Natural Computing (2021) 20:561–575
https://doi.org/10.1007/s11047-021-09841-7

Evolutionary algorithms and submodular functions: benefits of heavy-tailed mutations

Francesco Quinzan[1] · **Andreas Göbel**[1] · **Markus Wagner**[2] · **Tobias Friedrich**[1]

Accepted: 8 January 2021 / Published online: 16 February 2021

Abstract
A core operator of evolutionary algorithms (EAs) is the mutation. Recently, much attention has been devoted to the study of mutation operators with dynamic and non-uniform mutation rates. Following up on this area 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 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. We also consider the problem of maximizing a symmetric submodular function under a single matroid constraint and show that the $(1 + 1)$ EA using our operator finds a $(1/3)$-approximation within polynomial time. This performance matches that of combinatorial local search algorithms specifically designed to solve these problems and outperforms them with constant probability. Finally, we evaluate the performance of the $(1 + 1)$ EA using our operator experimentally by considering two applications: (a) the maximum directed cut problem on real-world graphs of different origins, with up to 6.6 million vertices and 56 million edges and (b) the symmetric mutual information problem using a four month period air pollution data set. In comparison with uniform mutation and a recently proposed dynamic scheme, our operator comes out on top on these instances.

Keywords Evolutionary algorithms · Mutation operators · Submodular functions · Matroids

1 Introduction

A key procedure of the $(1+1)$ EA that affects its performance is the *mutation operator*, i.e., the operator that determines at each step how a 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 procedure influences the performance of the $(1 + 1)$ EA and which is the optimal way of choosing the mutation rate (Eiben et al. 1999; Eiben and Smith 2003).

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 one bit from their predecessors. Witt (2005) shows that this choice of $p(n)$ is optimal for all pseudo-Boolean linear functions. Doerr et al. (2013) further observe that changing $p(n)$ by a constant factor can lead to large variations in 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 and Wegener (2006) 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. (2017) observe that this mutation rate is equivalent to a mutation rate of the

Dr. Markus Wagner has been supported by ARC Discovery Early Career Researcher Award DE160100850.

✉ Francesco Quinzan
francesco.quinzan@hpi.de

Andreas Göbel
andreas.goebel@hpi.de

Markus Wagner
markus.wagner@adelaide.edu.au

Tobias Friedrich
friedrich@hpi.de

[1] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

[2] University of Adelaide, Adelaide, Australia

form α/n, where α is chosen uniformly at random (u.a.r.) from the set $\{2^{(t-1) \mod (\lceil \log_2 n \rceil - 1)} \mid t \in \{1, \ldots, \lceil \log_2 n \rceil\}\}$. Doerr and Wagner (2018a, b) have proposed a simple on-the-fly mechanism that can approximate optimal mutation rates well for two unimodal functions.

Doerr et al. (2017) 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 yields a polynomial speed-up when using fmut_β.

Friedrich, Quinzan, and Wagner (2018) propose a new mutation operator, the $\mathsf{cMut}(p)$. This operator 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. (2017) when run on functions that exhibit large deceptive basins of attraction, i.e., local optima whose Hamming distance from the global optimum is in $\Theta(n)$.

As EAs are used extensively in real world applications (Dasgupta and Michalewicz 2013), it is important to extend the theoretical analysis of their performance to the more general classes of functions. To improve the performance of the $(1+1)$ EA in more complex landscapes, inspired by the recent results of Doerr et al. (2017) and Friedrich, Quinzan, and Wagner (2018) we propose a new mutation operator pmut_β. Our operator mutates n-bit string solutions as follows. At each step, pmut_β chooses $k \in \{1, \ldots, n\}$ from a power-law distribution. 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.

Run-time comparison on artificial landscapes

In Sect. 3.1 we show that the $(1+1)$ EA using pmut_β finds 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 Sect. 3.2 we consider the problem of maximizing the jump function $\mathsf{Jump}_{m,n}(x)$, first introduced by Droste et al. (2002). 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. (2002) showed a run-time of $\Theta(n^m + n \log n)$ in expectation. Doerr et al. (2017) 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 they limit the jump parameter to $m \leq n/2$.

Optimization of submodular functions

Our main focus in this article is to study the performance of the $(1+1)$ EA when optimizing submodular functions. Submodularity is a property that captures the notion of diminishing returns. Thus submodular functions find applicability in a large variety of problems. Examples include: maximum facility location problems (Ageev and Sviridenko 1999), maximum cut and maximum directed cut (Goemans and Williamson 1995), and restricted SAT instances (Håstad 2001). Submodular functions under a single matroid constraint arise in artificial intelligence and are connected to probabilistic fault diagnosis problems (Krause and Guestrin 2007; Lee et al. 2009).

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 (2015) give run-time bounds for the $(1+1)$ EA and GSEMO on this problem, assuming either monotonicity or symmetry. Qian et al. (2018, 2017) study the problem of maximizing (generalizations of) submodular functions with multi-objective EAs.

We show (Sect. 5.1) 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. (2011) 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 (Lehmann et al. 2006). We further show (Sect. 5.2) that the $(1+1)$ EA outperforms the local search of Feige et al. (2011) at least with constant probability (w.c.p.).

In Sect. 6 we consider the problem of maximizing a symmetric submodular function under a single matroid constraint. Our analysis shows that the $(1+1)$ EA using pmut_β finds a 1/3-approximation within polynomial time. Our analysis can be easily extended to show that the same results apply to the $(1+1)$ EA when using the uniform mutation operator or unif_p.

Additionally we evaluate the performance of the $(1+1)$ EA using pmut_β experimentally on the maximum directed

Table 1 Upper bounds on the run-time for the (1+1) EA with mutation pmut_β with parameter $\beta > 1$. The expected run-time in the unconstrained case is given in Sect. 5.1, whereas the improved upper-bound in Sect. 5.2. The expected run-time bounds for the $(1+1)$ EA in the constrained case are discussed in Sect. 6. Previous run-time bounds for deterministic local search algorithms are discussed in Feige et al. (2011) and Lee et al. (2009). We remark that *local operations* for the deterministic local search correspond to *favorable moves* in the analysis of the $(1+1)$ EA, and they are the same unit of measurement

submodular maximization	approximation guarantee	deterministic local search		single-objective evolutionary algorithm	
Unconstrained	$\frac{1}{3} - \frac{\varepsilon}{n}$	$\mathcal{O}\left(\frac{1}{\varepsilon}n^3 \log n\right)$	Fitness evaluations	$\mathcal{O}\left(\frac{1}{\varepsilon}n^3 \log\left(\frac{n}{\varepsilon}\right) + n^\beta\right)$	Fitness evaluations in expectation
				$\mathcal{O}\left(\frac{1}{\varepsilon}n^3 + n^\beta\right)$	Fitness evaluations at least w.c.p.
Symmetric, under a single matroid constraint		$\mathcal{O}\left(\frac{1}{\varepsilon}n^4 \log n\right)$	local operations	$\begin{cases}\mathcal{O}\left(\frac{1}{\varepsilon}n^2 \log \frac{n}{\varepsilon}\right) \\ \mathcal{O}\left(\frac{1}{\varepsilon}n^4 \log \frac{n}{\varepsilon}\right)\end{cases}$	Favorable moves in expectation
					Fitness evaluations in expectation

cut problem, on real-world graphs of different origins, and with up to 6.6 million vertices and 56 million edges. Our experiments show that pmut_β outperforms unif_p and the uniform mutation operator on these instances. This analysis appears in Sect. 7.1

To establish our results empirically, in Sect. 7.2 we consider the symmetric mutual information problem under a cardinality constraint. We consider an air pollution data set during a four month interval and use the (1+1) EA to identify the highly informative random variables of this data set. We observe that pmut_β performs better than the uniform mutation operator and unif_p for a small time budget and a small cardinality constraint, but for a large cardinality constraint all mutation operators have similar performance. This might suggest that large jumps allow for speed-up, although single bit-flips are sufficient to find a locally optimal solution.

A comparison of the previously known performance of deterministic local search algorithms on submodular functions and our results on the $(1+1)$ EA can be found in Table 1.

A preliminary version of this article was published at PPSN 18 (Friedrich, Göbel, Quinzan, & Wagner, 2018). In this article, we omit the results on the performance of the $(1+1)$ EA using pmut_β to find a minimum vertex cover (MVC) on complete bipartite graphs, and we extend the study on the general class of submodular maximization problems. Specifically, we extend the run-time analysis of $(1+1)$ EA using pmut_β, when optimizing non-negative submodular functions (Sect. 5.2). Furthermore, we analyze the performance of the $(1+1)$ EA using pmut_β when maximizing symmetric submodular functions under matroid constraints (Sect. 6). Finally we extend the experimental study on the maximum directed cut problem (Sect. 7.1) and perform a new set of experiments on the Maximum Symmetric Mutual Information problem (Sect. 7.2).

