Runtime Analysis with Variable Cost

Per Kristian Lehre
p.k.lehre@bham.ac.uk
University of Birmingham
United Kingdom

Andrew M. Sutton
amsutton@d.umn.edu
University of Minnesota Duluth
USA

ABSTRACT

The usual approach in runtime analysis is to derive estimates on the number of fitness function evaluations required by a method until a suitable element of the search space is found. One justification for this is that in real applications, fitness evaluation often contributes the most computational effort. A tacit assumption in this approach is that this effort is uniform and static across the search space. However, this assumption often does not hold in practice: some candidates may be far more expensive to evaluate than others. This might occur, for example, when fitness evaluation requires running a simulation or training a machine learning model.

Despite the availability of a wide range of benchmark functions coupled with various runtime performance guarantees, the runtime analysis community currently lacks a solid perspective of handling variable fitness cost. Our goal with this paper is to argue for incorporating this perspective into our theoretical toolbox. We introduce two models of handling variable cost: a simple non-adaptive model together with a more general adaptive model. We prove cost bounds in these scenarios and discuss the implications for taking into account costly regions in the search space.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**.

KEYWORDS

runtime analysis, variable cost model, adaptive strategies

ACM Reference Format:
Per Kristian Lehre and Andrew M. Sutton. 2023. Runtime Analysis with Variable Cost. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3583131.3590432

1 INTRODUCTION

A typical assumption in the computational complexity of evolutionary algorithms is that the process of evaluating the fitness function supplies the greatest contribution to the runtime, but that this cost is uniform over all search points. However, in many real world applications the cost of evaluating the fitness of a candidate solution can depend directly on its structure, and therefore varies throughout the search process. For example, this can occur in structural design applications, protein structure prediction [6], data streaming, robot exploration [9], or tuning machine learning models [10, 12]. In these situations, it has been observed that fitness function evaluations may only have a high cost in certain parts of the search space [10].

So far, settings in which the cost of fitness evaluation varies over the space have not been addressed by the runtime analysis community. With this paper, we provide a first step in this direction by introducing and analyzing variable cost models.

We first introduce a simple *non-adaptive cost model* in which an algorithm must optimize a function with both hidden cost and fitness variables. We show that a simple strategy of fitness tie-breaking to favor points that were cheaper at evaluation time can lead to asymptotically faster runtimes. We then consider a more refined *adaptive cost model* in which the algorithm may choose to abandon the evaluation of search point if it decides that it has spend too much time on it. We present a cost-adaptive fitness level theorem to provide general bounds on optimization costs in this setting. We also consider a simple benchmark function equipped with a costly region and prove runtime bounds on a cost-adaptive algorithm on this function.

The remainder of the paper is organized as follows. In the next section we review related work and introduce some technical preliminaries. We then introduce the non-adaptive cost model in Section 3 and study several scenarios. In Section 4 we introduce the non-adaptive cost model and the Cost-adaptive Fitness Level Theorem. We conclude the paper in Section 5.

2 BACKGROUND

2.1 Related Work

Costly fitness functions are often handled in practice by so-called surrogate models or meta-models [6, 10, 11]. These are some kind of approximation of the true fitness function using machine learning or statistical models. Most work in this area is focused on continuous search spaces, but some strategies also exist for surrogate models in discrete domains [1].

In the continuous setting, the field of Bayesian optimization deals with optimizing expensive functions by assuming the objective function is drawn from a stochastic process. Instead of optimizing the expensive function directly, the strategy is to only query the objective function periodically to place a prior over the distribution, and then maximize a cheaper acquisition function to propose the next query. This process repeats until a budget is exhausted [9]. Recent work in this area has focused on acquisition functions that prefer cheaper function points to query in order to reduce the overall computational cost [9, 12].

Our approach differs significantly from the case of surrogate modeling in that we are still interested in directly optimizing the

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590432

fitness function, but try to take advantage of the fact that some regions of the space may be cheaper to move through, and thus it becomes sensible to balance the evaluation cost with the progress of the search algorithm.

Mengshoel et al. [10] consider the trade-off between exploration and exploitation in stochastic local search (SLS) algorithms applied to computationally costly fitness functions. In particular, they point out that SLS strategies like WalkSAT often employ a noise step with some probability (selecting a random neighbor, regardless of fitness), otherwise they perform a greedy step by choosing the fittest element of the neighborhood. The noise step is comparatively cheap, as it requires only a single fitness evaluation, whereas the greedy step requires the evaluation of the entire Hamming neighborhood. The authors model the SLS process as a Markov chain and empirically analyze the hitting time (incorporating cost) as a function of noise step probability.

Jansen and Zarges [5] cautioned against over-simplified views of counting function evaluations and demonstrated that different ways of accounting for the cost of algorithm modules can lead to contradictory results. This underscores the importance of understanding how to deal with evaluation cost properly when conducting theoretical research.

2.2 Preliminaries

Consider any finite set $\mathcal{X}$ which we call the "search space" and any function $f : \mathcal{X} \to \mathbb{R}$, which we will call the fitness function. We consider a scenario where evaluating the f-value of any search point x is associated with a "cost" $g(x)$ given by a cost function $g : \mathcal{X} \to \mathbb{N}$.

Given f and g, we define the *cost* of an algorithm optimizing f as the total cost of every fitness evaluation required to locate the optimum. The classical idea of runtime is recovered by setting $g(x) = 1$ for all $x \in \mathcal{X}$.

For $n \in \mathbb{N}$, let $[n] := \{1, 2, \cdots, n\}$.

