\right) = (H_n^\beta)^{-1} k^{-\beta} \geq (H_n^\beta)^{-1} n^{-\beta}$$

for all choices of $x \in \{0,1\}^n$ and $k = 1, \ldots, n$. Using a known lower bound of the binomial coefficient we have that

$$\binom{n}{k}^{-1} \geq \binom{n}{n/2}^{-1} \geq (2e)^{n/2} \geq e^{n/e},$$

from which it follows that

$$\mathsf{Pr}\left(y = \mathsf{pmut}_\beta(x)\right) \geq (H_n^\beta)^{-1} \geq e^{n/e} n^{-\beta},$$

for any choice of x and y. We conclude by taking the inverse of the estimate above, which yields an upper-bound on the probability of convergence on any fitness function. □

Proof of Lemma 3. We use the fitness level method. Define the levels

$$A_i = \{x \in \{0,1\}^n : f(x) = i\}$$

for all $i = 1, \ldots, n$, and consider the quantities

$$s_i = \begin{cases} (n-i)(H_n^\beta)^{-1} & \text{if } 0 \leq i \leq n-m-1; \\ \binom{n}{m}^{-1} (H_n^\beta)^{-1} m^{-\beta} & \text{if } i = n-m; \\ i(H_n^\beta)^{-1} & \text{if } n-m+1 \leq i \leq n-1; \end{cases}$$

Then each s_i is a lower bound for the probability of reaching a higher fitness in one iteration from the level A_i. By the fitness level theorem we obtain an upper bound on the run time as

$$T_{\mathsf{pmut}_\beta}(f) \leq \sum_{i=0}^{n-1} \frac{1}{s_i} \quad \leq \binom{n}{m} H_n^\beta m^\beta + \sum_{i=0}^{n-m-1} \frac{H_n^\beta}{n-i} + \sum_{i=n-m+1}^{n-1} \frac{H_n^\beta}{i},$$
$$\leq \binom{n}{m} H_n^\beta m^\beta + 2H_n^\beta \int_m^n \frac{dx}{x} = \binom{n}{m} H_n^\beta m^\beta + 2H_n^\beta \ln \frac{n}{m},$$

for any choice of $\beta > 1$. Since we have that $1 < m < n$, then it follows that

$$2H_n^\beta \ln \frac{n}{m} \leq 2H_n^\beta \ln n \leq 2H_n^\beta \binom{n}{m},$$

and the lemma follows. □

Proof of Theorem 5 For $i \in \{0, \dots, t\}$, let x_i denote the solution reached by the algorithm after i fitness evaluations. The algorithm finds the global optimum if some $x_i = 1^n$. The solution x_0 is chosen uniformly at random. We will need to bound the probability that starting from x_0 our algorithm reaches the global optimum after at most t evaluations. To do this we condition on the event that the current solution x_i has at least $3n/4$ 1s. Note that the exact choice of $3/4$ is not important, as the constants will eventually be subsumed in the asymptotic notation. Using the definition of conditional probabilities we have,

$$\begin{aligned}\mathsf{Pr}\left(x_i = 1^n\right) &= \mathsf{Pr}\left(\mathsf{unif}_p(x_{i-1}) = 1^n \mid |x_{i-1}| \geq 3n/4\right) \mathsf{Pr}\left(|x_{i-1}|_1 \geq 3n/4\right) + \qquad (2)\\ &\quad + \mathsf{Pr}\left(\mathsf{unif}_p(x_{i-1}) = 1^n \mid |x_{i-1}|_1 < 3n/4\right) \mathsf{Pr}\left(|x_{i-i}|_1 < 3n/4\right)\\ &\leq \mathsf{Pr}\left(|x_{i-1}| \geq 3n/4\right) + \mathsf{Pr}\left(\mathsf{unif}_p(x_{i-1}) = 1^n \mid |x_{i-1}|_1 < 3n/4\right). \qquad (3)\end{aligned}$$