2 Preliminaries

2.1 The (1 + 1) EA and mutation rates

We study the run-time of the simple $(1+1)$ Evolutionary Algorithm 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.

Algorithm 1: The $(1+1)$ EA

input: a fitness function $f : 2^V \to \mathbb{R}_{\geq 0}$, over a ground set V;
output: an (approximate) global maximum of the function f;

// sample initial solution
choose $x \in \{0,1\}^n$ uniformly at random;

while *convergence criterion not met* **do**
  // apply mutation operator
  $y \leftarrow$ **Mutation**(x);
  // perform selection
  if $f(y) \geq f(x)$ **then**
    $x \leftarrow y$;
return x;

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 with 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. (2017). 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. $\Pr(X = k) = H^\beta_{n/2} k^{-\beta}$, where $H^\beta_\ell = \sum_{j=1}^{\ell} \frac{1}{j^\beta}$. The H^β_ℓs are known in the literature as generalized harmonic numbers. Interestingly, generalized harmonic numbers can be approximated with the Riemann Zeta function as $\zeta(\beta) = \lim_{\ell \to +\infty} H^\beta_\ell$. In particular, harmonic numbers $H^\beta_{n/2}$ are always upper-bounded by a constant, for increasing problem size and for a fixed $\beta > 1$. Note, however, that the values $\zeta(\beta)$ change significantly depending on β. In fact, for $\beta \to 1$ the Riemann Zeta function tends toward infinity, whereas for $\beta \to +\infty$ it tends toward 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 $P = [1, \ldots, n] \to \mathbb{R}$ the proposed mutation operator samples an element $k \in [1, \ldots, n]$ according to the distribution P, and flips *exactly* k-many bits in an input string $x = (x_1, \ldots x_n)$, chosen uniformly at random among all possibilities. This framework depends on the distribution P, which we always assume fixed throughout the optimization process.

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

input: a pseudo-Boolean array x;
output: a mutated pseudo-Boolean array y;

$y \leftarrow x$;
choose $k \in [1, \ldots, n]$ with distribution D^β_n;
flip k-bits of y chosen uniformly at random;

return y;

Based on the results of Doerr et al. (2017), we study a specialization of our non-uniform framework that uses a distribution of the form $P = D^\beta_n$. 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. For $x \in \{0,1\}^n$ and all $k \in \{1, \ldots, n\}$,

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

We remark that with this operator, for any two points $x, y \in \{0,1\}^n$, the probability

$\Pr\left(y = \mathsf{pmut}_\beta(x)\right)$ 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. We note that, for any choice of the constant $\beta > 1$ and all $x \in \{0,1\}^n$, $\Pr\left(\mathcal{H}\left(x, \mathsf{fmut}_\beta(x)\right) = 0\right) > 0$, while $\Pr\left(\mathcal{H}\left(x, \mathsf{pmut}_\beta(x)\right) = 0\right) = 0$. We discuss the advantages and disadvantages of these two operators in Sect. 3.

2.3 Submodular functions and matroids

Submodular set functions intuitively capture the notion of diminishing returns, i.e., the more you acquire the less your marginal gain. More formally, the following definition holds.

Definition 1 A set function $f : 2^V \to \mathbb{R}_{\geq 0}$ is submodular if it holds $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for all $S, T \subseteq V$.

We remark that in this context V is always a finite set. It is well-known that the defining axiom in Definition 1 is equivalent to the requirement

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T), \tag{2}$$

for all $S, T \subseteq V$ such that $S \subseteq T$ and $x \in V \setminus S$ (Welsh 2010).

We say that a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ is *symmetric* if it holds $f(S) = f(V \setminus S)$ for all $S \subseteq V$.

In some cases, feasible solutions are characterized as the independent sets of a matroid with base set V, as in the following definition.

Definition 2 Given a set V, a matroid $\mathcal{M} = (V, \mathcal{I})$ with base set V consists of a collection of subsets $\mathcal{I}$ of V with the following properties:

- $\emptyset \in \mathcal{I}$;
- if $T \in \mathcal{I}$, then $S \in \mathcal{I}$ for all subsets $S \subseteq T$;
- if $S, T \in \mathcal{I}$ and $|S| \leq |T|$, then there exists a point $x \in T \setminus S$ s.t. $S \cup \{x\} \in \mathcal{I}$.

From the axioms in Definition 2, it follows that two maximal independent sets always have the same number of elements. This number is called the *rank* of a matroid. It is possible to generalize this notion, as in the following definition.

Definition 3 Consider a matroid $\mathcal{M} = (V, \mathcal{I})$. For any subset $S \subseteq V$, the rank function $r(S)$ returns the size of the largest independent set in S, i.e., $r(S) = \arg\max_{T \subseteq S} \{|T| : T \in \mathcal{I}\}$.

2.4 Markov's inequality

We introduce a basic probabilistic inequality that is useful in the run-time analysis in Sect. 5.2. This simple tool is commonly referred to as Markov's Inequality. We use the following variation of it.

Lemma 1 (Markov's Inequality) *Let X be a random variable, where $X \in [0, 1]$. Then it holds*

$$\Pr(X \leq c) \leq \frac{1 - \mathbb{E}[X]}{1 - c},$$

for all $0 \leq c \leq \mathbb{E}[X]$.

For a discussion of Lemma 1, see Theorem 3.1 in Mitzenmacher and Upfal (2017).

2.5 The multiplicative drift theorem

The Multiplicative Drift theorem is a powerful tool to analyze the expected run-time of randomized algorithms such as the $(1+1)$ EA. Intuitively, for a fitness function $f : 2^V \to \mathbb{R}_{\geq 0}$ we view the run of the $(1+1)$ EA as a Markov chain $\{X_t\}_{t \geq 0}$, where X_t depends on the f-value reached at time-step t. The Multiplicative Drift theorem gives an upper-bound on the expected value of the first hitting time $T = \inf\{t : X_t = 0\}$, provided that the change of the average value of the process $\{X_t\}_{t \geq 0}$ is within a multiplicative factor of the previous solution. The following theorem holds.

Theorem 1 (Theorem 3 in Doerr et al. (2012)) *Let $\{X_t\}_{t \geq 0}$ be a random variable describing a Markov process over a finite state space $\mathcal{S} \subseteq \mathbb{R}$. Let T be the random variable that denotes the earliest point in time $t \in \mathbb{N}_0$ such that $X_t = 0$. Suppose that there exist $\delta > 0$, $c_{\min} > 0$, and $c_{\max} > 0$ such that*

- $\mathbb{E}[X_t - X_{t+1} \mid X_t] \geq \delta X_t$;
- $X_t \in [c_{\min}, c_{\max}] \cup \{0\}$;

for all $t < T$. Then it holds $\mathbb{E}[T] \leq \frac{2}{\delta} \ln\left(1 + \frac{c_{\max}}{c_{\min}}\right)$.

3 Artificial landscapes

3.1 General upper bounds for the (1 + 1) EA

In this section we bound from above the run-time of the $(1+1)$ EA using the mutation operator pmut_β on any fitness function $f : \{0,1\}^n \to \mathbb{R}$. 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 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 $\Omega(n^n)$ (Droste et al. 2002). Doerr et al. (2017) prove that on any fitness function $f : \{0,1\}^n \to \mathbb{R}$ 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 2 *On any fitness function $f : \{0,1\}^n \to \mathbb{R}$ the $(1+1)$ EA with mutation pmut_β finds the optimum solution after expected $\mathcal{O}\left(H^\beta_n e^n n^\beta\right)$ fitness evaluations, with the constant implicit in the asymptotic notation independent of β.*

Proof 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

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

From (1) we have that it holds $\Pr\left(\mathcal{H}\left(x, \mathsf{pmut}_\beta(x)\right) = k\right) = (H^\beta_n)^{-1} k^{-\beta} \geq (H^\beta_n)^{-1} n^{-\beta}$ 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

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

from which it follows that $\Pr\left(y = \mathsf{pmut}_\beta(x)\right) \geq (H^\beta_n)^{-1} e^{-n} n^{-\beta}$, for any choice of x and y. We can roughly estimate run-time as a geometric distribution with probability of success $\Pr\left(y = \mathsf{pmut}_\beta(x)\right)$. Hence, we conclude by taking the inverse of the estimate above, which yields an upper-bound on the probability of convergence on any fitness function. □

We consider the OneMax function, defined as $\text{OneMax}(x_1, \ldots, x_n) = \sum_{j=1}^{n} x_j$, for all input strings $(x_1, \ldots, x_n) \in \{0,1\}^n$. 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 (Doerr et al. 2017; Droste et al. 2002; Mühlenbein 1992). It can be easily shown that the $(1+1)$ EA with mutation operator pmut_β achieves similar performance on this instance.

