When Does Hillclimbing Fail on Monotone Functions: An entropy compression argument

Johannes Lengler* Anders Martinsson† Angelika Steger‡

Abstract

Hillclimbing is an essential part of optimization. An important benchmark for hillclimbing algorithms on functions $f : \{0,1\}^n \to \mathbb{R}$ are (strictly) *montone* functions, on which a surprising number of hillclimbers fail to be efficient. For example, the (1+1)-Evolutionary Algorithm is a standard hillclimber which flips each bit independently with probability c/n in each round. Perhaps surprisingly, this algorithm shows a phase transition: it optimizes any such monotone function in quasilinear time if $c < 1$, but there are monotone functions for which the algorithm needs exponential time if $c > 2.2$. But so far it was unclear whether the threshold is at $c = 1$.

In this paper we show that there exists a $c_0 > 1$ such that for all $0 < c \le c_0$ the $(1+1)$-Evolutionary Algorithm with rate c/n finds the optimum in $O(n \log^2 n)$ steps in expectation. Our proof is an adaptation of Moser's entropy compression argument. That is, we show that a long runtime would allow us to encode the random steps of the algorithm with less bits than their entropy.

1 Introduction

Hillclimbing is an essential part of optimization. The (1+1)-*Evolutionary Algorithm* or (1+1)-EA is a simple greedy hillclimbing scheme for maximizing an objective function $f : \{0,1\}^n \to \mathbb{R}$. A function of this form is called *pseudo-Boolean.* We start with a search point $X_0 \in \{0,1\}^n$ uniformly at random. In the t-th round we create an *offspring* X' from the *parent* X_t by flipping each bit of X_t independently with probability c/n, where c is the *mutation parameter.* Then we replace the current search point by X' if it has at least the same objective, i.e., we set $X_{t+1} := X'$ if $f(X') \ge f(X_t)$, and $X_{t+1} := X_t$ otherwise. The phrase $(1+1)$ reflects that in each round the next search point is chosen from one parent plus one offspring. It is clear that, on any function f with a unique global maximum, the $(1+1)$-EA will eventually fixate at at this maximum of f.

Here we study the performance of this algorithm on (strictly) monotone functions. A pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is said to be *monotone*[1] if $f(x) < f(y)$ whenever $x \ne y$ and $x^i \le y^i$ for all $i \in [n]$, where x^i denote the i-th coordinate of x. For any such function, the unique global maximum is the all-ones string. Monotone functions are an important class of benchmark functions for hillclimbing schemes, since there exist a large variety of hillclimbing schemes that optimize all monotone functions efficiently. For example, the (1+1) algorithm that creates the offspring by flipping exactly one random bit in each round resembles a coupon collector process, and thus finds the optimum in time $O(n \log n)$.[2] Nevertheless, a surprising number of hillclimbing schemes fail on some monotone functions, see [17] for an overview.

For the $(1+1)$-EA, while for any constant $c < 1$ it is easy to see that the algorithm needs time $O(n \log n)$ to find the optimum of any monotone function [9], it was shown in a sequence of papers [9, 10, 18] that for $c > 2.13\ldots$ there are monotone functions (dubbed HotTopic functions in [17]) on which the algorithm needs exponential time. The standard proof techniques for upper runtime bounds fail precisely at $c = 1$, and there were split opinions in the community on whether there should be a phase transition from polynomial to exponential at $c = 1$ [8]. On the presumed threshold $c = 1$ it follows from a more general model of Jansen [15] that the runtime is $O(n^{3/2})$, but it remained unclear whether the runtime is quasilinear or not.

The value $c = 1$ is of special interest, for several reasons. From a practical perspective, it is considered

*Department of Computer Science, ETH Zürich, Switzerland
†Department of Computer Science, ETH Zürich, Switzerland
‡Department of Computer Science, ETH Zürich, Switzerland

[1] We define "monotone" in a way that otherwise might rather be called "strictly monotone", for ease of terminology. Note that we can't expect efficient runtimes for functions which are monotone in a non-strict sense. For example, we could have $f(x) = 0$ for $x \not\equiv 1$ and $f(1) = 1$ otherwise and the search for the optimal solution amounts to a blind random walk. Therefore, we define monotone functions in this strict sense in our paper.

[2] This algorithm is called *Random Local Search.* Despite its good behaviour on monotone functions it has severe limitations in practice. For example, other than the (1+1)-EA it can't escape local optima and is highly susceptible to any form of noise.

the standard choice for the mutation parameter and explicitly recommended by textbooks on the subject [3, 4]. From a theoretical perspective, $c = 1$ is known to be the optimal parameter choice for linear functions, i.e., for functions of the form $f(X) = \sum_{i=1}^{n} w_i X^i$, where the w_i are fixed weights. More precisely, the choice $c = 1$ gives runtime $(1 + o(1))en \log n$ on any linear function, while any other (constant) choice of c gives a strictly worse leading constant on any linear function [23].

In this paper we use an entropy compression argument to show that there is an $\varepsilon > 0$ such that for $c = 1 + \varepsilon$ the runtime remains quasilinear for all monotone pseudo-Boolean functions. More precisely, we show that a long runtime would allow us to encode the random trajectory of the algorithm with fewer bits than its entropy, which is an information theoretic contradiction. This type of argument is attributed to Moser, who used the technique in his celebrated algorithmic proof of the Lovász local lemma [19, 13]. Since then, the method has been used to extend and apply the local lemma [20, 2], and used to some extent for colouring problems [1, 21, 12, 14, 11]. The same idea has been used for analysing optimal data structures, e.g., for cell probing [6, 16] and for sampling [5]. Despite these results, and despite popular blogposts [22, 13], the method still does not seem to be widely known outside of these communities.

We use this technique to prove that no phase transition occurs at $c = 1$. More precisely, we show the following.

THEOREM 1.1. *There exists an $\varepsilon > 0$ such that, for any (strictly) monotone function f and any constant $0 < c \leq 1 + \varepsilon$, the $(1 + 1)$-EA with mutation parameter c requires an expected number of $O(n \log^2 n)$ steps until it finds the maximum of f, and it visits an expected number of $O(n)$ search points. The same remains true if the initial search point of the algorithm is chosen by an adversary.*

To be more precise, we show that there are constants $\varepsilon, C > 0$ such that for all $0 < c \leq 1 + \varepsilon$ the runtime is at most $C/c \cdot n \log^2 n$, and the number of search points is at most $C \cdot n$.