The following notation will be used in Section 4. For any finite set $\mathcal{X}$, assume that $(A_1, \ldots, A_m)$ is a partition of $\mathcal{X}$. Then, for all $j \in [m]$ we write $A_{\geq j} := \cup_{i=j}^{m} A_j$. Assume furthermore that for each $j \in [m]$, $(A_{j0}, \ldots, A_{jm_j})$ is a partition of A_j for some $m_j \geq 0$. Then, for all $j \in [m]$ and $i \in [m_j]$, we write $A_{\geq ji} := \left(\cup_{k=i}^{m_j} A_{jk}\right) \cup A_{\geq j+1}$. For any function $f : \mathcal{X} \to \mathbb{R}$, we say that a partition is f-*based* if for all $x, y \in \mathcal{X}$, $f(x) < f(y)$ if and only if there exist $i < j$ such that $x \in A_i$ and $y \in A_j$.

In Section 3 we will make use of the following two important drift theorems, which we restate here for completeness.

Theorem 1 (Witt [13]). *Consider an algorithm $\mathcal{A}$ maximizing a function f and a partition of the search space into nonempty sets $A_1, \ldots, A_m$ where for all $x \in A_i$, $y \in A_j$ with $1 \leq i < j \leq m$, it holds that $f(x) < f(y)$. If p_i is a lower bound on the probability that a step of $\mathcal{A}$ escapes fitness level A_i, independently of previous steps, then for any $\delta > 0$ the first hitting time of A_m is at most*

$$\sum_{i=1}^{m-1} \frac{1}{p_i} + \delta$$

with probability at least $1 - e^{-\frac{\delta}{4} \cdot \min\{\frac{\delta}{s}, h\}}$, for any finite $s \geq \sum_{i=1}^{m-1} \frac{1}{p_i^2}$ and $h = \min\{p_i \mid i = 1, \ldots, m-1\}$.

Theorem 2 (Lengler and Spooner [8]). *Let $(X_t)_{t \geq 0}$ be a stochastic process, and let $b \in \mathbb{N}$ and $0 < \delta < 1$. If for all $t \leq b$, $\mathrm{E}[X_t - X_{t+1} \mid X_t = x] \geq \delta x$, then $E[X_b \mid X_0] \leq X_0(1-\delta)^b \leq X_0 e^{-\delta b}$. If for all $t \leq b$, $\mathrm{E}[X_t - X_{t+1} \mid X_t = x] \leq \delta x$, then $E[X_b \mid X_0] \geq X_0(1-\delta)^b \geq X_0 e^{-2\delta b}$ for $\delta \leq 0.797$.*

3 NON-ADAPTIVE COST MODEL

For our first model, we assume that the fitness function f depends on a subset $\mathcal{I}_f \subseteq [n]$ decision variables, and that the cost function g depends on a subset $\mathcal{I}_g \subseteq [n]$ decision variables. Moreover, we assume that $\mathcal{I}_f$ and $\mathcal{I}_g$ are hidden variables in the sense that they are not known to the algorithm. The fitness evaluation component is thus somewhat similar to the setting of *unknown problem lengths* in which only a subset of bits are fitness relevant [2–4].

In their work on stochastic local search, Mengshoel et al. [10] consider three variants of the cost function, linear, quadratic and exponential. They point out that experiments suggest a *linear* cost model $g(x) = \alpha|x| + \beta$ is a reasonable approximate for various applications. This also seems to be the case in the continuous setting, e.g., Luong et al.[9] also use a linear cost function and point out that cost is often assumed to be linear in real applications.

Thus, for this section, we employ a linear cost function with $\alpha = \beta = 1$, i.e.,

$$g(x) = 1 + \sum_{i \in \mathcal{I}_g} x_i.$$

We point out, however, that the runtime results we present in this section are easily generalized to arbitrary α and β parameters.

We first consider the classical (1+1) EA (Algorithm 1). Note that when fitness evaluation is a bottleneck, it is advisable to store the result to eliminate unnecessary function calls [5].

Algorithm 1: (1+1) EA

```
Choose x from {0,1}^n uniformly at random
f_x ← value of f(x)
for t = 0, 1, 2, ... until termination condition met do
    Obtain y by flipping each bit in x with probability 1/n.
    f_y ← value of f(y)
    if f_y ≥ f_x then
        x ← y
        f_x ← f_y
    end if
end for
```

We also introduce a slight variation to the (1+1) EA that, all else being equal, prefers points that are cheaper to evaluate. This is similar to some strategies in continuous optimization of costly black-box functions in which both the function value and the function query cost are minimized in parallel [12]. This variant, called the Cost-Aware (1+1) EA, is illustrated in Algorithm 2. Essentially, it is identical to Algorithm 1, except that is breaks fitness ties by taking the solution that was cheaper to compute.

We consider the stochastic process $(Y_t)_{t \in \mathbb{N}}$ where Y_t is the random variable corresponding to the cost $g(y)$ to compute $f(y)$ in line 5 on iteration t of Algorithm 2.

Algorithm 2: Cost-Aware (1+1) EA

Choose x from $\{0,1\}^n$ uniformly at random
$(f_x, g_x) \leftarrow$ value of $f(x)$, cost required to compute it
for $t = 0, 1, 2, \ldots$ until termination condition met **do**
  Obtain y by flipping each bit in x with probability $1/n$.
  $(f_y, g_y) \leftarrow$ value of $f(y)$, cost required to compute it
  if $f_y > f_x$ **or** $\left(f_y = f_x \text{ and } g_y < g_x\right)$ **then**
    $x \leftarrow y$
    $f_x \leftarrow f_y$
  end if
end for

The following lemma supplies a general upper and lower bound to the drift of the random cost variables during the optimization of Algorithm 2.