Lemma 3 *The $(1+1)$ EA with mutation pmut_β finds the global optimum of the* OneMax *after expected $\mathcal{O}\left(H_n^\beta n \log n\right)$ fitness evaluations, with the constant implicit in the asymptotic notation independent of β.*

Proof We use the fitness level method outlined in Wegener (2001). Define the levels $A_i = \{x \in \{0,1\}^n : f(x) = i\}$, and consider the quantities $s_i = (n-i)(nH_n^\beta)^{-1}$, for all $i = 1, \ldots, n$. Then each s_i is a lower-bound on the probability of reaching a higher fitness in one iteration. Denote with $T_{\text{pmut}_\beta}(f)$ the run-time of the $(1+1)$ EA with mutation pmut_β on the function $f =$OneMax. By the fitness level theorem, we obtain an upper-bound on the run-time as

$$T_{\text{pmut}_\beta}(f) \leq \sum_{i=0}^{n-1} \frac{1}{s_i} \leq H_n^\beta n \int_0^{n-1} \frac{dx}{n-x} \leq H_n^\beta n \log n$$

and the claim follows. □

3.2 A comparison with static uniform mutations

Droste et al. (2002) defined the following jump function.

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

with $|x|_1$ the function that returns the number of 1s in the input string. For $1 < m < n$ this function exhibits a single *local* maximum and a single *global* maximum. The first parameter of $\text{Jump}_{m,n}$ 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 $\text{Jump}_{m,n}$ with mutation operator pmut_β. Then, following Doerr et al. (2017), we compare the performance of pmut_β with static mutation operators on jump functions for all $m \leq n/2$.

Lemma 4 *Consider a jump function $f = \text{Jump}_{m,n}$ with $m \leq n/2$ and denote with $T_{\text{pmut}_\beta}(f)$ the expected run-time of the $(1+1)$ EA using the mutation pmut_β on the function f. $T_{\text{pmut}_\beta}(f) = H_n^\beta \binom{n}{m} \mathcal{O}\left(m^\beta\right)$, where the constant implicit in the asymptotic notation is independent of m and β.*

Proof We use the fitness level method outlined in Wegener (2001).
Define the levels $A_i = \{x \in \{0,1\}^n : f(x) = i\}$ for all $i = 1, \ldots, n$, and consider the quantities

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

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

$$\begin{aligned} T_{\text{pmut}_\beta}(f) &\leq \binom{n}{m} H_n^\beta m^\beta + \sum_{i=0}^{n-m-1} \frac{nH_n^\beta}{n-i} + \sum_{i=n-m+1}^{n-1} \frac{nH_n^\beta}{i} \\ &\leq \binom{n}{m} H_n^\beta m^\beta + 2nH_n^\beta \int_m^n \frac{dx}{x} \\ &= \binom{n}{m} H_n^\beta m^\beta + 2nH_n^\beta \ln \frac{n}{m}, \end{aligned}$$

for any choice of $\beta > 1$. Since we have that $1 < m < n$ and $2 \leq m \leq n/2$, then it follows that

$$2nH_n^\beta \ln \frac{n}{m} \leq 2nH_n^\beta \ln n \leq 8H_n^\beta \binom{n}{2} \leq 8H_n^\beta \binom{n}{m},$$

and the lemma follows. □

Note that the upper-bound on the run-time given in Lemma 4 yields polynomial run-time on all functions $\text{Jump}_{m,n}$ with m constant for increasing problem size and also with $n-m$ constant for increasing problem size.

Following the analysis of Doerr et al. (2017), 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 $\text{Jump}_{m,n}$ for $m \leq n/2$.

Corollary 1 *Consider a jump function $f = \text{Jump}_{m,n}$ with $m \leq n/2$ and denote with $T_{\text{pmut}_\beta}(f)$ 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 $T_{\text{pmut}_\beta}(f) \leq cm^{\beta-0.5} H_n^\beta T_{\text{OPT}}(f)$, for a constant c independent of m and β.*

The result above holds because Theorem 5.5 in Doerr et al. (2017) prove that the best possible optimization time for a static mutation rate a function $f = \text{Jump}_{m,n}$ with $m \leq n/2$ is lower-bounded as $c\sqrt{m}T_{\text{OPT}}(f)$.

4 Large jumps are useful

We give a concrete example of a combinatorial problem, showing that large jumps are useful. We study the minimum vertex cover problem (MVC) to this end. Given a graph $G = (V, E)$ MVC asks to find a set of vertices of minimum size that includes at least one endpoint of every edge of the graph. This problem appears on the famous list of NP-complete problems by Karp (1972). For a fixed indexing on the nodes of G, sets of vertices can be represented as bit-strings x of length $n = |V|$. Here, a one in the i-th position of x denotes that the i-th vertex is included in the corresponding set of vertices, whereas a zero denotes that the i-th vertex not included in that set. Following Friedrich et al. (2010) and Oliveto et al. (2009), we use the $(1 + 1)$ EA for the MVC, by maximizing the fitness function $f(x) = (n + 1)u(x) + |x|_1$, where $u(x)$ denotes the number of uncovered edges by x.

Our example consists of finding the minimum vertex cover of a complete bipartite graph G. In a complete bipartite graph, nodes can be partitioned into two sets V_1 and V_2. Each node in V_1 is connected to every node in V_2 and each node in V_2 is connected to every node in V_1. Furthermore, there is no edge connecting two nodes in V_1, or two nodes in V_2.

The MVC on a complete bipartite graph G offers an example where there exists a deceptive local optimum with a large basin of attraction. This is the case when the partition $\{V_1, V_2\}$ of the nodes is such that $|V_1| = \varepsilon n$ and $|V_2| = (1-\varepsilon)n$, with $n^{1-\delta} \leq \varepsilon \leq 1/2$ and $\delta > 0$ a constant. In this case, the set V_2 is a deceptive local optimum. If the $(1 + 1)$ EA reaches this set, then a jump of n bit-flips is required to reach the global optimum V_1, since any smaller jump will lower the fitness f. For this reason, the $(1 + 1)$ EA with the standard uniform mutation has exponential expected run-time, as discussed in (Friedrich et al. 2010, Theorem 5). However, we prove that the $(1 + 1)$ EA with mutation pmut_β yields expected polynomial run-tie on this instance. Or analysis also yields an improved upper-bound over the $\mathcal{O}\left(n^\beta 2^{\varepsilon n}\right)$ expected run-time of Doerr et al. (2017), which they prove for the $(1 + 1)$ EA with fmut_β, on this MVC instance with $\varepsilon \leq 1/3$. First, we note that the following lemma holds.

Lemma 5 *The $(1 + 1)$ EA with mutation operator pmut_β finds a minimum vertex cover set in time $\mathcal{O}\left(H_n^\beta n \log n\right)$.*

This lemma follows from Theorem 2 in Friedrich et al. (2010). In fact, the proof technique developed in Friedrich et al. (2010) for this theorem uses single bit-flips only, and it trivially extends to our mutation operator, taking into account that the probability of performing a single bit-flip with the pmut_β operator is $H_n^{-\beta}$.

Note that the minimum vertex cover found by the $(1 + 1)$ EA as in Lemma 5 is not necessarily a solution to the MVC. In the case of a complete bipartite graph G as described earlier, Lemma 5 gives the expected run-time until the $(1 + 1)$ EA reaches either the set V_1 or the set V_2, which are the only minimum vertex cover sets of G. The following lemma ensures that the $(1 + 1)$ EA finds the solution for the MVC in polynomial time, on complete bipartite graphs.

Lemma 6 *The $(1+1)$ EA with mutation operator pmut_β finds a minimum vertex cover in time $\mathcal{O}\left(H_n^\beta (n \log n + n^\beta)\right)$.*

Proof From Lemma 5, the $(1+1)$ EA finds a minimum vertex cover after $\mathcal{O}\left(H_n^\beta n \log n\right)$ expected run-time. This solution consists either of the set V_1 or the the set V_2. If the $(1 + 1)$ EA finds the solution V_1, then the claim holds. Suppose that the $(1 + 1)$ EA finds the solution V_2. Then an n bit-flip is sufficient to escape V_2 and jump to the global optimum V_1. The probability of an n-bit flip to occur is $(H_n^\beta)^{-1} n^{-\beta}$, and the expected run-time of this bit-flip to occur is $H_n^\beta n^\beta$. The lemma follows □

5 The unconstrained submodular maximization problem

We study the problem of maximizing a non-negative submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ with no side constraints. More formally, we study the problem

$$\text{argmax}_{C \subseteq V} f(C). \tag{3}$$

This problem is APX-complete. That is, this problem is NP-hard and does not admit a polynomial time approximation scheme (PTAS), unless P = NP (Nemhauser and Wolsey 1978).