To apply an entropy compression argument, it turns out to be natural to not measure performance in the number of time steps, but in the number of *updates*, i.e. the number of times $X_{t+1} \neq X_t$. We show that the expected number of updates until the algorithm finds the maximum of any monotone f is $O(n)$. Towards the end of the proof, we link the expected number of updates to the expected number of time steps and show that they only differ by a polylogarithmic factor.

We first give some intuition on the behaviour of the algorithm and on our proof. For a general monotone function f, it is natural to measure the progress of the $(1 + 1)$-EA is terms of number of one-bits in the current search point. In order to make an update, it is necessary to flip at least one zero-bit into a one-bit, since otherwise the offspring would be rejected due to monotonicity (unless it is identical to the parent, in which case there is no update either). Thus, if the average update does not flip too many ones to zeros, the number of ones in the current search point will tend to n efficiently. For a small mutation parameter (specifically for $c < 1$), this is indeed true as the average number of ones flipped to zeros is at most c, and this remains true for update steps. For larger c, one might expect this to still hold as, intuitively, any offspring with more ones flipped to zeros than zeros flipped to ones should be unlikely to be fitter than its parent. To see why this intuition fails for large c, consider the following situation for a linear objective function. If a zero-bit of high weight is flipped to a one, then the algorithm may accept the offspring even if many low weight one-bits are flipped at the same time. The larger c is, the more such one-bits are flipped in expectation. While such bad steps do not occur too often for linear functions, it has been shown that there exist monotone (locally linear) functions, such as HOTTOPIC functions [17], where this effect keeps the algorithm away from the optimum for exponentially long time.

Based on this intuition, we define *good* and *bad* updates. The bad updates capture cases in which a zero-bit with a disproportionally high weight is flipped. Then we study the entropy of the update steps of the algorithms. On the one hand, we will analyze the algorithm in a forward manner to give a lower bound on the entropy of each update. On the other hand, we give a backwards encoding (from last step to first) of the updates steps, and this encoding saves some bits in bad update steps. Since the expected number of bits needed for the encoding is lower bounded by the entropy, we get an upper bound for the expected number of bad update steps. This, in turn, gives us a linear upper bound for the expected number of update steps. Finally, the runtime bound follows by a slight refinement of the calculation, in which we compute how many steps we need to decrease the number of zero-bits from 2^m to 2^{m-1}, for $m = \log n, \ldots, 1$.

2 Preliminaries: properties of single updates

In this section we collect properties of single update steps. Throughout this section we will use the following notation. We assume that we are in an (arbitrary, but fixed) state $y \in \{0, 1\}^n$. We also assume that k denotes the number of ones in y. We denote by Y' the string obtained by flipping each coordinate independently with probability c/n. With $\mathcal{E}_{keep}$ we denote the event that $f(Y') \geq f(y)$, corresponding to the event that Y' is accepted as the new state. We also denote by U the number of bits that are zero in y, but one in Y' (upflips) and by D the number of bits that are one in y, but zero in Y' (downflips).

In this section we will repeatedly make use of the following fact. Let X be a random variable, and $\mathcal{E}$ be some event. With $\mathbb{1}_{\mathcal{E}}$ we denote the indicator variable for the event $\mathcal{E}$. Then by the law of conditional expectation,

$$\mathbb{E}[X|\mathcal{E}] = \mathbb{E}[X \cdot \mathbb{1}_{\mathcal{E}}]/\Pr[\mathcal{E}]. \tag{2.1}$$

2.1 Expected number of bits flips $\mathbb{E}[U + D]$ is easily computed by linearity of expectation to be equal to c. However, we are interested in $\mathbb{E}[U + D \mid \mathcal{E}_{keep}]$. Observe that $\Pr[\mathcal{E}_{keep}]$ is at least the probability that we flip exactly one zero-bit to a one and no other bit. Thus, $\Pr[\mathcal{E}_{keep}] \geq (n-k)\frac{c}{n}(1-\frac{c}{n})^{n-1}$. To estimate the latter term, we will use the general bounds for the exponential $(1-c/n)^n \leq e^{-c} \leq (1-c/n)^{n-1}$, where the first part follows from Bernoulli's inequality for all $0 \leq c \leq n$, and the second inequality holds for all $0 \leq c < 2$ and n sufficiently large, and follows from the expansion $1 - c/n = e^{-c/n - c^2/(2n^2) + O(1/n^3)}$ by raising both sides to the $(n-1)$-st power. In our case, it implies $\Pr[\mathcal{E}_{keep}] \geq \frac{n-k}{n} c e^{-c}$. Observe also that $\mathbb{1}_{\mathcal{E}_{keep}} \leq \mathbb{1}_{\mathcal{A}_1} + \ldots + \mathbb{1}_{\mathcal{A}_{n-k}}$, where $\mathcal{A}_i$ denotes the event that we flip the i-th zero-bit to a one, where $1 \leq i \leq n-k$. Thus (2.1) implies that for $0 < c < 2$, and n sufficiently large,

$$\begin{aligned}\mathbb{E}[U + D \mid \mathcal{E}_{keep}] &\leq \frac{\sum_{i=1}^{n-k} \mathbb{E}[(U+D)\cdot \mathbb{1}_{\mathcal{A}_i}]}{\Pr[\mathcal{E}_{keep}]} \\ &\leq \frac{(n-k)\frac{c}{n}(1+c)}{\frac{n-k}{n} c e^{-c}} = (1+c)e^c.\end{aligned} \tag{2.2}$$

2.2 Change in the number of ones Our goal in this section is to (lower) bound the change in the number of ones, i.e., to bound $\mathbb{E}[U - D \mid \mathcal{E}_{keep}]$. Clearly, $\mathbb{E}[U \mid \mathcal{E}_{keep}] \geq 1$ by monotonicity. To bound D, note that it is intuitively clear that $\mathbb{E}[D \mid \mathcal{E}_{keep}] \leq \mathbb{E}[D] \leq c$. A full proof can be found in the appendix. For $c < 1$ we thus get from (2.1) that $\mathbb{E}[U - D|\mathcal{E}_{keep}] \geq 1 - c$. Standard drift arguments thus imply that the expected number of updates till EA reaches the all-ones string is $O(n)$.