Lemma 1. *Suppose $\mathcal{I}_f \cap \mathcal{I}_g = \emptyset$ and denote by $\mathcal{E}_t^>$ the event that the offspring produced in generation t has strictly greater fitness. Denote by $\mathcal{E}_t^=$ the event that the fitness of the offspring generated in iteration t is equal in fitness. Then the drift in cost for Algorithm 2 is bounded as follows.*

$$\begin{aligned}\mathrm{E}\left[Y_t - Y_{t+1} \mid Y_t = s\right] &\geq \frac{\Pr(\mathcal{E}_t^>)}{n}(2s - |\mathcal{I}_g|) \\ &\quad + \frac{s\Pr(\mathcal{E}_t^=)}{n}\left(1-\frac{1}{n}\right)^{|\mathcal{I}_g|-1}, \\ \mathrm{E}\left[Y_t - Y_{t+1} \mid Y_t = s\right] &\leq \frac{\Pr(\mathcal{E}_t^>)}{n}(2s - |\mathcal{I}_g|) + \frac{s}{n}.\end{aligned}$$

Proof. Under the event $\mathcal{E}_t^>$, the offspring is accepted but since $\mathcal{I}_f \cap \mathcal{I}_g = \emptyset$, selection is independent of the cost variables. Each cost variable flips with exactly probability $1/n$, which yields

$$\mathrm{E}\left[Y_t - Y_{t+1} \mid Y_t = s \cap \mathcal{E}_t^>\right] = \frac{s}{n} - \frac{(|\mathcal{I}_g| - s)}{n}.$$

Under $\mathcal{E}_t^=$, the offspring is only accepted if its cost is reduced, which yields the conditional drift

$$\begin{aligned}\mathrm{E}\left[Y_t - Y_{t+1} \mid Y_t = s \cap \mathcal{E}_t^=\right] &\geq \sum_{i \in \mathcal{I}_g : x_i = 1} \frac{1}{n}\left(1-\frac{1}{n}\right)^{|\mathcal{I}_g|-1} \\ &= \frac{s}{n}\left(1-\frac{1}{n}\right)^{|\mathcal{I}_g|-1}, \\ \mathrm{E}\left[Y_t - Y_{t+1} \mid Y_t = s \cap \mathcal{E}_t^=\right] &\leq \sum_{i \in \mathcal{I}_g : x_i = 1} \frac{1}{n} = \frac{s}{n}.\end{aligned}$$

The bounds follow from the law of total probability. □

3.1 LeadingOnes

To begin the analysis we make the simplifying assumption that $\mathcal{I}_f \cap \mathcal{I}_g = \emptyset$, i.e., the fitness and cost variables are decoupled, which is required by Lemma 1. We also assume that $|\mathcal{I}_f| = |\mathcal{I}_g| = n/2$, but point out that our results hold as long as both variable sets contain $\Theta(n)$ positions each. We consider the well-known LeadingOnes benchmark function:

$$f(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_i.$$

Since the standard (1+1) EA simply ignores evaluation cost, every function evaluation costs $\Theta(n)$ in expectation, and we have the following.

Theorem 3. *Let f be the LeadingOnes function on the variables in $\mathcal{I}_f$ with $|\mathcal{I}_f| = |\mathcal{I}_g| = n/2$ with $\mathcal{I}_f \cap \mathcal{I}_g = \emptyset$. Then the optimization cost of the (1+1) EA (Algorithm 1) is $\Theta(n^3)$.*

The strategy of preferring neutral mutations that were cheaper to evaluate leads to a linear factor improvement, as we prove in the following theorem.

Theorem 4. *Let f be the LeadingOnes function on the variables in $\mathcal{I}_f$ with $|\mathcal{I}_f| = |\mathcal{I}_g| = n/2$ with $\mathcal{I}_f \cap \mathcal{I}_g = \emptyset$. Then the optimization cost of the Cost-Aware (1+1) EA (Algorithm 2) is $\Theta(n^2)$.*

Proof. In each iteration t, a strictly improving LeadingOnes offspring y can be created from x if the leading zero in $\mathcal{I}_f$ is flipped, while none of the $f(x)$ leading ones in $\mathcal{I}_f$ are changed. This occurs with probability $\Pr(\mathcal{E}_t^>) = \frac{1}{n}\left(1-\frac{1}{n}\right)^{f(x)}$. Similarly, the event $\mathcal{E}_t^=$ occurs when none of the $f(x)$ leading ones nor the leading zero in $\mathcal{I}_f$ are changed. This event occurs with probability $\left(1-\frac{1}{n}\right)^{f(x)+1}$. Recall that Y_t denotes the random variable that measures the cost of computing the fitness of the offspring generated in line 5 of Algorithm 2. By Lemma 1, the cost drift is bounded as

$$\begin{aligned}\mathrm{E}[Y_t - Y_{t+1} \mid Y_t = s] &\geq \left(1-\frac{1}{n}\right)^{f(x)} \frac{\left(2s - \frac{n}{2}\right)}{n^2} \\ &\quad + \left(1-\frac{1}{n}\right)^{f(x)+1}\left(1-\frac{1}{n}\right)^{\frac{n}{2}-1}\frac{s}{n} \\ &\geq \frac{s}{e}\left(\frac{1}{n} + \frac{2}{n^2}\right) - \frac{1}{2en} \\ &\geq \frac{s}{e}\left(\frac{1}{n} + \frac{2}{n^2}\right) - \frac{s}{2en} \geq \frac{s}{2en}.\end{aligned}$$