We denote with OPT any solution of Problem (3), and we denote with n the size of V.

5.1 A general upper-bound on the run-time

We prove that the $(1 + 1)$ EA with mutation pmut_β is a $(1/3 - \varepsilon/n)$-approximation algorithm for Problem 3. 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 4 Let $f : 2^V \to \mathbb{R}_{\geq 0}$ 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 $\alpha > 0$.

This definition is useful in the analysis because it makes possible to prove that either $(1 + \alpha)$-local optima or their complement always yield a good approximation of the global maximum, as in the following theorem.

Theorem 2 (Theorem 3.4 in Feige et al. (2011)) *Consider a non-negative submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ and let S*

be a $(1+\alpha)$*-local optimum as in Definition 4. Then,* $2(1+n\alpha)f(S)+f(V\setminus S)\geq \text{OPT}$.

From Theorem 2, it follows that with $\alpha \leq \varepsilon/n^2$, then either S or $V\setminus S$ is a $(1/3-\varepsilon/n)$-approximation of the global maximum of f. 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 for Problem (3), unless the fitness function is symmetric.

We can use Theorem 2 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.

We do not perform the analysis on a given submodular function f directly, but we consider a corresponding *potential function* $g_{f,\varepsilon}$ instead. Intuitively, we introduce an additive term in the fitness function f, to ensure that the initial solution is always lower-bounded by a positive constant. This assumption allows us to obtain a good upper-bound on the expected run-time. We define potential functions as in the following lemma.

Lemma 7 *Consider a non-negative submodular function* $f\colon 2^V\to\mathbb{R}_{\geq 0}$. *Consider the function* $g_{f,\varepsilon}(U)=f(U)+\varepsilon\frac{\text{OPT}}{n}$, *for all* $U\subseteq V$. *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

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

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. □

Using potential functions and their properties, we can prove the following result.

Theorem 3 *The* $(1+1)$ *EA with mutation* pmut_β *is a* $(1/3-\varepsilon/n)$*-approximation algorithm for Problem* (3). *Its expected run-time is* $\mathcal{O}\left(\frac{1}{\varepsilon}n^3\log\frac{n}{\varepsilon}+n^\beta\right)$.

Proof We prove that for all $\varepsilon>0$, the $(1+1)$ EA with mutation pmut_β finds a $(1/3-\varepsilon/n)$-approximation of $g_{f,\varepsilon}$ (as in Lemma 7) within expected $\mathcal{O}\left(\frac{1}{\varepsilon}n^3\log\frac{n}{\varepsilon}+n^\beta\right)$ fitness evaluations. We then use this knowledge to conclude that the $(1+1)$ EA with mutation pmut_β finds a $(1/3-2\varepsilon/n)$-approximation of f within $\mathcal{O}\left(n^\beta+\frac{1}{\varepsilon}n^3\log\frac{n}{\varepsilon}\right)$ fitness evaluations and the theorem follows.

We divide the run-time into two phases. During Phase 1, the (1+1) EA finds a solution that is at least a $(1+\varepsilon/n^2)$-local optimum of $g_{f,\varepsilon}$. 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 2 uses the fact that the solution found in Phase 1 is at least a $(1+\varepsilon/n^2)$-local optimum. If a solution found in Phase 1 is a $(1+\alpha)$-local optimum with $\alpha\leq\varepsilon/n^2$, then this only results into an improved approximation guarantee (see Theorem 2).

Phase 1 Let x_t be 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 performing a single bit-flip, 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 a *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 inequality

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

where we have used for the initial solution x_0, $g_{f,\varepsilon}(x_0)\geq\varepsilon\text{OPT}/n$ (see Lemma 7(2)). By taking the logarithm on both sides and solving this inequality on k, it follows that the $(1+1)$ EA with mutation pmut_β reaches a $(1+\varepsilon/n^2)$-local maximum after at most $k=\mathcal{O}\left(\frac{1}{\varepsilon}n^2\log\frac{n}{\varepsilon}\right)$ 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 $\mathcal{O}\left(\frac{1}{\varepsilon}n^3\log\frac{n}{\varepsilon}\right)$.

Phase 2 Assume that a $(1+\varepsilon/n^2)$-local optimum has been found. Then by Theorem 2 it 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, an 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. □

5.2 An improved upper-bound on the run-time

We prove that the $(1+1)$ EA with mutation pmut_{β} yields an improved upper-bound on the run-time over that of Theorem 3, at least with constant probability. This upper-bound yields an improvement over the run-time analysis of a standard deterministic Local Search (LS) algorithm (Feige et al. 2011, Theorem 3.4), at least with constant probability. To this end, we exploit a well-known property of submodular functions, by which randomly chosen sets yield a constant-factor approximation of the optimal solution. More formally, the following theorem holds.

Theorem 4 (Theorem 2.1 in Feige et al. (2011)) *Let $f : 2^V \rightarrow \mathbb{R}_{\geq 0}$ be a submodular function, and denote with $R \subseteq V$ a set chosen uniformly at random. Then $\mathbb{E}[f(R)] \geq \text{OPT}/4$.*

We exploit this result to obtain an improved upper-bound on the run-time. Intuitively, the initial solution sampled by the $(1+1)$ EA yields a constant-factor approximation guarantee at least with constant probability. We can use this result to prove the following theorem.

Theorem 5 *The $(1+1)$ EA with mutation pmut_{β} finds a $(1/3-\varepsilon/n)$-approximation algorithm for Problem (3) after $\mathcal{O}\left(\frac{1}{\varepsilon}n^3 + n^{\beta}\right)$ fitness evaluations at least with constant probability (w.c.p.).*

Proof This proof is similar to that of Theorem 3. We denote with x_t a solution reached by the $(1+1)$ EA at time step t. We first prove that the definition of submodularity implies that with high probability the initial solution x_0 yields a constant-factor approximation guarantee. We then perform a run-time analysis as in Theorem 3, by counting the expected time until the fittest individual is chosen for selection, and a local improvement of at least $(1+\varepsilon/n^2)$ is made, assuming that the initial solution yields a constant-factor approximation guarantee.

Denote with $R \subseteq V$ a set chosen uniformly at random and fix a constant $\delta > 1$. We combine Theorem 4 with Lemma 1, by choosing $X = f(R)/\text{OPT}$ and obtain,

$$\mathsf{Pr}\left(f(R) \leq \frac{1}{4\delta}\text{OPT}\right) = \mathsf{Pr}\left(X \leq \frac{1}{4\delta}\right) \leq \frac{1-1/4}{1-1/4\delta} = \frac{3\delta}{4\delta-1},$$

where the last inequality follows by applying Theorem 4 and the linearity of expectation to the r.v. $X = f(R)/\text{OPT}$, to obtain that $\mathbb{E}[X] \geq 1/4$. We have that

$$\mathsf{Pr}\left(x_0 > \frac{1}{4\delta}\text{OPT}\right) \geq 1 - \mathsf{Pr}\left(f(R) \leq \frac{1}{4\delta}\text{OPT}\right) \geq 1 - \frac{3\delta}{4\delta-1}.$$

In the following, for a fixed constant $\delta > 1$, we perform the run-time analysis as in Theorem 3 conditional on the event $\mathcal{A} = \{x_0 > \text{OPT}/4\delta\}$, which occurs at least w.c.p.

Again, we divide the run-time into two phases. During phase 1, the $(1+1)$ EA finds a $(1+\varepsilon/n^2)$-local optimum of f. During phase 2 the algorithm finds a $(1/3-\varepsilon/n)$-approximation of the global optimum of f using the heavy-tailed mutation.

Phase 1 For any solution x_t it is always possible to make an improvement of $(1+\varepsilon/n^2)f(x_t)$ on the fitness in the next iteration, by adding or removing a single element—the favorable bit-flip, unless x_t is already a $(1+\varepsilon/n^2)$-local optimum. Again, 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

$$\left(1+\frac{\varepsilon}{n^2}\right)^k \frac{\text{OPT}}{4\delta} \leq \text{OPT} \Longleftrightarrow \left(1+\frac{\varepsilon}{n^2}\right)^k \leq 4\delta,$$

from which it follows that the $(1+1)$ EA with mutation pmut_{β} reaches a $(1+\varepsilon/n^2)$-local maximum after at most $k = \mathcal{O}\left(\frac{1}{\varepsilon}n^2\right)$ favorable moves. 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 $\mathcal{O}\left(\frac{1}{\varepsilon}n^3\right)$.