In order to also be able to apply a similar argument for $c > 1$, we need to be more careful. What we will do is to partition updates into good and bad ones, i.e., we let $\mathcal{E}_{keep} = \mathcal{E}_{good} \uplus \mathcal{E}_{bad}$ and define $\mathcal{E}_{bad}$ in such a way that $\mathcal{E}_{bad}$ happens "rarely" and $\mathbb{E}[U - D|\mathcal{E}_{good}]$ is positive.

To do so, observe that whenever $U = 1$, say bit i is flipped from a zero to a one, we can attribute a *value* to each 1-bit in $y + e_i$ (that is, to the indices $j \in \{i\} \cup \{a \in [n] : y^a = 1\}$) according to

$$val_{y+e_i}(j) := f(y + e_i) - f(y + e_i - e_j),$$

where $e_1, e_2, \ldots, e_n$ denotes the standard basis vectors, and $+$ and $-$ denotes vector addition and subtraction respectively. It is natural to think of this value as the "cost" of flipping bit j to a zero. Indeed if Y' is obtained from y by flipping bit i from a zero to a one, and bits $j_1, j_2, \ldots$ from ones to zeros, then if at least one of the j-bits, say j_1, has a strictly higher value than i, we have

$$\begin{aligned}f(Y') - f(y) &\leq f(y + e_i - e_{j_1}) - f(y) \\ &= val_{y+e_i}(i) - val_{y+e_i}(j_1) < 0,\end{aligned}$$

which means such a Y' will never be kept. Similarly, if j_1 has equal value to i, and this is not the only 1-bit flipped to a zero, then Y' will not be kept.

With this notion at hand we say that an update from y to Y' belongs to $\mathcal{E}_{bad}$ iff

1. $Y' \neq y$ and $f(Y') \geq f(y)$, i.e., we actually make an update,
2. there is exactly one zero-bit i that is flipped, i.e, $U = 1$, and
3. there exist at least $(1-\alpha)n$ one-bits in $y + e_i$ whose value is strictly smaller than the value of bit i.

To get some intuition behind this definition, observe that it indeed captures cases in which the number of ones may likely *decrease*: we only flip one bit from zero to one *and* there are many candidate bits for which we may be able to flip two or more of them back to zero. This intuition is formalized by the following proposition.

PROPOSITION 2.1. *For any $0 \leq \alpha \leq \frac{1}{2}$ and $0 \leq c \leq 1/(1-\alpha)$, we have for all $n \geq 3$*

$$\begin{aligned}\mathbb{E}[U - D \mid \mathcal{E}_{bad}] &\geq 1 - c, \\ \mathbb{E}[U - D \mid \mathcal{E}_{good}] &\geq e^{-2\alpha c}\left(1 - (1-\alpha)c\right).\end{aligned}$$

Proof. Let $\mathcal{B}_i$ denote the event that the i-th one-bit in y gets flipped in Y'. Then, by linearity of expectation, we have $\mathbb{E}[U - D \mid \mathcal{E}_{bad}] \geq 1 - \sum_{i=1}^{k} \Pr[\mathcal{B}_i \mid \mathcal{E}_{bad}]$. By Bayes' Theorem, we have $\Pr[\mathcal{B}_i \mid \mathcal{E}_{bad}] = \frac{c}{n} \cdot \frac{\Pr[\mathcal{E}_{bad}|\mathcal{B}_i]}{\Pr[\mathcal{E}_{bad}]} \leq \frac{c}{n}$, where the last step follows by a simple coupling argument, as in Section 2.2. Hence $\mathbb{E}[U - D \mid \mathcal{E}_{bad}] \geq 1 - c\frac{k}{n} \geq 1 - c$, as desired.

As for $\mathcal{E}_{good}$, we will show

$$\mathbb{E}[U - D \mid \mathcal{E}_{good}] \geq \left(1 - \frac{c}{n}\right)^{\alpha n} (1 - (1-\alpha)c)\,. \tag{2.3}$$

The proposition will then follow from $1 - (1-\alpha)c \geq 0$, since $1 - x \geq e^{-2x}$ for all $0 \leq x \leq 2/3$. To prove (2.3), let $\mathcal{U}$ denote the set of indices of zero-bits in y that get flipped to one-bits in Y'. Then

$$\mathbb{E}[U - D \mid \mathcal{E}_{good}] = \sum_A \Big(\Pr[\mathcal{U} = A \mid \mathcal{E}_{good}] \cdot \\ \cdot \mathbb{E}[U - D \mid \mathcal{E}_{good} \cap \{\mathcal{U} = A\}] \Big).$$

Thus, it suffices to estimate $\mathbb{E}[U - D \mid \mathcal{E}_{good} \cap \{\mathcal{U} = A\}]$ for any set $A \subset [n]$ such that $\Pr[\mathcal{U} = A | \mathcal{E}_{good}]$ is non-zero.

If $|A| \geq 2$, the same argument as for $\mathcal{E}_{bad}$ gives

$$\mathbb{E}[U - D \mid \mathcal{E}_{good} \cap \{\mathcal{U} = A\}] \geq 2 - c.$$

It remains to consider the case of $|A| = 1$, say $A = \{i\}$. In this case, let $k' = \min(k, \lfloor (1-\alpha)n \rfloor)$, and order the one-bits in y, $j_1, j_2, \ldots j_k$, in descending order with respect to $val_{y+e_i}(j)$ with ties broken arbitrarily. In order for $\mathcal{U} = A$ to be compatible with a good update, we can assume that the values of $j_1, j_2, \ldots j_{k-k'}$ must be greater than or equal to the value of i.