Let T be the first iteration where Algorithm 2 finds an optimal solution maximizing f. By Theorem 1, setting $p_i = 1/(en) \leq \Pr(\mathcal{E}_t^>)$, for any constant $0 < \epsilon < 1$, we have

$$\Pr\left(T \leq \sum_{i=0}^{n-1} en + \epsilon en^2 \leq (1+\epsilon)en^2\right) \geq 1 - e^{-\epsilon^2 n/4}.$$

Conditioning on the event $\{T < (1+\epsilon)en^2\}$, we apply Theorem 2 to obtain the following upper bound on the total expected cost.

$$\begin{aligned}\mathrm{E}\left[\sum_{t=1}^{(1+\epsilon)en^2} Y_t \,\middle|\, Y_0\right] &\leq Y_0 \sum_{t=1}^{(1+\epsilon)en^2} e^{-t/(2en)} \\ &\leq Y_0 \int_0^{(1+\epsilon)en^2} e^{-t/(2en)}\,dt \\ &\leq en^2(1-o(1)),\end{aligned}$$

since $Y_0 \leq |\mathcal{I}_g| = n/2$. Furthermore, since $\Pr(T > (1+\epsilon)en^2) \leq e^{-\Omega(n)}$ and the maximum cost is $g(x) \leq n$, we have that for every polynomial T, the contribution to the expectation vanishes exponentially fast. Applying the law of total expectation, the upper bound on the total cost is $O(n^2)$. The lower bound trivially follows from the well-known expected runtime bound on the standard (1+1) EA on LeadingOnes together with the fact that $g(x) \geq 1$. □

3.2 OneMax

The success of Algorithm 2 on LeadingOnes is due to the fact that it has the side effect of making significant progress minimizing the cost function while waiting for an improvement in the fitness function. This generous waiting time afforded by LeadingOnes does not arise with OneMax, and the case here is not as clear.

Theorem 5. *Let f be the OneMax function on the variables in $\mathcal{I}_f$ with $|\mathcal{I}_f| = |\mathcal{I}_g| = n/2$ with $\mathcal{I}_f \cap \mathcal{I}_g = \emptyset$. Then for $0 < \epsilon < 1$, the optimization cost of the Cost-Aware (1+1) EA (Algorithm 2) is $\Omega(n^2) \cap O(n^2 \log n)$.*

Proof. Let T^* be the runtime of the Cost-Aware (1+1) EA. Since the tie breaking rule of the Cost-Aware (1+1) EA is independent of the fitness variables, T^* is distributed identically to the (1+1) EA. Applying a lower tail bound to the multiplicative drift [7, Theorem 8], $\Pr(T^* \geq \frac{en}{4} \ln n - \frac{cn}{2}) = 1 - o(1)$ for a constant $c > 0$. We condition on this event for the remainder of the proof.

By Lemma 1,

$$\begin{aligned}\mathrm{E}[Y_t - Y_{t+1} \mid Y_t = s] &\leq \frac{\Pr(\mathcal{E}_t^>)}{n}\left(2s - \frac{n}{2}\right) + \frac{s}{n} \\ &= \frac{s}{n}\left(2\Pr(\mathcal{E}_t^>) + 1\right) - \frac{\Pr(\mathcal{E}_t^>)}{2} \leq \frac{3s}{n}.\end{aligned}$$

By Theorem 2, we have $\mathrm{E}[Y_t] = \mathrm{E}[\mathrm{E}[Y_t \mid Y_0]] \geq \mathrm{E}[Y_0]e^{-6t/n}$. It is therefore possible to bound the total optimization cost as

$$\begin{aligned}\sum_{t=1}^{T^*} \mathrm{E}[Y_t] &\geq \mathrm{E}[Y_0] \int_1^{T^*+1} e^{-6t/n} dt \\ &= \mathrm{E}[Y_0]\left(\frac{n}{6}\right) \cdot \left(e^{-6/n} - e^{-6(T^*+1)/n}\right) = \Omega(n^2),\end{aligned}$$

since $\mathrm{E}[Y_0] = n/4$ and we have conditioned on $T^* \geq \frac{en}{4} \ln n - \frac{cn}{2}$.

The asymptotic upper bound on the optimization cost follows immediately from the fact $Y_t \leq n/2$ for all $t \geq 0$ and the $O(n \log n)$ runtime of the standard (1+1) EA. □

We point out that the quadratic lower bound in Theorem 5 arises only from the trivial upper bound $\Pr(\mathcal{E}_t^>) \leq 1$, and therefore is possibly too weak. Nevertheless, we conjecture that the Cost-Aware (1+1) EA has an advantage over the standard (1+1) EA in this setting, and that the quadratic bound is closer to the truth. Empirical support for this conjecture is provided by Figure 1, which plots the mean optimization time over 100 runs each for $n = 100, 200, \ldots, 1000$ of both Algorithms 1 and 2.

3.3 Shared fitness and cost variables

We now consider a case where the cost and fitness variables overlap. In particular, for $k = \omega(\log n)$, the fitness function is LeadingOnes

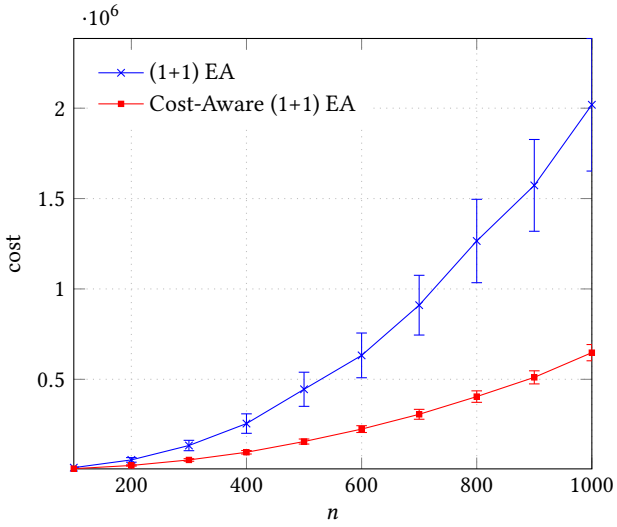

Figure 1: Mean optimization cost of the (1+1) EA and the Cost-Aware (1+1) EA on OneMax with $|\mathcal{I}_f| = |\mathcal{I}_g| = n/2$. Error bars denote standard deviation over 100 trials for each n.