Phase 2 In applying the heavy-tailed mutation we conclude: 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 f. The probability that this event will 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} performs an n-nit flip, thus reaching the desired approximation ratio. □

6 Symmetric submodular functions under a matroid constraint

In this section we consider the problem of maximizing a non-negative submodular function $f : 2^V \rightarrow \mathbb{R}_{\geq 0}$ under a single matroid constraint $\mathcal{M} = (V, \mathcal{I})$. More formally, we study the problem

$$\text{argmax}_{C \in \mathcal{I}} f(C). \tag{4}$$

We denote with OPT any solution of Problem (4), and we denote with n the size of V. Note that this definition of OPT differs from that of Sect. 5.

We approach this problem, by maximizing the following fitness function

$$z_f(C) = \begin{cases} f(C) & \text{if } C \in \mathcal{I}; \\ r(C) - |C| & \text{otherwise;} \end{cases} \tag{5}$$

with r the rank function as in Definition 3. If a solution C is unfeasible, then $z_f(C)$ returns a negative number, whereas if C is feasible, then $z_f(C)$ outputs a non-negative number.

When studying additional constraints on the solution space the problem becomes more involved, so we require a different notion of local optimality.

Definition 5 Let $f\colon 2^V \to \mathbb{R}_{\geq 0}$ be a submodular function, let $\mathcal{M} = (V, \mathcal{I})$ be a matroid and let $\alpha > 0$. A set $S \in \mathcal{I}$ is a $(1+\alpha)$-local optimum if the following hold.

- $(1+\alpha) f(S) \geq f(S \setminus \{u\})$ for all $u \in S$;
- $(1+\alpha) f(S) \geq f(S \cup \{v\})$ for all $v \in V \setminus S$ s.t. $S \cup \{v\} \in \mathcal{I}$;
- $(1+\alpha) f(S) \geq f((S \setminus \{u\}) \cup \{v\})$ for all $u \in S$ and $v \in V \setminus S$ s.t. $(S \setminus \{u\}) \cup \{v\} \in \mathcal{I}$.

We prove that, in the case of a *symmetric* submodular function, a $(1+\alpha)$-local optimum as in Definition 5 yields a constant-factor approximation ratio. To this end, we make use of the following well-known result.

Theorem 6 (Theorem 1 in Lee et al. (2009)) *Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid and $I, J \in \mathcal{I}$ be two independent sets. Then there is a mapping $\pi : J \setminus I \to (I \setminus J) \cup \{\emptyset\}$ such that*

- $(I \setminus \{\pi(b)\}) \cup \{b\} \in \mathcal{I}$ *for all* $b \in J \setminus I$;
- $\left|\pi^{-1}(e)\right| \leq 1$ *for all* $e \in I \setminus J$.

Based on the work of Lee et al. (2009), the following lemma holds. Our lemma differs in that, since we assume symmetry, the analysis significantly simplifies and it requires no divide and conquer. Furthermore, since side constraints consist of a single matroid, we need only to search for $(1+\varepsilon/n^2)$-local optima, instead of $(1+\varepsilon/n^4)$-local optima.

Lemma 8 *Consider a non-negative symmetric submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$, a matroid $\mathcal{M} = (V, \mathcal{I})$ and let S be a $(1+\varepsilon/n^2)$-local optimum as in Definition 5. Then S is a $(1/3 - \epsilon/n)$-approximation for Problem (4).*

Proof Fix a constant $\varepsilon > 0$ and a set $C \in \mathcal{I}$. Consider a mapping $\pi : C \setminus S \to (S \setminus C) \cup \{\emptyset\}$ as in Theorem 6. Since S is a $(1+\varepsilon/n^2)$-local optimum it holds

$$\left(1 + \frac{\varepsilon}{n^2}\right) f(S) \geq f((S \setminus \{\pi(b)\}) \cup b); \tag{6}$$

for all $b \in C \setminus S$. Thus, it holds

$$\begin{aligned} & f(S \cup \{b\}) - f(S) \\ & \leq f((S \setminus \{\pi(b)\}) \cup \{b\}) - f(S \setminus \{\pi(b)\}) \\ & \leq \left(1 + \frac{\varepsilon}{n^2}\right) f(S) - f(S \setminus \{\pi(b)\}), \end{aligned}$$

where the first inequality follows from (2), and the second one follows from (6). Summing these inequalities for each $b \in C \setminus S$ and using submodularity as in (2) we obtain,

$$\begin{aligned} & f(S \cup C) - f(S) \\ & \leq \sum_{b \in C \setminus S} [f(S \cup \{b\}) - f(S)] \\ & \leq \sum_{b \in C \setminus S} \left[\left(1 + \frac{\varepsilon}{n^2}\right) f(S) - f(S \setminus \{\pi(b)\})\right]. \end{aligned}$$

Consider a given order of the elements in $b \in C \setminus S$, i.e., $C \setminus S = \{b_1, \ldots, b_k\}$. Then it holds

$$\begin{aligned} & \sum_{b \in C \setminus S} \left[\left(1 + \frac{\varepsilon}{n^2}\right) f(S) - f(S \setminus \{\pi(b)\})\right] \\ & = \sum_{j=1}^{k} [f(S) - f(S \setminus \{\pi(b_j)\})] + k\frac{\varepsilon}{n^2} f(S) \\ & \leq \sum_{j=2}^{k} f\left((S \cap C) \bigcup_{\ell=1}^{j} \{\pi(b_\ell)\}\right) \\ & \quad - \sum_{j=2}^{k} f\left((S \cap C) \bigcup_{\ell=1}^{j-1} \{\pi(b_\ell)\}\right) \\ & \quad + f((S \cap C) \cup \{\pi(b_1)\}) - f(S \cap C) \\ & \quad + k\frac{\varepsilon}{n^2} f(S) \leq \left(1 + \frac{\varepsilon}{n}\right) f(S) - f(S \cap C) \end{aligned}$$

where the first inequality follows from (2) and the second inequality follows by taking the telescopic sum together with the fact that $k \leq n$. Thus, it follows that

$$2\left(1 + \frac{\varepsilon}{n}\right) f(S) \geq f(S \cup C) + f(S \cap C),$$

Since f is symmetric, $f(S) = f(V \setminus S)$ and we have that,

$$\begin{aligned} 3\left(1 + \frac{\varepsilon}{n}\right) f(S) & \geq f(\overline{S}) + f(S \cup C) + f(S \cap C) \\ & \geq f(C \setminus S) + f(C \cap S) \geq f(C). \end{aligned}$$

The claim follows by choosing $C = \text{OPT}$. □

We use Lemma 8 to perform a run-time analysis of the $(1+1)$ EA. We consider the case of the pmut_β mutation, although our proof easily extends to the standard uniform mutation and fmut_β. We experimentally compare these operators in Sect. 7.2. We perform the analysis by estimating the expected run-time until a $(1+\varepsilon/n^2)$-local optimum is reached and apply

Lemma 8 to obtain the desired approximation guarantee. Our analysis yields an improved upper-bound on the run-time over that of Friedrich and Neumann (2015). The following theorem holds.

Theorem 7 *The* $(1+1)$ *EA with mutation* pmut_β *is a* $(1/3-\varepsilon/n)$*-approximation algorithm for Problem* (4)*. Its expected run-time is* $\mathcal{O}\left(\frac{1}{\epsilon}n^4\log\frac{n}{\varepsilon}\right)$.

Proof We perform the analysis assuming that a fitness function as in (5) is used. We divide the run-time in two phases. During phase 1 the $(1+1)$ EA finds a feasible solution, whereas in phase 2 it finds a $(1+\varepsilon/n^2)$-local optimum, given that an independent set has been found.

Phase 1: Assuming that the initial solution is not an independent set then the $(1+1)$ EA maximizes the function $r(C)-|C|$ until a feasible solution is found. This is equivalent to minimizing the function $|C|-r(C)$. We estimate the run-time using the multiplicative drift theorem (Theorem 1). Denote with x_t a solution found by the $(1+1)$ EA after t steps, consider the Markov chain $X_t = |x_t| - r(x_t)$ and consider the first hitting time $T = \min\{t\colon X_t = 0\}$. Then it holds $X_t \in \{0\}\cup[1,n]$. Moreover, since the probability of removing a single chosen bit-flip from the current solution is $(nH_n^\beta)^{-1}$, we have, $\mathbb{E}\left[X_t - X_{t+1} \mid X_t\right] \geq \frac{X_t}{(nH_n^\beta)}$. Theorem 1 now yields, $\mathbb{E}[T] \leq 2nH_n^\beta\log(1+n)$. We conclude that we can upper-bound the run-time in this initial phase as $\mathcal{O}(n\log n)$.