With these definitions at hand, we write

$$\mathbb{E}[U - D \mid \mathcal{E}_{good} \cap \{\mathcal{U} = A\}] \\ = \frac{1}{\Pr[\mathcal{E}_{good} \mid \mathcal{U} = A]} \mathbb{E}[\mathbb{1}_{\mathcal{E}_{good}}(U - D) \mid \mathcal{U} = A].$$

Note that, conditioned on $\mathcal{U} = A$, $\mathbb{1}_{\mathcal{E}_{good}}(U - D) = 0$ whenever one of the bits $j_1, \ldots j_{k-k'}$ are flipped. This is because this is either the only bit flipped to a zero, in which case $U - D = 1 - 1 = 0$, or one additional bit is flipped to a zero, in which case $f(Y') < f(y)$. Whenever the bits $j_1, \ldots j_{k-k'}$ remain ones, on the other hand, we can lower bound $\mathbb{1}_{\mathcal{E}_{good}}(U - D)$ by one minus the number of bits among $j_{k-k'+1}, \ldots j_k$ that get flipped. Thus

$$\mathbb{E}[\mathbb{1}_{\mathcal{E}_{good}}(U - D) \mid \mathcal{U} = A] \geq (1 - \frac{c}{n})^{k-k'}(1 - \frac{c}{n}k').$$

By assumption, we have $1 - \frac{c}{n}k' \geq 1 - (1-\alpha)c \geq 0$ which means that

$$\mathbb{E}[U - D \mid \mathcal{E}_{good} \cap \{\mathcal{U} = A\}] \\ \geq \mathbb{E}[\mathbb{1}_{\mathcal{E}_{good}}(U - D) \mid \mathcal{U} = A] \\ \geq (1 - \frac{c}{n})^{\alpha n}(1 - (1-\alpha)c),$$

as desired. The proposition follows by observing that $2 - c \geq 1 - (1-\alpha)c$ as $2 - c - 1 + (1-\alpha)c = 1 - \alpha c \geq 1 - \frac{\alpha}{1-\alpha} \geq 0$, where in the second to last step we used $c \leq 1/(1-\alpha)$.

Note that for any fixed $0 < \alpha \leq 1/2$ the considered range in Proposition 2.1 also contains some values $c > 1$. Moreover, for the considered range the lower bound for $\mathbb{E}[U - D \mid \mathcal{E}_{good}]$ is positive. If we could thus show that $\mathcal{E}_{bad}$ occurs only sufficiently rarely, then we might hope to be able to bound the number of updates. This is what we will do with the entropy compression argument.

2.3 Entropy of an update step In this section we study the entropy of a single update starting from a fixed state y. We refer the reader who is not familiar with information theory to the introduction in [7]. Naturally, the entropy will depend on y. Recall that the random variables U and D denote the number of upflips and downflips, respectively.

PROPOSITION 2.2. *For any $c < 4/3$, any $0 \leq k < n$ and any $y \in \{0,1\}^n$ with exactly k ones we have*

$$\mathbb{H}(Y' \mid \mathcal{E}_{keep}) \geq \mathbb{E}\left[\log_2\left(\binom{n-k}{U}\binom{k+U}{D}\right)\middle|\mathcal{E}_{keep}\right],$$

where $\mathbb{H}(Y' \mid \mathcal{E}_{keep})$ denotes the binary entropy of the conditional distribution of Y' given $\mathcal{E}_{keep}$.

Proof. Let $\mathcal{A}_{u,d}$ denote the set of all strings $z \in \{0,1\}^n$ such that $f(z) \geq f(y)$ and such that z can be obtained from y by flipping precisely u zeros to ones and d ones to zeros. Then

$$p_{keep} := \Pr[\mathcal{E}_{keep}] \\ = \sum_{u=1}^{n-k}\sum_{d=0}^{k}\sum_{z \in \mathcal{A}_{u,d}} \left(\frac{c}{n}\right)^{u+d}\left(1 - \frac{c}{n}\right)^{n-u-d}.$$

To simplify notation we write

$$q_{u,d} := |\mathcal{A}_{ud}| \left(\frac{c}{n}\right)^{u+d}\left(1 - \frac{c}{n}\right)^{n-u-d}$$

and

$$a_{u,d} := \left(\frac{c}{n}\right)^{u+d}\left(1 - \frac{c}{n}\right)^{n-u-d}.$$

By the definition of entropy we have

$$\mathbb{H}(Y' \mid \mathcal{E}_{keep}) \\ = -\frac{1}{p_{keep}}\sum_{u=1}^{n-k}\sum_{d=0}^{k}\sum_{z \in \mathcal{A}_{u,d}} a_{ud}\log_2\left(\frac{a_{u,d}}{p_{keep}}\right) \\ = \frac{1}{p_{keep}}\sum_{u=1}^{n-k}\sum_{d=0}^{k} q_{ud}\log_2\left(\frac{p_{keep}}{a_{u,d}}\right). \tag{2.4}$$

We want to show that this is at least as large as the expectation in the statement of the proposition. With

$$b_{u,d} := \binom{n-k}{u}\binom{k+u}{d}$$

we can write this expectation as

$$\begin{aligned}
&\mathbb{E}\left[\log_2\left(\binom{n-k}{U}\binom{k+U}{D}\right)\middle|\mathcal{E}_{keep}\right] \\
(2.5)\qquad &= \frac{1}{p_{keep}}\sum_{u=1}^{n-k}\sum_{d=0}^{k} q_{u,d}\log_2\left(b_{u,d}\right).
\end{aligned}$$

Hence the proposition follows if we can show that the difference (2.4) minus (2.5) is non-negative. That is, we have to show that

$$\frac{1}{p_{keep}}\sum_{u=1}^{n-k}\sum_{d=0}^{k} q_{u,d}\log_2\left(\frac{p_{keep}}{a_{u,d}b_{u,d}}\right) \overset{!}{\geq} 0.$$

Multiplying by $p_{keep} \geq 0$ and partitioning the log amounts to showing that
(2.6)

$$\Delta := p_{keep}\log_2 p_{keep} + \sum_{u=1}^{n-k}\sum_{d=0}^{k} q_{u,d}\log_2\left(\frac{1}{a_{u,d}b_{u,d}}\right) \overset{!}{\geq} 0.$$

To this end, an elementary calculation shows that the product $a_{u,d}b_{u,d}$ is either maximized by the case $u = 1$, $d = 0$, or by $u = d = 1$. We defer the calculation to the appendix.