With similar calculations as above for each $i \in \{1, \dots, t\}$ we can bound,

$$\mathsf{Pr}\left(|x_i| \geq 3n/4\right) \leq \mathsf{Pr}\left(|x_{i-1}| \geq 3n/4\right) + \mathsf{Pr}\left(|x_i|_1 \geq 3n/4 \mid |x_{i-1}|_1 < 3n/4\right).$$

Since the algorithm runs on the jump function f with $k \leq 3n/4$, for each $i \in \{1, \dots, t\}$, with $|x_i|_1 \leq 3n/4$, either $\mathsf{unif}_p(x_i) = 1^n$, or $|\mathsf{unif}_p(x_i)|_1 \leq 3n/4$, so we can estimate the second summand of the last shown inequality and obtain,

$$\mathsf{Pr}\left(|x_i| \geq 3n/4\right) \leq \mathsf{Pr}\left(|x_{i-1}| \geq 3n/4\right) + \mathsf{Pr}\left(\mathsf{unif}_p(x_{i-1}) = 1^n \mid |x_{i-1}|_1 < 3n/4\right).$$

By recursively substituting, for each $i \in \{1, \dots, t\}$, the upper-bound of $\mathsf{Pr}\left(|x_i| \geq 3n/4\right)$ given above in (3) we obtain,

$$\mathsf{Pr}\left(x_t = 1^n\right) \leq \mathsf{Pr}\left(|x_0| \geq 3n/4\right) + \sum_{i=1}^{t-1} \mathsf{Pr}\left(\mathsf{unif}_p(x_i) = 1^n \mid |x_i|_1 < 3n/4\right). \qquad (4)$$

For any $x \in \{0,1\}^n$, $\mathsf{Pr}\left(\mathsf{unif}_p(x) = 1^n\right) = (1 - p/n)^{|x|_1}(p/n)^{n-|x|_1}$. Since $p \leq n/2$, this function is non-decreasing with respect to $|x|_1$. Thus, for any $x \in \{0,1\}^n$, $\mathsf{Pr}\left(\mathsf{unif}_p(x) = 1^n \mid |x|_1 \leq 3n/4\right) \leq (1 - p/n)^{3n/4}(p/n)^{n/4} \leq 2^{-\Omega(n)}$. We can use the following upper bound for the left hand-side of (4),

$$\mathsf{Pr}\left(x_t = 1^n\right) \leq \mathsf{Pr}\left(|x_0| \geq 3n/4\right) + (t-1)2^{-\Omega(n)} \qquad (5)$$

As x_0 is chosen uniformly at random, we can apply a Chernoff bound to obtain $\mathsf{Pr}\left(|x_0| \geq 3n/4\right) \leq 2^{-\Omega(n)}$ (See [21, Section 4.2.2]). This yields,

$$\mathsf{Pr}\left(x_t = 1^n\right) \leq t2^{-\Omega(n)}.$$

Since every run of $\mathcal{R}$ is independent, the theorem follows from a union bound.

□

B Omitted Proofs of Section 4

To prove Theorem 8 we do not perform the run time analysis on submodular functions directly, instead, we consider corresponding *potential functions*. Intuitively, for any submodular function f, we define a function g_f that exhibits the same landscape of f, but with additional properties that simplify the analysis. To this end, we consider the following lemma.

Lemma 11. *Consider a non-negative submodular function $f\colon 2^V \longrightarrow \mathbb{R}_{\geq 0}$ and denote with* OPT *its global maximum. For all $U \subseteq V$ let, $g_{f,\varepsilon}(U) = f(U) + \varepsilon\frac{\text{OPT}}{n}$. The following conditions hold*

(1) $g_{f,\varepsilon}(U)$ is submodular.
(2) $g_{f,\varepsilon}(U) \geq \varepsilon\,\text{OPT}/n$, for all subsets $U \subseteq V$.
(3) Suppose that a solution $U \subseteq V$ is a δ-approximation for $g_{f,\varepsilon}$, for a constant $0 < \delta < 1$. Then U is a $(\delta - \varepsilon/n)$-approximation for f.