on the first k bits. In this setting, we have $[k] = \mathcal{I}_f \subseteq \mathcal{I}_g = [n]$,

$$f(x) = f_k(x) = \sum_{i=1}^{k} \prod_{j=1}^{i} x_i.$$

Theorem 6. *Let $k = \omega(\log n)$. The expected cost of the standard (1+1) EA (Algorithm 1) on k-LeadingOnes and shared linear cost is $\Theta(kn^2)$.*

Proof. For the upper bound, note that the cost of each evaluation is at most n, and the expected number of evaluations until a strictly better offspring is generated is $\Theta(n)$. The function is optimized after at most k improvements, so the runtime is at $O(kn^2)$.

For the lower bound, let $T' := \min\{t : f(x) \geq k/2\}$, which cannot be larger than the runtime. Let $(Y_t)_{t\geq 0}$ be the stochastic process that corresponds to the contribution to the cost at time t from the trailing $n - k/2$ bits. Clearly, if y is the offspring generated in time t, we would have $g(y) \geq Y_t$. Moreover, since none of the $n - k/2$ trailing bits are under selection before time T', the Y_t variables may be regarded as i.i.d., and thus we may apply Wald's equation to bound the cost until T':

$$\mathrm{E}\left[\sum_{t=0}^{T'} Y_t\right] = \mathrm{E}[T']\,\mathrm{E}[Y_0] = \Omega(kn^2).$$

The asymptotic bound follows from $\mathrm{E}[T'] = \Omega(kn)$ and $\mathrm{E}[Y_0] = (n - k/2)/2 = \Omega(n)$ since $k \leq n$. □

Theorem 7. *Let $k = \omega(\log n)$. The expected cost of the Cost-Aware (1+1) EA (Algorithm 2) on k-LeadingOnes and shared linear cost is $O(n^2 + k^2 n)$.*

Proof. We decompose the cost function as $g(x) = g_1(x) + g_2(x)$ where

$$g_1(x) = \sum_{i=1}^{k} x_i, \quad \text{and,} \quad g_2(x) = \sum_{i=k+1}^{n} x_i,$$

and let $(Y_t)_{t\geq 0}$ denote the sequence of random variables corresponding to the "trailing bits" cost g_2 of the offspring generated in iteration t. Let T be the random variable corresponding to the runtime. Conditioning on the event $\{T \leq ckn\}$ for a positive constant $c > 0$, we bound the conditional expected contribution of the cost from g_2 as

$$\mathrm{E}\left[\sum_{t=1}^{T} Y_t\right] \leq \sum_{t=1}^{ckn} \mathrm{E}[Y_t] \leq n \sum_{t=1}^{ckn} e^{-t/(2en)} = O(n^2),$$

where the sum can again be bounded by a definite integral similar to the proof of Theorem 4. We point out that Theorem 4 requires $\mathcal{I}_f$ and $\mathcal{I}_g$ to be disjoint, which does not hold in this setting. However, note that we only bound here the cost contribution from g_2, which comes from a proper subset of $\mathcal{I}_g$, and this subset is in fact disjoint from $\mathcal{I}_f$.

Since $0 \leq g_1(x) \leq k$ for any $x \in \{0,1\}^n$, the contribution from g_1 can be trivially bounded as $kT = O(k^2 n)$.

We now argue that $\Pr(T > ckn)$ approaches zero superpolynomially fast, and thus the contribution to the total expected cost conditioned on this event vanishes. For this, we appeal to Theorem 1, since T is the hitting time of fitness level k. Each fitness level has an escape probability of at most $1/(en)$, so we may choose $\delta = ckn$, $s = ke^2n^2$ and $h = (en)^{-1}$, thereby bounding $\Pr(T \leq ckn) \geq 1 - e^{-\Omega(k)}$. Since we require $k = \omega(\log n)$, the proof is complete. □

4 ADAPTIVE COST MODEL

In the non-adaptive cost model described above, the algorithm cannot abort the evaluation of a search point once evaluation has been initiated. Hence, if evaluating the wrong search points, the algorithm may suffer exceedingly high evaluation costs. This is an unrealistic assumption in some practical settings. Suppose for example that fitness values are obtained through a simulation. It is reasonable to assume that the algorithm can abort the simulation if it takes too long.

We will introduce a more refined cost model that captures this scenario. We assume that the oracle accepts queries composed of an evaluation budget c and a search point x. More formally, assuming fitness function $f : \mathcal{X} \to \mathbb{R}$ and cost function $g : \mathcal{X} \to \mathbb{N}$, the oracle responds to the query $(c, x) \in \mathbb{N} \times \mathcal{X}$ with the value

$$r(c,x) := \begin{cases} f(x) & \text{if } g(x) \leq c, \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

The special response $\bot$ signifies that the evaluation time for search point x exceeded the requested budget c and therefore that the evaluation was aborted before the fitness value $f(x)$ was obtained.