Assume first that $a_{1,1}b_{1,1} \leq a_{1,0}b_{1,0}$. Then the left hand side in (2.6) satisfies

$$\Delta \geq p_{keep}\log_2 p_{keep} + p_{keep}\log_2(\frac{1}{a_{1,0}b_{1,0}})$$

which is non-negative, as $p_{keep} \geq q_{1,0} = a_{1,0}b_{1,0}$. For the other case, assume $a_{1,1}b_{1,1} > a_{1,0}b_{1,0}$. Writing $p_{keep} = (1+x)q_{1,0}$ for some $x \geq 0$ and noting that $q_{1,0} = a_{1,0}b_{1,0}$ and $\sum_{u,d\geq 1} q_{u,d} = xq_{1,0}$ we obtain

$$\begin{aligned}
\Delta &\geq (1+x)q_{1,0}\log_2\left((1+x)q_{1,0}\right) + \\
&\quad + q_{1,0}\log_2\left(\frac{1}{a_{1,0}b_{1,0}}\right) + xq_{1,0}\log_2\left(\frac{1}{a_{1,1}b_{1,1}}\right) \\
&= q_{1,0}\cdot\left((1+x)\log_2(1+x) + x\log_2\left(\frac{a_{1,0}b_{1,0}}{a_{1,1}b_{1,1}}\right)\right).
\end{aligned}$$

Now observe that $\frac{a_{1,0}b_{1,0}}{a_{1,1}b_{1,1}} = \frac{1-c/n}{c}\frac{n}{k+1} > 3/4$, for n sufficiently large. The claim now follows from $(1+x)\log_2(1+x) + x\log_2(3/4) \geq 0$ for all $x \geq 0$.

3 Entropy of the Markov chain

The aim of this section is to provide bounds on the entropy of the sequence of updates in the (1 + 1)-EA. Given the sequence $(X_t)_{t=0}^{\infty}$ of search points, as generated by the algorithm when started in some fixed state $X_0 = x \in \{0,1\}^n$, we define T as the number of updates, that is, the number of times $t = 0, 1, \ldots$ such that $X_t \neq X_{t+1}$. For each $t = 0, 1, \ldots T$, we denote by Y_t the state of the algorithm after t update steps. To simplify notation later on, we want Y_t be defined for all $t \geq 0$, so we define Y_t to be the all-one string if $t > T$. We note that as $(X_t)_{t=0}^{\infty}$ is a Markov chain, so is $(Y_t)_{t=0}^{\infty}$. More precisely, the transition probabilities of $(Y_t)_{t=0}^{\infty}$ are the ones obtained from transition probabilities of $(X_t)_{t=0}^{\infty}$ by removing self-transitions from all states besides the all ones state.

We will use $\#0_t$, $\#1_t$ to denote the number of zero-bits and one-bits in Y_t, respectively. Furthermore, we use U_t and D_t to denote the number of upflips (zero-to-one) and downflips (one-to-zero) from Y_t and Y_{t+1}, respectively.

Recall that the entropy of (the trajectory of) the Markov chain $(Y_t)_{t=0}^{\infty}$, as described above, can be written as

$$\mathbb{H}((Y_t)_{t=0}^{\infty}) = \sum_{t=0}^{\infty}\mathbb{H}(Y_{t+1} \mid Y_t),$$

where

$$\mathbb{H}(Y_{t+1} \mid Y_t) = -\sum_{y}\Pr[Y_t = y]\mathbb{H}(Y_{t+1} \mid Y_t = y).$$

This entropy is finite, since the Markov chain will converge to the absorbing state almost surely. The next two propositions bound this entropy from below and above.

Proposition 3.1.

$$\mathbb{H}((Y_t)_{t=0}^{\infty}) \geq \mathbb{E}\left[\sum_{t=0}^{T-1}\log_2\left(\binom{\#0_t}{U_t}\binom{\#1_t + U_t}{D_t}\right)\right].$$

Proof. Observe that, for any y, the term $\mathbb{H}(Y_{t+1} \mid Y_t = y)$ does not depend on t, as $(Y_t)_{t=0}^{\infty}$ is a time-homogeneous Markov chain. If we thus let $g(y) := \mathbb{H}(Y_{t+1} \mid Y_t = y)$, we get

$$\mathbb{H}(Y_{t+1} \mid Y_t) = \mathbb{E}[g(Y_t)]$$

and thus

$$\begin{aligned}
\mathbb{H}((Y_t)_{t=0}^{\infty}) &\geq \sum_{t=0}^{\infty}\mathbb{E}\left[g(Y_t)\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} g(Y_t)\right] = \mathbb{E}\left[\sum_{t=0}^{T-1} g(Y_t)\right],
\end{aligned}$$

where in the last step we have used that the conditional entropy of an update is zero once we have reached the all-ones state. As Proposition 2.2 implies that $\mathbb{E}[g(Y_t)] \geq \mathbb{E}[\log_2\left(\binom{\#0_t}{U_t}\binom{\#1_t+U_t}{D_t}\right)]$, the proposition follows.

PROPOSITION 3.2. *There exists a constant C such that for all $0 < \alpha < 1$ and for all $0 < c < 2$,*

$$\mathbb{H}((Y_t)_{t=0}^{\infty}) \leq C\mathbb{E}[T] - \log_2(1/\alpha)\mathbb{E}[T_{bad}] + \\ + \mathbb{E}\left[\sum_{t=0}^{T-1} \log_2\left(\binom{\#0_{t+1}}{D_t}\binom{\#1_{t+1}+D_t}{U_t}\right)\right],$$

where T_{bad} denotes the number of bad updates (as defined in Section 2.2).

Proof. Recall that the entropy represents a lower bound on the expected number of bits needed to represent all information of the process. Thus, the expected length of any encoding strategy for the traces of the chain will form an upper bound on the entropy. We proceed as follows. We encode the process *backwards*, i.e. we start from the all-ones vector that is the unique absorbing state of the process. For each update we encode

(i) whether the update is good or bad,

(ii) the number of downflips D_t and the number of upflips U_t,

(iii) the actual choice of which D_t zero-bits in Y_{t+1} were the bits that were flipped from one to zero in the update from Y_t to Y_{t+1}, and

(iv) the actual choice of which U_t one-bits in Y_{t+1} were the bits that were flipped from zero to one.