Proof. *(1)* The submodularity of $g_{f,\varepsilon}(U)$ follows immediately from the fact that $f(U)$ is submodular, together with the fact that the term $\varepsilon\text{OPT}/n$ is constant.
(2) The property follows directly from the definition of $g_{f,\varepsilon}(U)$, together with the assumption that f is non-negative.
(3) Fix a subset $U \subseteq V$ that is an δ-approximation for $g_{f,\varepsilon}$. Then we have that

$$g_{f,\varepsilon}(U) \geq \delta\left(\text{OPT} + \varepsilon\frac{\text{OPT}}{n}\right) \Longrightarrow f(U) \geq \delta\left(\text{OPT} + \varepsilon\frac{\text{OPT}}{n}\right) - \varepsilon\frac{\text{OPT}}{n}.$$

It follows that

$$f(U) \geq \delta\text{OPT} - (1-\delta)\varepsilon\frac{\text{OPT}}{n} \geq \delta\text{OPT} - \varepsilon\frac{\text{OPT}}{n},$$

where the last inequality follows from the assumption that $0 < \delta < 1$. The lemma follows. □

Proof of Theorem 8 We do not perform the analysis on f directly but we consider the potential function from Lemma 11, which is also a submodular function. We will prove that for all $\varepsilon > 0$, the (1+1) EA with mutation pmut_β finds a $(1/3 - \varepsilon/n)$-approximation of $g_{f,\varepsilon}$ within expected $\mathcal{O}\left(\frac{1}{\varepsilon}n^3\log\left(\frac{n}{\varepsilon}\right) + n^\beta\right)$ fitness evaluations. We then apply Lemma 11(3) to conclude that the (1+1) EA with mutation pmut_β finds a $(1/3 - 2\varepsilon/n)$-approximation of f within $\mathcal{O}\left(\frac{1}{\varepsilon}n^3\log\left(\frac{n}{\varepsilon}\right) + n^\beta\right)$ fitness evaluations and the theorem follows.

We denote with OPT the global optimum of f. We divide the run time in two phases. During phase 1, the (1+1) EA finds a $(1 + \varepsilon/n^2)$-local optimum of g_f. During phase 2 the algorithm finds a $(1/3 - \varepsilon/n)$-approximation of the global optimum of g_f using the heavy-tailed mutation.
(Phase 1.) We estimate the run time in this phase with the multiplicative increase method. Denote with x_t the solution found by the (1+1) EA at time step t, for all $t \geq 0$. Then for any solution x_t it is always possible to make an improvement of $(1 + \varepsilon/n^2)g_{f,\varepsilon}(x_t)$ on the fitness in the next iteration, by adding or removing a single vertex, unless x_t is already a $(1 + \varepsilon/n^2)$-local optimum. We refer to any single bit-flip that yields such an improvement of a fitness as *favorable bit-flip.* We give an upper-bound on the number of favorable bit-flips k to reach a $(1 + \varepsilon/n^2)$-local optimum, by solving the following equation

$$\left(1 + \frac{\varepsilon}{n^2}\right)^k \varepsilon\frac{\text{OPT}}{n} \leq \text{OPT} + \varepsilon\frac{\text{OPT}}{n} \Longleftrightarrow \left(1 + \frac{\varepsilon}{n^2}\right)^k \leq \frac{n}{\varepsilon} + 1,$$