Phase 2: We estimate the run-time in this phase with the multiplicative increase method. Assuming that a feasible solution is reached, then all subsequent solutions are feasible, since $z_f(C)\geq 0$ for all feasible solutions and $z_f(C)<0$ for all infeasible solutions.

To estimate the run-time in this phase we do not perform the analysis on f directly but we consider the potential function $g_{f,\varepsilon}$ from Lemma 7 (recall that in this case OPT is not the global optimum of f, but the highest f-value among all feasible solutions). We prove that for all $\varepsilon>0$, the $(1+1)$ EA with mutation pmut_β finds a $(1/3-\varepsilon/n)$-approximation of $g_{f,\varepsilon}(S) = f(S)+\frac{\mathrm{OPT}}{\varepsilon}$, within expected $\mathcal{O}\left(\frac{1}{\varepsilon}n^4\log\frac{n}{\varepsilon}\right)$ fitness evaluations. We apply Lemma 7(3) and conclude that the $(1+1)$ EA with mutation pmut_β finds a $(1/3-2\varepsilon/n)$-approximation of f within $\mathcal{O}\left(\frac{1}{\varepsilon}n^4\log\frac{n}{\varepsilon}\right)$ fitness evaluations.

Denote with y_t the solution found by the $(1+1)$ EA at time step $t+\ell$, for all $t\geq 0$, with ℓ the number of steps in Phase 1. In other words, y_0 is the first feasible solution found by the $(1+1)$ EA, and y_t is the solution found after additional t steps. Again, the solutions y_t are independent sets for all $t\geq 0$. For any solution y_t it is always possible to make an improvement of $(1+\varepsilon/n^2)g_{f,\varepsilon}(y_t)$ on the fitness in the next iteration, by adding or removing a single vertex, or by swapping two bits, unless y_t is already a $(1+\varepsilon/n^2)$-local optimum. Again, we refer to any single bit-flip or swap that yields such an improvement of a fitness as favorable move. We give an upper-bound on the number of favorable moves k to reach a $(1+\varepsilon/n^2)$-local optimum, by solving the following equation

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

where we have used for the initial solution y_0, $g_{f,\varepsilon}(y_0)\geq \varepsilon\mathrm{OPT}/n$ (see Lemma 7(2)). From solving the inequality it follows that the $(1+1)$ EA reaches a $(1+\varepsilon/n^2)$-local maximum after at most $k=\mathcal{O}\left(\frac{1}{\varepsilon}n^2\log\frac{n}{\varepsilon}\right)$ favorable moves. Since the probability of performing a single chosen bit-flip or a swap is at least $H_n^{-\beta}2^{-\beta}n^{-2}$, then the expected waiting time for a favorable bit-flip to occur is at most $\mathcal{O}\left(n^2\right)$, hence we can upper-bound the expected run-time in Phase 2 as $\mathcal{O}\left(\frac{1}{\varepsilon}n^4\log\frac{n}{\varepsilon}\right)$. □

We remark that a similar result hold for the $(1+1)$ EA using uniform and fmut_β.

7 Experiments

7.1 The maximum directed cut problem

Given a directed graph $G=(V,E)$, we consider the problem of finding a subset $U\subseteq V$ of nodes such that the sum of 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.

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

$$f(U)=|\Delta(U)|\,. \tag{7}$$

The Max-Di-Cut can be approached by maximizing the cut function as in (7). Note that this function is non-negative. Moreover, it is always submodular and, in general, non-monotone (Feige et al. 2011; Friedrich and Neumann 2015). Hence, this approach to the Max-Di-Cut can be formalized as in Problem (3) in Sect. 5.

We select the 123 large instances used by Wagner et al. (2017); the number of vertices ranges from about 379 to over 6.6 million, and the number of edges ranges from 914 to over 56 million. All 123 instances are available online (Rossi and Ahmed 2015).

The instances come from a wide range of origins. For example, there are 14 collaboration networks (ca-*, from various sources such as Citeseer, DBLP, and also Hollywood productions), five infrastructure networks (inf-*), six interaction networks (ia-*, i.e., about email exchange), 21

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

Mutation	Average rank	
	$t = 10,000$	$t = 100,000$
$\mathsf{fmut}_{1.5}$	4.1	5.9
$\mathsf{fmut}_{2.5}$	5.7	4.6
$\mathsf{fmut}_{3.5}$	6.6	4.0
$\mathsf{pmut}_{1.5}$	2.4	3.0
$\mathsf{pmut}_{2.5}$	3.0	1.8
$\mathsf{pmut}_{3.5}$	4.0	1.1
unif_1	2.1	6.7

general social networks (soc-*, i.e., Flickr, LastFM, Twitter, Youtube), 44 subnets of Facebook (socfb-*, mostly from different American universities), and 14 web graphs (web-*, showing the state of various subsets of the Internet at particular points in time). We take these graphs and run Algorithm 1 with seven mutation operators: fmut_β and pmut_β with $\beta \in \{1.5, 2.5, 3.5\}$ and unif_1.[1] We use an intuitive bit-string representation based on vertices, and we initialize uniformly at random. Each edge has a weight of 1.

For each instance-mutation pair, we perform 100 independent runs (100 000 evaluations each) and with an overall computation budget of 72 hours per pair. Out of the initial 123 instances 67 finish their 100 repetitions per instance within this time limit.[2] We report on these 67 in the following. We use the average cut size achieved in the 100 runs as the basis for our analyses.

Firstly, we rank the seven approaches based on the average cut size achieved (best rank is 1, worst rank is 7). Table 2 shows the average rank achieved by the different mutation approaches. unif_1 performs best at the lower budget and worst at the higher budget, which we take as a strong indication that few bit-flips are initially helpful to quickly improve the cut size, while more flips are helpful later in the search to escape local optima. At the higher budget, both fmut_β and pmut_β perform better than unif_1, independent of the parameter chosen. In particular, pmut_β clearly performs better than fmut_β at both budgets, 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.

To investigate the relative performance difference and the statistical significance thereof, we perform a Nemenyi two-tailed test (see Fig. 1). This test performs all-pairs comparisons on Friedman-type ranked data. The results are as expected and consistent with the average ranks reported in Table 2.

Across the 67 instances, the achieved cut sizes vary significantly (see Table 3). For example, the average gap between the worst and the best approach is 42.1% at 10 000 iterations and it still is 7.4% 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 2.3% and 0.8% on average, with a maximum of 4.7% and 6.3%t, i.e., for 10 000 and 100 000 evaluations.

To investigate the extent to which mutation performance and instance features are correlated, we perform a 2D projection using a principal component analysis of the instance feature space based on the features collected from Rossi and Ahmed (2015). We then consider the performance of the seven mutation operators at a budget of 100,000 evaluations, and we visualize it in the 2D space (see Fig. 2). In these projections, the very dense cluster in the top left is formed exclusively by the socfb-* instances, and the ridge from the very top left to the bottom left is made up of (from top to bottom) ia-*, tech-*, web*, and ca-* instances. The "outlier" on the right is web-BerkStan, due to its extremely high values of the average vertex degree, the number of triangles formed by three edges (3-cliques), the maximum triangles formed by an edge, and the maximum i-core number, where an i-core of a graph is a maximal induced subgraph and each vertex has degree at least i.

Interestingly, the performance seems to be correlated with the instance features and thus, indirectly, with their origin. For example, we can see in Fig. 2g that unif_1 does not reach a cut size that is within 1% of the best observed average for many of the socfb-* instances (shown as many black dots in the tight socfb*-cluster). In contrast to this, $\mathsf{pmut}_{3.5}$'s corresponding Fig. 2f shows only red dots, indicating that it always performs within 1% of the best-observed.

Lastly, we summarize the results in Fig. 2h based on the concept of instance difficulty. Here, the color denotes the number of instances that achieve a cut size within 1% of the best observed average. Interestingly, many ia-*, ca-*, web-* and tech-* instances are solved well by many mutation operators. In contrast to this, many socfb-* instances are blue, meaning that are solved well by just very few mutation operators—in particular, by our $\mathsf{pmut}_{3.5}$.

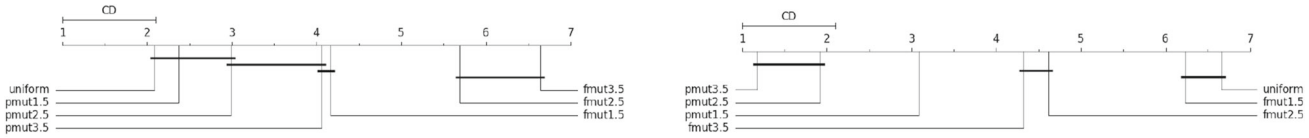
7.2 The symmetric mutual information problem

We study an instance of the general *feature selection* problem: Given a set of observations, find a subset of relevant features (variables, predictors) for use in model construction.