A black-box optimization algorithm in this setting now works as follows. In each iteration $t \in \mathbb{N}$, using a "history" I^{t-1} of the past $t-1$ queries,

$$I^{t-1} := \left\{(1, x^1, c^1, r(c^1, x^1)), \ldots, (t-1, x^{t-1}, c^{t-1}, r(c^{t-1}, x^{t-1}))\right\},$$

algorithm A makes an oracle-query composed of a pair $(c^t, x^t) \in \mathbb{N} \times \mathcal{X}$, and obtains the oracle response $r(c^t, x^t)$. The cost-adaptive runtime of algorithm A on fitness function f with cost function g is defined as

$$T_{A,f,g} := \sum_{t=1}^{\tau_{A,f}} \min\{c^t, g(x^t)\}$$

where

$$\tau_{A,f} := \min\{t \in \mathbb{N} \mid g(x^t) \leq c^t \text{ and } f(x^t) \geq f(y) \text{ for all } y \in \mathcal{X}\}.$$

We obtain the classical black-box scenario for the constant cost function $g(x) = 1$ for all $x \in \mathcal{X}$.

Note that the algorithm cannot accumulate evaluation time in the cost-adaptive model through repetitive evaluation of a search point. In particular, if the evaluation at time t of a search point y is aborted after c^t time, then no information is stored about the fitness evaluation. Hence, if the algorithm attempts to evaluate the same search point y at some later time $t' > t$, the algorithm obtains $f(y)$ if and only if $c^{t'} \geq g(y)$.

4.1 A cost-adaptive algorithm

We now design an algorithm which adapts evaluation budgets to the cost of solutions. A challenging aspect is that the algorithm has no *a priori* information about the cost values. To make the problem more tractable, we will assume that the search space is composed of a cheap region where the cost equals $g(x) = 1$, and an expensive region where the cost is $g(x) > 1$. Furthermore, we assume that the probability that a uniformly sampled search point is cheap is relatively high.

Based on these assumptions, the idea behind Algorithm 3 is to maintain two search points, a search point x which is constrained to the cheap region, and a search point y which is free to explore the entire search space. Based on these constraints, new candidate search points are produced via standard bitwise mutation, akin to the (1+1) EA. The challenge is to distribute the evaluation budget between the two search points x and y. The evaluation budget is adapted using variable b_t.

In line 2-7, the algorithm assumes that two "cheap" initial search points x and y with cost 1 can be found by sampling uniformly at random from the search space. We leave open to future work how to design initialization mechanisms for problems when this assumption is not met.

We call the search point y' obtained from search point y in line 13 an "offspring" of search point y. Similarly, the search point x' obtained in line 23 is an "offspring" of search point x.

During one iteration of lines 8–27 in Algorithm 3, the value of t increases by 1. We call such an iteration an *outer iteration*. The start of the outer loop (line 14–20) ensures that the following two invariants are satisfied: search point x is the fittest search point discovered so far with cost 1 (lines 14–16), and search point y is the fittest among all search points observed by the algorithm.

In any outer iteration t, the algorithm spends a cost of at most $2b_t$ corresponding to evaluate one offspring of search point y, first with budget 1 then with budget $b_t - 1$. Then, the algorithm evaluates b_t offspring of search point x with budget 1 each. Hence, both search points are updated with the same amount of evaluation budget in each outer loop.

Algorithm 3: (2+2) Cost-balancing Evolutionary Algorithm (CBEA)

Initialize $x, y \sim \text{Unif}(\{0,1\}^n)$ and $b_0 \leftarrow 1$
while $r(1,x) = \perp$ **do**
  $x \sim \text{Unif}(\{0,1\}^n)$
end while
while $r(1,y) = \perp$ **do**
  $y \sim \text{Unif}(\{0,1\}^n)$
end while
for $t = 0, 1, 2, \ldots$ until termination condition met **do**
  if $f(x) \geq f(y)$ **then**
    $y \leftarrow x$.
    $b_t \leftarrow 1$
  end if
  Obtain y' by flipping each bit in y with probability $1/n$.
  if $r(1,y') \neq \perp$ and $f(y') \geq f(y)$ **then**
    $y \leftarrow y'$ and $x \leftarrow y'$
    $b_t \leftarrow 1$
  else if $r(b_t - 1, y') \neq \perp$ and $f(y') \geq f(y)$ **then**
    $y \leftarrow y'$
    $b_{t+1} \leftarrow 1$
  else
    $b_{t+1} \leftarrow 2b_t$
    for $t' = 1, \ldots, b_t$ **do**
      Obtain x' by flipping each bit in x with probability $1/n$
      if $r(1,x') \neq \perp$ and $f(x') \geq f(x)$ **then**
        $x \leftarrow x'$
        $b_{t+1} \leftarrow 1$
      end if
    end for
  end if
end for

Theorem 8 (Cost-adaptive Fitness Level Theorem). *Assume an f-based partition $(A_1, \ldots, A_m)$, and for each A_j, an f-based sub-partition $(A_{j,0}, \ldots, A_{j,m_j})$ such that for all $j \in [m]$,*

0. *The initial search points have cost $g(x) = 1$ and $g(y) = 1$.*
1. *All search points $x \in \mathcal{X}$, have cost $g(x) \leq c_{\max}$.*
2. *All search points $x \in A_{j,0}$ have cost $g(x) = 1$.*
3. $\Pr\left(\text{mut}(x) \in A_{\geq j,i+1}\right) \geq s_{j,i}$ *for all $x \in A_{j,i}$.*
4. $\Pr\left(\text{mut}(x) \in \cup_{k=j+1}^{m} A_{k,0}\right) \geq p_j$ *for all $x \in A_{j,0}$.*
5. $1 + \sum_{i=0}^{m_j} \frac{1}{s_{j,i}} \leq \frac{1}{4c_{\max}\Pr(\mathcal{F}_j)p_j}$, *where event $\mathcal{F}_j$ occurs if $y \in A_{\geq j,1}$ before $x \in A_{\geq 0,j}$.*

then Algorithm 3 obtains an element in A_m with expected cost at most

$$4\sum_{j=1}^{m-1} \min\left\{\frac{1}{p_j}, c_{\max}\left(1 + \sum_{i=0}^{m_j-1}\frac{1}{s_{j,i}}\right)\right\}.$$