where we have used that that the initial solution x_0 is s.t. $g_{f,\varepsilon}(x_0) \geq \varepsilon \text{OPT}/n$ (cf. Lemma 11(2)). From solving the inequality it follows that the (1+1) EA with mutation pmut_β reaches a $(1+\varepsilon/n^2)$-local maximum after at most $k = \mathcal{O}\left(\frac{1}{\varepsilon} n^2 \log\left(\frac{n}{\varepsilon}\right)\right)$ favorable bit-flips. Since the probability of performing a single chosen bit-flip is at least $(H_n^\beta)^{-1} n^{-1} = \Omega(1/n)$, then the expected waiting time for a favorable bit-flip to occur is $\mathcal{O}(n)$, we can upper-bound the expected run time in this initial phase as $\mathcal{O}\left(\frac{1}{\varepsilon} n^3 \log\left(\frac{n}{\varepsilon}\right)\right)$.
(Phase 2.) Assume that a $(1+\varepsilon/n^2)$-local optimum has been found. Then from Theorem 7 follows that either this local optimum or its complement is a $(1/3 - \varepsilon/n)$-approximation of the global maximum. Thus, if the solution found in Phase 1 does not yield the desired approximation ratio, a n-bit flip is sufficient to find a $(1/3 - \varepsilon/n)$-approximation of the global optimum of g_f. The probability of this event to occur is at least $(H_n^\beta)^{-1} n^{-\beta} = \Omega(n^{-\beta})$ by (1). After an additional phase of expected $\mathcal{O}\left(n^\beta\right)$ fitness evaluations the (1+1) EA with mutation pmut_β reaches the desired approximation of the global maximum. □

C Application: Minimum Vertex Cover

In this section, we study the *minimum vertex cover* problem (MVC): Given a graph $G = (V, E)$ with n vertices, find a minimal subset $U \subseteq V$ such that each edge in E is incident to at least one vertex in U. Following Friedrich et al. [10], we approach MVC by minimizing the functions $(u(x), |x|_1)$ in lexicographical order, where $u(x)$ is the number of uncovered edges.

Lemma 12. *On any graph $G = (V, E)$, the (1+1) EA with mutation pmut_β finds a minimum vertex cover after expected $\mathcal{O}\left(H_n^\beta n \log n\right)$ fitness evaluations.*

This lemma follows from Friedrich et al. [10, Theorem 1 and Theorem 2] and (1) for $k = 1$.

The $(1+1)$ *EA* using unif_p as a mutation operator, when solving MVC on complete bipartite graphs, does not find the global optimum within polynomial time. Consider the complete bipartite graph $G = (V, E)$ with partitions V_1, V_2 of size m and $n - m$ respectively, where $0 < m < n/2$. The expected run time of the $(1+1)$ *EA* using unif_p on this instance is at least $\Omega\left(mn^{m-1} + n \log n\right)$. For $m \leq n/3$ the $(1+1)$ *EA* using mutation fmut_β finds the global optimum of MVC after at most $\mathcal{O}\left(H_{n/2}^\beta n^\beta 2^m\right)$ fitness evaluations in expectation and for $m \geq n/3$ after at most $\mathcal{O}\left(H_{n/2}^\beta n^\beta 2^n\right)$ fitness evaluations in expectation. For a discussion on these run time bounds see Friedrich et al. [10] and Doerr et al. [4].

Theorem 13. *On any complete bipartite graph $G = (V, E)$, the (1+1) EA using mutation pmut_β finds a solution to the MVC after expected $\mathcal{O}\left(H_n^\beta (n \log n + n^\beta)\right)$ fitness evaluations.*

Proof. Denote with V_1, V_2 the partition on the nodes of G and suppose that $|V_1| \leq |V_2|$. From Lemma 12 we have that the (1+1) EA with mutation pmut_β

finds a minimum vertex cover within expected $\mathcal{O}\left(H_n^\beta n \log n\right)$ fitness evaluations. This local solution consists of either V_1 or V_2. If V_1 is reached within this phase then the theorem follows. Otherwise, a step that flips all n-bits in one iteration is sufficient to reach the global optimum. From (1), the probability of this event occurring is at least $(H_n^\beta)^{-1} n^{-\beta}$. After an additional phase of length $H_n^\beta n^\beta$ the (1+1) EA reaches the global optimum. $\square$