We consider the following framework. Suppose that n time series $X^{(1)}, \ldots, X^{(n)}$ are given, each one representing a

[1] In contrast to our earlier work (Friedrich, Göbel, et al., 2018), we are comparing against unif_1, which performs at least one flip, thus making it a fairer comparison.

[2] 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.

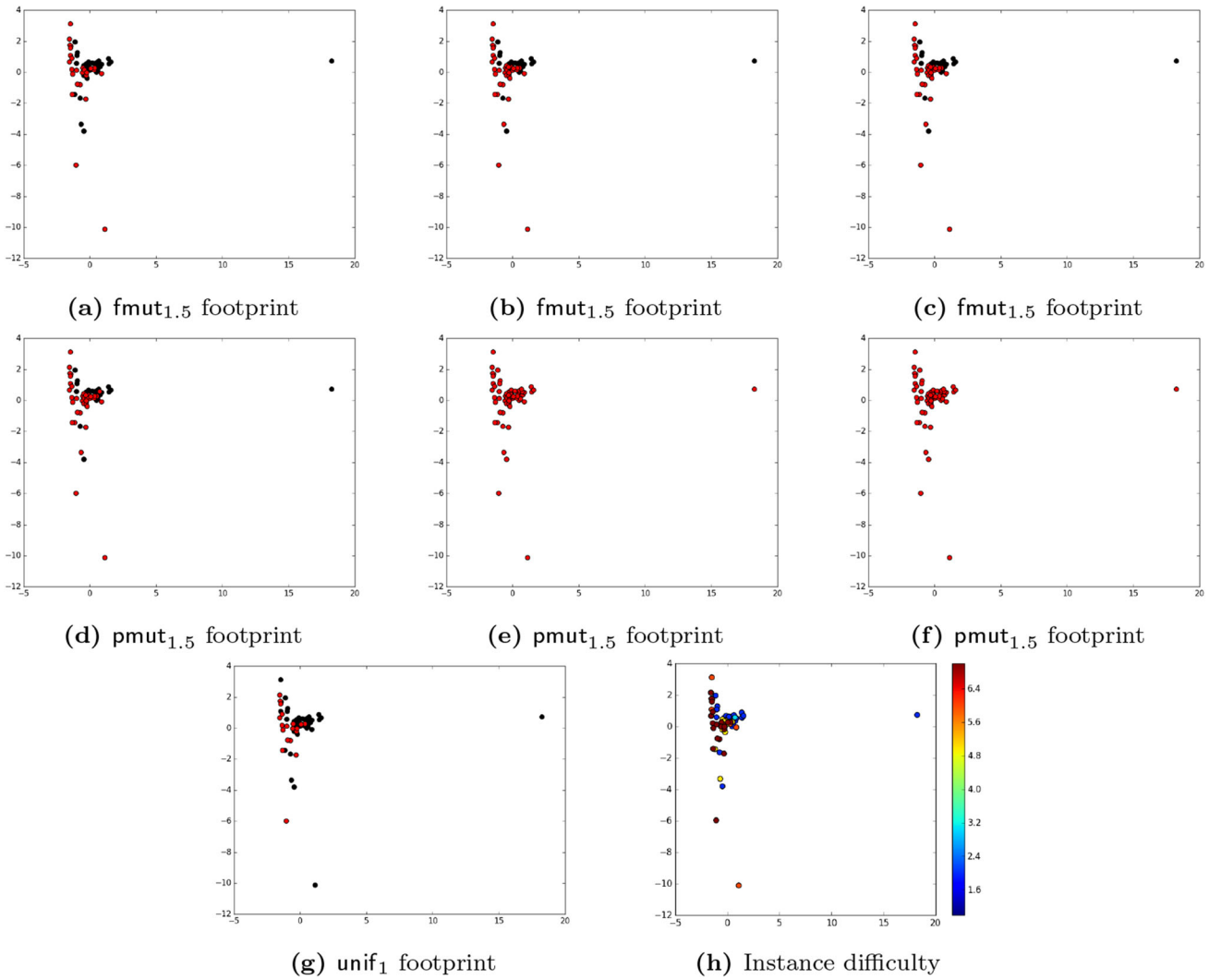


(a) 10 000 evaluations (b) 100 000 evaluations

Fig. 1 Critical Distance (CD) diagram based on a Nemenyi two-tailed test using the average rankings. CD (top left) shows the critical distance. Distances larger than CD correspond to a statistical significant difference in the ranking. Relationships within a critical distance are marked with a horizontal bar

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 2

	$t = 10k$		$t = 100k$	
	Total	$\mathsf{pmut}_{1.5}$ vs $\mathsf{fmut}_{1.5}$	Total	$\mathsf{pmut}_{3.5}$ vs $\mathsf{fmut}_{3.5}$
Min gap	0.8%	1.1%	0.0%	0.0%
Mean gap	13.0%	2.3%	1.9%	0.8%
Max gap	42.1%	4.7%	7.4%	6.3%



(a) $\mathsf{fmut}_{1.5}$ footprint (b) $\mathsf{fmut}_{1.5}$ footprint (c) $\mathsf{fmut}_{1.5}$ footprint

(d) $\mathsf{pmut}_{1.5}$ footprint (e) $\mathsf{pmut}_{1.5}$ footprint (f) $\mathsf{pmut}_{1.5}$ footprint

(g) unif_1 footprint (h) Instance difficulty

Fig. 2 Mutation operator footprints (**a**–**g**): instances are marked red if the mutation are at most 1% away from the best-observed performance. *Instance difficulty (h):* the color encodes the number of algorithms that perform within 1% of the oracle performance. Note: a principal component analysis is used for the projection of the instances from the feature space into 2D

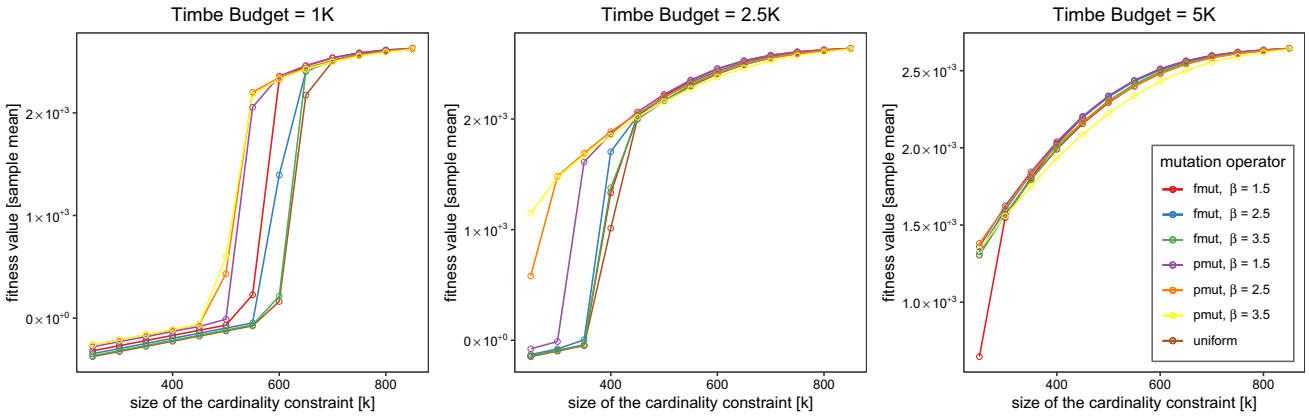


Fig. 3 Solution quality achieved by the $(1+1)$ EA with various mutation rates on a fitness function as in (9), for fixed cardinality constraint k, and varying time budget. We consider the $(1+1)$ EA with uniform mutation, pmut_β and fmut_β with $\beta = 1.5, 2.5, 3.5$. Each dot corresponds to the sample mean of 100 independent runs

sequence of temporal observations. For each sequence $X^{(i)}$, define the corresponding temporal variation as a sequence $Y^{(i)}$ with $Y^{(i)}_j = X^{(i)}_j - X^{(i)}_{j-1}$.

We perform feature selection of the variables $Y^{(i)}$, assuming that the joint probability distribution $p(Y^{(1)}, \ldots, Y^{(n)})$ is Gaussian. Specifically, given a cardinality constraint k, we search for a subset $S \in [n]$ of size at most k s.t. the corresponding series $\chi_S := \{Y^{(i)} : i \in S\}$ are optimal predictors for the overall variation in the model. Variations of this setting are found in many applications (Singh et al. 2009; Zhu and Stein 2006; Zimmerman 2006).