Proof. Following the proof of the classical fitness-based partition theorem, we split the runtime cost into $m-1$ phases, where T_j is the cost accumulated to obtain $x, y \in A_{\geq j+1,0}$, assuming that the algorithm starts with $y \in A_{\geq j,1}$. We say that phase j starts with failure event $\mathcal{F}_j$ if $x \notin A_{\geq j,0}$ and $y \in A_{\geq j,1}$. In the case of a failure, we assume $x \notin A_{\geq j,0}$ holds until the end of the phase.

For each sub-level $A_{j,i}$, $i \in [0..m_j]$, we let $T_{j,i}$ denote the accumulated cost from having a search point $y \in A_{j,i}$ until $y \in A_{\geq j,i+1}$. We divide this time into two sub-phases, first until $b_t \geq c_{j,i}$, and second until $y \in A_{\geq j,i+1}$. After at most $\log(c_{j,i}) + 1$ outer iterations, the computation budget satisfies $b_t \geq c_{j,i}$ or $y \in A_{\geq j,i+1}$. Hence, by condition 1, the expected cost of the first sub-phase, assuming failure, is at most

$$\sum_{i=0}^{\log(c_{j,i})+1} 2 \cdot 2^i = 2^{\log(c_{j,i})+2} - 1 < 4c_{\max}.$$

During the second sub-phase, in the worst case, the algorithm will increase b_t until $y \in A_{\geq j,i+1}$. By conditions 1 and 4, the expected cost will be at most

$$2c_{\max}/s_{j,i}.$$

Hence, the expected cost until $y \in A_{\geq j+1}$ is at most

$$\mathrm{E}\left[T_j \mid \mathcal{F}_j\right] \leq \sum_{i=0}^{m_j} \mathrm{E}\left[T_{i,j}\right] \leq 4c_{\max}\left(1 + \sum_{i=0}^{m_j}\frac{1}{s_{j,i}}\right).$$

We now consider the duration of the phase, assuming no failure, i.e., we start the phase with $x \in A_{j,0}$. Within a cost of b, the algorithm evaluates $b/2$ offspring from x. Hence, the expected cost until $x \in \cup_{k=j+1}^{m} A_{k,0}$ is at most $2/p_j$. The duration of the phase is at most

$$\mathrm{E}\left[T_j \mid \overline{\mathcal{F}_j}\right] \leq 4\min\left\{\frac{1}{p_j}, c_{\max}\left(1 + \sum_{i=0}^{m_j}\frac{1}{s_{j,i}}\right)\right\}.$$

Taking into account the failure probabilities (condition 5), the unconditional expected runtime is

$$\begin{aligned}\mathrm{E}[T] &\leq \sum_{j=1}^{m-1} \Pr\left(\mathcal{F}_j\right)\mathrm{E}\left[T_j \mid \mathcal{F}_j\right] + \Pr\left(\overline{\mathcal{F}}_j\right)\mathrm{E}\left[T_j \mid \overline{\mathcal{F}_j}\right] \\ &\leq 4\sum_{j=1}^{m-1} \min\left\{\frac{1}{p_j}, c_{\max}\left(1 + \sum_{i=0}^{m_j}\frac{1}{s_{j,i}}\right)\right\}.\end{aligned}$$

□

For two parameters $k, c_{\max} \in \mathbb{N}$, we consider the variable cost problem $\textsc{JumpCost}_{k,c_{\max}}$ which has fitness function

$$f(x) = \textsc{OneMax}(x) = \sum_{i=1}^{n} x_i$$

and cost function

$$g(x) = \begin{cases} c_{\max} & \text{if } \textsc{OneMax}(x) \in [n-k+1..n-1] \\ 1 & \text{otherwise.} \end{cases}$$

This function is illustrated in Figure 2.

Theorem 9. *Assuming that the initial search points x and y satisfy $\textsc{OneMax}(x) \leq n-k$ and $\textsc{OneMax}(y) \leq n-k$, the expected optimization cost of Algorithm 3 on $\textsc{JumpCost}$ is $O(n\log(n/k) + \min\{n^k, c_{\max} n \log k\})$.*

Proof. We apply Theorem 8 with $m = n - k + 2$ levels. For all $j \in [0..n-k]$, we define level,

$$A_j = \{x \mid \textsc{OneMax}(x) = j\}.$$

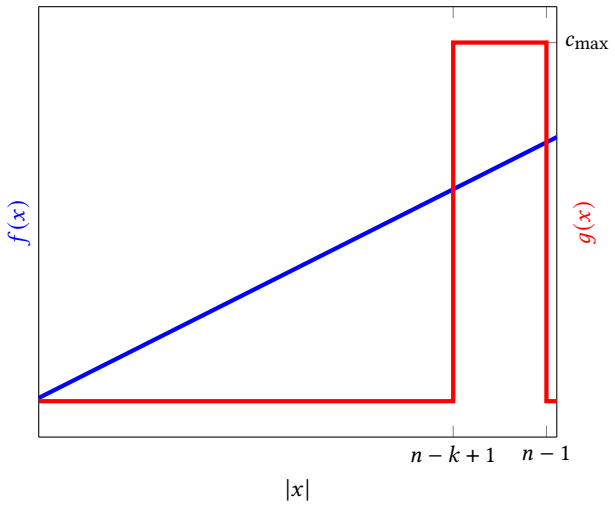

Figure 2: The $\textsc{JumpCost}_{k,c_{\max}}$ benchmark function.