To mark the global end of our encoding, we are somewhat wasteful: we start the encoding of each update with a one-bit and conclude the whole encoding with a single zero-bit. For each update we encode (i) with a single bit, and D_t and U_t with $D_t + U_t + 2$ bits using a unary encoding. From (2.2) we deduce that there exists a constant $C > 0$ such the expected length of the encoding of (i) and (ii) for all updates is bounded by $C\mathbb{E}[T]$. For the encoding of (iii) observe that the D_t bits that correspond to downflips have to be chosen from the zero-bits in Y_{t+1}. We thus can encode the actual choice by $\lceil \log_2 \binom{\#0_{t+1}}{D_t} \rceil$ bits. For the encoding in (iv) we distinguish between good and bad updates. For a good update we proceed similarly as in (iii). As the D_t bits that correspond to upflips have to be chosen from one-bits in Y_{t+1}, we can encode the actual choice by $\lceil \log_2 \binom{\#1_{t+1}}{U_t} \rceil$ bits. For a bad update we can be more efficient in (iv). Observe first that a bad update implies $U_t = 1$. We thus need to specify only a single bit. Recall also that the definition of bad updates implies that there exist at least $(1-\alpha)n$ one-bits in Y_t that have lower value than the bit that we want to flip. As the number of one-bits in Y_t is bounded by $\#1_{t+1} + D_t$ we thus see that we have at most $\#1_{t+1} + D_t - (1-\alpha)n \leq \alpha(\#1_{t+1} + D_t)$ bits from which we can choose the bit that corresponds to the (single) upflip. We can thus encode the choice of this bit with $\lceil \log_2(\alpha(\#1_{t+1} + D_t)) \rceil$ bits, which is less than $\lceil \log_2 \binom{\#1_{t+1}+D_t}{U_t} \rceil - \log_2(1/\alpha) + 1$ bits. The claimed bound in the proposition follows by collecting all terms.

4 Proof of the theorem

We first obtain a bound on the expected number of bad updates by comparing upper and lower bounds on the entropy of the Markov chain.

PROPOSITION 4.1. *If the algorithm starts in a state with exactly k ones, then*

$$\mathbb{E}[T_{bad}] \leq \tfrac{C}{\log_2(1/\alpha)}\mathbb{E}[T] + \tfrac{1}{\log_2(1/\alpha)}\log_2\binom{n}{k},$$

where C is the constant from Proposition 3.2.

Proof. Collecting and rearranging the terms from Propositions 3.1 and 3.2 we get

(4.7)
$$\mathbb{E}[T_{bad}] \leq \tfrac{C}{\log_2(1/\alpha)}\mathbb{E}[T] + \\ + \frac{1}{\log_2(1/\alpha)}\mathbb{E}\left[\log_2\left(\prod_{t=0}^{T-1} \frac{\binom{\#0_{t+1}}{D_t}\binom{\#1_{t+1}+D_t}{U_t}}{\binom{\#0_t}{U_t}\binom{\#1_t+U_t}{D_t}}\right)\right].$$

Using the formulas $\#1_{t+1} = \#1_t + U_t - D_t$ and $\#0_{t+1} = \#0_t - U_t + D_t$, it is easy to see that, for any $0 \leq t \leq T-1$,

$$\frac{\binom{\#0_{t+1}}{D_t}\binom{\#1_{t+1}+D_t}{U_t}}{\binom{\#0_t}{U_t}\binom{\#1_t+U_t}{D_t}} = \frac{\#0_{t+1}!}{\#0_t!} \cdot \frac{\#1_{t+1}!}{\#1_t!}.$$

Hence, the product in (4.7) is telescoping, and we get

$$\mathbb{E}[T_{bad}] \leq \tfrac{C}{\log_2(1/\alpha)}\mathbb{E}[T] + \\ + \frac{1}{\log_2(1/\alpha)}\mathbb{E}\left[\log_2\left(\frac{\#0_T! \cdot \#1_T!}{\#0_0! \cdot \#1_0!}\right)\right].$$

The claim now follows from $\frac{\#0_T! \cdot \#1_T!}{\#0_0! \cdot \#1_0!} = \binom{n}{\#1_0} / \binom{n}{\#1_T} = \binom{n}{\#1_0}$, as $\#1_0 = k$ and $\#1_T = n$.

From this and Proposition 2.1 we obtain an upper bound on the number of updates.

PROPOSITION 4.2. *There exists an $\varepsilon > 0$ and $\beta > 0$ such that for any $0 < c \leq 1 + \varepsilon$ the following holds. If the algorithm starts in a state with exactly k ones, then*

$$\mathbb{E}[T] \leq \beta(n-k) + \beta\log_2\binom{n}{k}.$$

Proof. Observe that $\sum_{t=0}^{\infty}(\#1_{t+1} - \#1_t) = n - \#1_0$. If we thus start in a state with exactly k ones, then whenever $0 \le \alpha \le 1/2$ and $0 < c \le 1/(1-\alpha)$ we get from linearity of expectation and Proposition 2.1 that

$$\begin{aligned} n-k &= \sum_{t=0}^{\infty} \mathbb{E}[\#1_{t+1} - \#1_t] \\ &= \sum_{t=0}^{\infty} \Big(\mathbb{E}[\#1_{t+1} - \#1_t | \mathcal{E}_{good}(t)] \cdot \Pr[\mathcal{E}_{good}(t)] + \\ &\quad + \mathbb{E}[\#1_{t+1} - \#1_t | \mathcal{E}_{bad}(t)] \cdot \Pr[\mathcal{E}_{bad}(t)] \Big) \\ &\overset{P2.1}{\ge} e^{-2\alpha c}(1-(1-\alpha)c) \cdot \mathbb{E}[T - T_{bad}] + \\ &\quad + (1-c) \cdot \mathbb{E}[T_{bad}], \end{aligned} \tag{4.8}$$

where we have used $\mathcal{E}_{good/bad}(t)$ to denote the event that Y_t is not the all-ones string and the update from Y_t to Y_{t+1} is bad or good, respectively. (Note that once the Markov chains has reached the all-ones state, neither $\mathcal{E}_{bad}(t)$ nor $\mathcal{E}_{good}(t)$ can occur and the contribution of the corresponding term in the last sum is zero, as is desired.)

For ease of notation let $D := e^{-2\alpha c}(1-(1-\alpha)c)$. Then the above can be rewritten as

$$n - k \ge D\mathbb{E}[T] - (D - 1 + c)\mathbb{E}[T_{bad}].$$

Let C be the constant from Proposition 4.1. We may assume $C > 1/2$. For a fixed $0 < \alpha < 1$ (to be determined later) we will choose an $\varepsilon > 0$ such that $1+\varepsilon < 1/(1-\alpha)$. Then (4.8) holds for all $c \in [0, 1+\varepsilon]$, and we have $D > 0$ for any such c. Moreover, since D is a continuous function, it attains a minimum $D_{\min} = D_{\min}(\alpha) > 0$ on the compact interval $c \in [0, 1+\varepsilon]$. We may assume that $\varepsilon \le 1/3$ and $\varepsilon < D_{\min}/2$, and by Proposition 4.1, for all $0 < c \le 1+\varepsilon$,

$$\begin{aligned} n-k &\ge \left(D - \frac{C}{\log_2(1/\alpha)}(D-1+c) \right) \cdot \mathbb{E}[T] - \\ &- \frac{D-1+c}{\log_2(1/\alpha)} \log_2 \binom{n}{k}. \end{aligned}$$

Set now $\alpha = 2^{-2C} < 1/2$ and observe that then the term in front of $\mathbb{E}[T]$ is equal to $\frac{1}{2}(D+1-c) \ge \frac{1}{4}D_{\min}$; the claim of the proposition follows.