We use the *mutual information* as an optimization criterion for identifying highly informative random variables among the $\{Y^{(i)}\}$ (Caselton and Zidek 1984). For a subset $S \in [n]$, we define the corresponding mutual information as

$$\mathsf{MI}(S) = -\frac{1}{2}\sum_i (1-\rho_i^2), \tag{8}$$

where the ρ_i are the canonical correlations between χ_S and $\chi_{V\setminus S}$. It is well-known that the mutual information as in (8) is a symmetric non-negative submodular function (Krause et al. 2008). Note also that a cardinality constraint k is equivalent to a matroid constraint, with independent sets all subsets $S \in [n]$ of cardinality at most k. Hence, this approach to feature selection consists of maximizing a non-negative symmetric submodular function under a matroid constraint, as in Problem (4). Following the framework outlined in Sect. 6, we approach this problem by maximizing the following fitness function

$$z_{\mathsf{MI}}(S) = \begin{cases} \mathsf{MI}(S) & \text{if } |S| \leq k; \\ k - |S| & \text{otherwise;} \end{cases} \tag{9}$$

We apply this methodology to perform feature selection on an air pollution dataset (Rhode and Muller 2015)[3]. This dataset consists of hourly air NO_2 data from over 1500 sites, during a four month interval April 5-August 5, 2014.

For a fixed cardinality constraint $k = 200, \ldots, 850$, we let the $(1+1)$ EA with various mutation rates run for a fixed time budget at 1K, 2.5K, and 5K fitness evaluations. For each set of parameters, we perform 100 runs and take the sample mean over all resulting fitness values. We consider the $(1+1)$ EA with uniform mutation, pmut_β and fmut_β with $\beta = 1.5, 2.5, 3.5$. The results are displayed in Fig. 3.

We observe that for a small time budget and small k, heavy tailed-mutations outperform the standard uniform mutation and the fmut_β. We observe that for large k all mutation operators achieve similar performance. These results suggest that for a small time budget, and small k, larger jumps are beneficial, whereas standard mutation operators may be sufficient to achieve a good approximation of the optimum, given more resources.

We remark that Krause et al. (2008) show that it is possible to use a simple greedy algorithm to maximize the function $\mathsf{MI}(S)$ as in (8), if this function is ε-approximately monotone. Here, the constant ε of the approximate monotonicity depends on the discretization level of locations, and it affects the approximation guarantee. While in various applications it is reasonable to assume that ε is bounded, there are instance classes of submodular functions that are not approximately monotone (Lee et al. 2009). Without approximate monotonicity, the greedy algorithm yields poor performance. However, our (1+1) EA, which uses the properties of approximately local optimal solutions, maintains a constant-factor approximation guarantee on functions that are not approximately monotone.

[3] This dataset is publicly available at www.berkleyearth.org.

8 Conclusions

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

Acknowledgements The authors would like to thank Sören Tietböhl for helping in retrieving some of the data for the experiments. The authors would also like to thank Dr. Timo Kötzing, for useful discussions concering the applicability of our results

References

Ageev AA, Sviridenko M (1999) An 0.828-approximation algorithm for the uncapacitated facility location problem an 0.828-approximation algorithm for the uncapacitated facility location problem. Discrete Appl Math 93(2–3):149–156

Caselton WF, Zidek J (1984) Optimal monitoring network designs optimal monitoring network designs. Stat Probab Lett 2(4):223–227

Dasgupta D, Michalewicz Z (2013) Evolutionary algorithms in engineering applications. Springer, Berlin

Doerr B, Jansen T, Sudholt D, Winzen C, Zarges C (2013) Mutation rate matters even when optimizing monotonic functions mutation rate matters even when optimizing monotonic functions. Evol Comput 21(1):1–27

Doerr B, Johannsen D, Winzen C (2012) Multiplicative drift analysis. Algorithmica 64(4):673–697

Doerr B, Le HP, Makhmara R, Nguyen TD (2017) Fast genetic algorithms. In: Proceedings of GECCO, pp 777–784

Doerr C, Wagner M (2018a) Sensitivity of parameter control mechanisms with respect to their initialization. In: Proceedings of of PPSN, pp 360–372

Doerr C, Wagner M (2018b) Simple on-the-fly parameter selection mechanisms for two classical discrete black-box optimization benchmark problems. In: Proceedings of GECCO, pp 943–950

Droste S, Jansen T, Wegener I (2002) On the analysis of the (1+1) evolutionary algorithm. Theor Comput Sci 276(1–2):51–81

Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3(2):124–141

Eiben AE, Smith JE (2003) Introduction to evolutionary computation. Springer, Berlin

Feige U, Mirrokni VS, Vondrák J (2011) Maximizing non-monotone submodular functions. SIAM J Comput 40(4):1133–1153

Friedrich T, Göbel A, Quinzan F, Wagner M (2018a) Heavy-tailed mutation operators in single-objective combinatorial optimization. In: Proceedings of PPSN, pp 134–145

Friedrich T, He J, Hebbinghaus N, Neumann F, Witt C (2010) Approximating covering problems by randomized search heuristics using multi-objective models. Evol Comput 18(4):617–633

Friedrich T, Neumann F (2015) Maximizing submodular functions under matroid constraints by evolutionary algorithms. Evol Comput 23(4):543–558

Friedrich T, Quinzan F, Wagner M (2018b) Escaping large deceptive basins of attraction with heavy-tailed mutation operators. In Proceedings of GECCO, pp 293–300

Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J ACM 42(6):1115–1145

Håstad J (2001) Some optimal inapproximability results. J ACM 48(4):798–859

Jansen T, Wegener I (2006) On the analysis of a dynamic evolutionary algorithm. J Discrete Algorithms 4(1):181–199

Karp RM (1972) Reducibility among combinatorial problems. Complexity of computer computations. Springer, US, Boston, pp 85–103

Krause A, Guestrin C (2007) Near-optimal observation selection using submodular functions. In Proceedings of AAAI, pp 1650–1654

Krause A, Singh AP, Guestrin C (2008) Near-optimal sensor placements in gaussian processes: theory, efficient algorithms and empirical studies. J Mach Learn Res 9:235–284

Lee J, Mirrokni VS, Nagarajan V, Sviridenko M (2009) Non-monotone submodular maximization under matroid and knapsack constraints. In: Proceedings of STOC, pp 323–332

Lehmann B, Lehmann DJ, Nisan N (2006) Combinatorial auctions with decreasing marginal utilities. Games Econ Behav 55(2):270–296

Mitzenmacher M, Upfal E (2017) Probability and computing: randomization and probabilistic techniques in algorithms and data analysis. Cambridge University Press, Cambridge

Mühlenbein H (1992) How genetic algorithms really work: mutation and hillclimbing. In: Proceedings of PPSN, pp 15–26

Nemhauser GL, Wolsey LA (1978) Best algorithms for approximating the maximum of a submodular set function. Math Oper Res 3(3):177–188

Oliveto PS, He J, Yao X (2009) Analysis of the (1+1)-EA for finding approximate solutions to vertex cover problems. IEEE Trans Evol Comput 13(5):1006–1029

Qian C, Shi J, Tang K, Zhou Z (2018) Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. IEEE Trans Evol Comput 22(4):595–608

Qian C, Yu Y, Tang K, Yao X, Zhou Z (2017) Maximizing non-monotone/non-submodular functions by multi-objective evolutionary algorithms. CoRRabs/1711.07214

Rhode RA, Muller RA (2015) Air pollution in China: mapping of concentrations and sources. PLoS One 10(8):e0135749

Rossi RA, Ahmed NK (2015) The network data repository with interactive graph analytics and visualization (Website). http://networkrepository.com

Singh A, Krause A, Guestrin C, Kaiser WJ (2009) Efficient informative sensing using multiple robots. J Artif Intell Res 34:707–755

Wagner M, Friedrich T, Lindauer M (2017) Improving local search in a minimum vertex cover solver for classes of networks. In: Proceedings of CEC, pp 1704–1711

Wegener I (2001) Theoretical aspects of evolutionary algorithms. In: Proceedings of ICALP, pp 64–78

Welsh DJ (2010) Matroid theory. Courier Corporation

Witt C (2005) Worst-case and average-case approximations by simple randomized search heuristics. In: Proceedings of STACS, pp 44–56

Zhu Z, Stein ML (2006) Spatial sampling design for prediction with estimated parameters. J Agric Biol Environ Stat 11(1):24–44

Zimmerman DL (2006) Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction. Environmetrics 17(6):635–652