Proof. [Proof of Theorem 1.1] As $\log_2 \binom{n}{k} \le n$, the claim on the number of search points follows immediately from Proposition 4.2. For the bound on the number of steps we have to be more careful, as we also have to count the number of steps *between* updates. To do so the following observation is useful. Suppose we are in a state with exactly k ones. Then the probability that we flip exactly one zero-bit in the next step is $(n-k)\frac{c}{n}(1-\frac{c}{n})^{n-k} \ge \frac{1}{2}ce^{-c}\frac{n-k}{n}$, which holds for n sufficiently large. Note that we may assume $c < 4/3$, in which case we obtain a probability of at least $\frac{1}{10}c\frac{n-k}{n}$. As we will accept any of these moves, we thus see that the expected number of steps until the next update is at most $\frac{10}{c} \cdot n/(n-k)$.

This together with Proposition 4.2 implies a bound of $O(n^2/c)$ on the running time of the (1 + 1)-EA. To get a quasilinear bound we partition the trace of the algorithm in phases. For this, let $S_m \subseteq \{0,1\}^n$ denote the set of strings with at most $2^m - 1$ zeros, where $0 \le m \le \lfloor \log_2 n \rfloor - 1 =: m_0$.

Before reaching a state from S_{m_0} the expected time between two updates is just $O(1/c)$ (as then we still have a constant fraction of zeros to choose from). From Proposition 4.2 we thus know that the Markov chain $(X_t)_{t=0}^{\infty}$ will reach a state from S_{m_0} in $O(n/c)$ steps. Next we consider the phases in which we start in a state from S_m and terminate (the phase) when we reach a state from S_{m-1} (for the first time). Denote by T_m the number of update steps in this phase. We can use the bound from Proposition 4.2 (that considers the run of the Markov chain until it reaches the all-ones string) to obtain

$$\begin{aligned} \mathbb{E}[T_m] &\le \beta(n - (n - 2^m + 1)) + \beta \log_2 \binom{n}{n - 2^m + 1} \\ &\le 2\beta 2^m \log n. \end{aligned}$$

As we argued above, in this phase the expected number of steps between updates is bounded by $O(n/(2^m c))$ (as we always have at least 2^{m-1} zeros). The expected number of steps in this phase is thus bounded by $O(n \log n / c)$, where the hidden constant holds uniformly for all phases. Since we assumed c to be a positive constant, the expected number of steps per phase is $O(n \log n)$. As the number of phases is $O(\log n)$, the theorem follows.

A Proof that $\mathbb{E}[D \mid \mathcal{E}_{keep}] \le \mathbb{E}[D]$.

In this section we prove that $\mathbb{E}[D \mid \mathcal{E}_{keep}] \le \mathbb{E}[D]$, which is used in Section 2.2. For a one-bit j, let $\mathcal{B}_j$ denote the event that this bit is flipped into zero. We can divide all potential offspring into pairs y_0, y_1 which agree in all bits except that $y_0^j = 0$, but $y_1^j = 1$. Note that $\Pr[Y' = y_0 \mid \mathcal{B}_j] = \Pr[Y' = y_1 \mid \neg\mathcal{B}_j]$, because all bits are flipped independently. Moreover, since $f(y_0) < f(y_1)$, we have the implication "y_0 is accepted $\implies$ y_1 is accepted". Hence,

$$\begin{aligned}\Pr[\mathcal{E}_{keep} \mid \mathcal{B}_j] &= \sum_{\substack{y_0 \in \{0,1\}^n \\ y_0^j = 0}} \Pr[Y' = y_0] \cdot \mathbb{1}_{\mathcal{E}_{keep}}(y_0) \\ &\le \sum_{\substack{y_1 \in \{0,1\}^n, \\ y_1^j = 1}} \Pr[Y' = y_1] \cdot \mathbb{1}_{\mathcal{E}_{keep}}(y_1) \qquad \text{(A.1)} \\ &= \Pr[\mathcal{E}_{keep} \mid \neg\mathcal{B}_j],\end{aligned}$$

and therefore,

$$\begin{aligned}&\Pr[\mathcal{E}_{keep}] \\ &= \Pr[\mathcal{E}_{keep} \mid \mathcal{B}_j] \cdot \Pr[\mathcal{B}_j] + \Pr[\mathcal{E}_{keep} \mid \neg\mathcal{B}_j] \cdot \Pr[\neg\mathcal{B}_j] \\ &\overset{(A.1)}{\ge} \Pr[\mathcal{E}_{keep} \mid \mathcal{B}_j] \cdot \Pr[\mathcal{B}_j] + \Pr[\mathcal{E}_{keep} \mid \mathcal{B}_j] \cdot (1 - \Pr[\mathcal{B}_j]) \\ &= \Pr[\mathcal{E}_{keep} \mid \mathcal{B}_j].\end{aligned}$$

In particular, this implies

$$\Pr[\mathcal{B}_j \mid \mathcal{E}_{keep}] = \frac{\Pr[\mathcal{E}_{keep} \mid \mathcal{B}_j] \Pr[\mathcal{B}_j]}{\Pr[\mathcal{E}_{keep}]} \le \Pr[\mathcal{B}_j].$$

Since this holds for all one-bits j, and since $D = \sum_j \mathbb{1}_{\mathcal{B}_j}$, where the sum runs over all one-bits j, we obtain $\mathbb{E}[D \mid \mathcal{E}_{keep}] \le \mathbb{E}[D] \le c$.

B Missing details in the proof of Proposition 2.2

Here we show that for $c < 4/3$ and n large enough, the product $a_{u,d}b_{u,d}$ is maximized either for $u = 1$, $d = 0$, or for $u = d = 1$. As a reminder, we repeat the definitions of $a_{u,d}$ and $b_{u,d}$.

$$\begin{aligned}a_{u,d} &:= \left(\frac{c}{n}\right)^{u+d} \left(1 - \frac{c}{n}\right)^{n-u-d}. \\ b_{u,d} &:= \binom{n-k}{u}\binom{k+u}{d}\end{aligned}$$

We first observe $a_{u,d}/a_{u-1,d-1} < \frac{17}{9}\frac{1}{n^2}$ for all $u, d \ge 1$. Thus

$$\begin{aligned}\frac{b_{u,d}}{b_{u-1,d-1}} &= \frac{\binom{n-k}{u}}{\binom{n-k}{u-1}} \cdot \frac{\binom{k+u}{d}}{\binom{k+u-1}{d-1}} \\ &= \frac{n-k-u+1}{u} \cdot \frac{k+u-d+1}{d} \\ &\le \frac{n^2}{ud}\end{aligned}$$

implies that $a_{u,d}b_{u,d} \le a_{u-1,d-1}b_{u-1,d-1}$ for all $u, d \ge 2$. Similarly, since for $u \ge 1$,

$$\begin{aligned}\frac{b_{u,1}}{b_{u-1,1}} &= \frac{\binom{n-k}{u}}{\binom{n-k}{u-1}} \cdot \frac{\binom{k+u}{1}}{\binom{k+u-1}{1}} = \frac{n-k-u+1}{u} \cdot \frac{k+u}{k+u-1} \\ &\overset{k \ge d = 1}{\le} \frac{3}{2u} \cdot n\end{aligned}$$

we deduce that $a_{u,1}b_{u,1} \le a_{1,1}b_{1,1}$ for all $u \ge 2$ and from

$$\frac{b_{1,d}}{b_{1,d-1}} = \frac{\binom{k+1}{d}}{\binom{k+1}{d-1}} = \frac{k+1-d+1}{d} \le \frac{n}{d}$$

we get $a_{1,d}b_{1,d} \le a_{1,1}b_{1,1}$ for all $d \ge 2$. Together, this gives $a_{u,d}b_{u,d} \le a_{1,1}b_{1,1}$ for all $u, d \ge 1$. Finally, for $u \ge 2$ and n sufficiently large,

$$\begin{aligned}\frac{a_{u,0}b_{u,0}}{a_{u-1,0}b_{u-1,0}} &= \frac{\frac{c}{n}}{(1 - \frac{c}{n})} \cdot \frac{\binom{n-k}{u}}{\binom{n-k}{u-1}} \\ &= \frac{c}{n-c} \cdot \frac{n-k-u+1}{u} \le \frac{n}{n-c} \cdot \frac{c}{u} \le 1,\end{aligned}$$

so $a_{u,0}b_{u,0} \le a_{1,0}b_{1,0}$ for all $u \ge 2$. Altogether, we thus have $a_{u,d}b_{u,d} \le \max\{a_{1,0}b_{1,0}, a_{1,1}b_{1,1}\}$ for all $u \ge 1, d \ge 0$, as required.

References

[1] D. Achlioptas and F. Iliopoulos. Focused stochastic local search and the lovász local lemma. In *Symposium on Discrete Algorithms (SODA)*, pages 2024–2038. SIAM, 2016.

[2] D. Achlioptas and F. Iliopoulos. Random walks that find perfect objects and the lovász local lemma. *Journal of the ACM*, 63(3):22, 2016.

[3] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford university press, 1996.

[4] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation.* CRC Press, 1997.

[5] K. Bringmann and K. G. Larsen. Succinct sampling from discrete distributions. In *Symposium on Theory of Computing (STOC)*, pages 775–782. ACM, 2013.

[6] J. Brody and K. G. Larsen. Adapt or die: Polynomial lower bounds for non-adaptive dynamic data structures. *Theory of Computing*, 11(19):471–489, 2015.

[7] T. M. Cover and J. A. Thomas. *Elements of information theory.* John Wiley & Sons, 2012.

[8] B. Doerr, C. Doerr, and T. Kötzing. Personal communication.

[9] B. Doerr, T. Jansen, D. Sudholt, C. Winzen, and C. Zarges. Optimizing monotone functions can be difficult. In *Parallel Problem Solving from Nature (PPSN)*, 2010.

[10] 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.

[11] V. Dujmović, G. Joret, J. Kozik, and D. R. Wood. Nonrepetitive colouring via entropy compression. *Combinatorica*, 36(6):661–686, 2016.

[12] L. Esperet and A. Parreau. Acyclic edge-coloring using entropy compression. *European Journal of Combinatorics*, 34(6):1019–1027, 2013.

[13] L. Fortnow. A Kolmogorov Complexity Proof of the Lovász Local Lemma. Blogpost, 2009. `https://blog.computationalcomplexity.org/2009/06/kolmogorov-complexity-proof-of-lov.html`.

[14] J. Grytczuk, J. Kozik, and P. Micek. New approach to nonrepetitive sequences. *Random Structures & Algorithms*, 42(2):214–225, 2013.

[15] T. Jansen. On the brittleness of evolutionary algorithms. In *Foundations of Genetic Algorithms (FOGA)*, pages 54–69. Springer, 2007.

[16] K. G. Larsen. The cell probe complexity of dynamic range counting. In *Symposium on Theory of Computing (STOC)*, pages 85–94. ACM, 2012.

[17] J. Lengler. A general dichotomy of evolutionary algorithms on monotone functions. In *Parallel Problem Solving from Nature (PPSN), full version at arXiv:1803.09227*, 2018.

[18] J. Lengler and A. Steger. Drift Analysis and Evolutionary Algorithms Revisited. *Combinatorics, Probability and Computing*, 27(4):643–666, 2018.

[19] R. Moser. A constructive proof of the lovász local lemma. In *Symposium on Theory of Computing (STOC)*, pages 343–350, 2009.

[20] R. Moser and G. Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM*, 57(2):11, 2010.

[21] J. Przybyło, J. Schreyer, and E. Škrabuláková. On the facial thue choice number of plane graphs via entropy compression method. *Graphs and Combinatorics*, 32(3):1137–1153, 2016.

[22] T. Tao. Moser's entropy compression argument. Blogpost, 2009. `https://terrytao.wordpress.com/2009/08/05/mosers-entropy-compression-argument/`.

[23] C. Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability and Computing*, 22(2):294–318, 2013.