For levels $j < n-k$ and $j = n-k+1$, there are $m_j = 0$ sub-levels. And for level $j = n-k$, there are $m_j = k$ sub-levels, where sub-level $i \in [0..k-1]$ is defined as

$$A_{n-k,i} = \{x \mid \textsc{OneMax}(x) = n-k+i\}.$$

Finally, for $j = n-k+1$, we define the level

$$A_{n-k+1} = \{x \mid \textsc{OneMax}(x) = n\}.$$

Condition 0 is satisfied by the assumptions $\textsc{OneMax}(x) \leq n-k$ and $\textsc{OneMax}(y) \leq n-k$. Conditions 1-4 of the theorem can be satisfied with the following parameters. A search point in sub-level $A_{n-k,i}$ has $k-i$ 0-bits, and can be "upgraded" by flipping any single of these bits. Hence, we can use the parameter $s_{n-k,i} = \Omega((k-i)/n)$. A search point in level $A_{n-k,0}$ has k 0-bits and can be "upgraded" to the final level A_{n-k+1} by mutating all of these bits simultaneously, which occurs with probability $p_{n-k} = \Omega(1/n^k)$. Finally, any search point in level A_j for $j < n-k$ have $n-j$ 0-bits, and can be upgraded to level A_{j+1} by flipping exactly one 0-bit, which occurs with probability $p_j = \Omega((n-j)/n)$.

Condition 5 is trivially satisfied because there exists only one level with high cost.

By Theorem 8, the expected cost is

$$\begin{aligned}
&4\sum_{j=1}^{m-1} \min\left\{\frac{1}{p_j}, c_{\max}\left(1+\sum_{i=0}^{m_j}\frac{1}{s_{j,i}}\right)\right\}. \\
&\quad\leq 4\sum_{j=0}^{n-k-1}\frac{1}{p_j} + 4\min\left\{\frac{1}{p_{n-k}}, c_{\max}\left(1+\sum_{i=0}^{m_{n-k}}\frac{1}{s_{n-k,i}}\right)\right\} \\
&\quad= O\Big(n\log(n/k) + \min\{n^k, c_{\max} n\log(k)\}\Big). \qquad \square
\end{aligned}$$

5 CONCLUSION

With this paper we have considered the setting in which the cost to evaluate the fitness of a solution is not uniform across the search space. These kinds of situations can occur in practical situations, for example, when obtaining an objective value requires running a costly simulation or comes from the accuracy measurement of a machine learning model that must first be trained. Traditional runtime analysis techniques assume fitness evaluations have unit cost and are therefore blind to these scenarios.

We have introduced two models of managing variable cost: a simple non-adaptive model that prefers points that were cheaper during the evaluation phase and an adaptive model that can decide to abort an incomplete evaluation in order to apportion its resources to better balance the total optimization cost.

In a non-adaptive setting, the simple strategy of favoring cheaper neutral mutations can be effective, especially on highly neutral landscapes such as LeadingOnes, in which case we proved the strategy gains a linear improvement factor when the hidden cost and fitness variables do not overlap. When the cost and fitness variables are shared, the advantage depends on the degree of the overlap.

For situations in which the process of evaluating the fitness function is costly but abortable, such as when fitness values are collected from simulation runs, we introduced the (2+2) Cost-balancing Evolutionary Algorithm (CBEA) that maintains two search points in parallel to balance the evaluation cost against the waiting time to jump over costly regions of the search space. For these scenarios we developed a cost-adaptive fitness level theorem and provided runtime bounds that depend on the correlation between maximum cost and escape probability.

ACKNOWLEDGMENTS

P.K. Lehre was supported by a Turing AI Fellowship (EPSRC grant ref EP/V025562/1). A.M. Sutton was supported by NSF grant 2144080.

REFERENCES

[1] Thomas Bartz-Beielstein and Martin Zaefferer. 2017. Model-based methods for continuous and discrete global optimization. *Applied Soft Computing* 55 (2017), 154–167. https://doi.org/10.1016/j.asoc.2017.01.039

[2] Stephan Cathabard, Per Kristian Lehre, and Xin Yao. 2011. Non-uniform mutation rates for problems with unknown solution lengths. In *Foundations of Genetic Algorithms, 11th International Workshop, FOGA 2011, Schwarzenberg, Austria, January 5-8, 2011, Proceedings*, Hans-Georg Beyer and William B. Langdon (Eds.). ACM, 173–180. https://doi.org/10.1145/1967654.1967670

[3] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2019. Solving Problems with Unknown Solution Length at Almost No Extra Cost. *Algorithmica* 81, 2 (2019), 703–748. https://doi.org/10.1007/s00453-018-0477-7

[4] Hafsteinn Einarsson, Marcelo Matheus Gauy, Johannes Lengler, Florian Meier, Asier Mujika, Angelika Steger, and Felix Weissenberger. 2019. The linear hidden subset problem for the (1+1) EA with scheduled and adaptive mutation rates. *Theoretical Computer Science* 785 (2019), 150–170. https://doi.org/10.1016/j.tcs.2019.05.021

[5] Thomas Jansen and Christine Zarges. 2011. Analysis of evolutionary algorithms: from computational complexity analysis to algorithm engineering. In *Foundations of Genetic Algorithms, 11th International Workshop, FOGA 2011, Schwarzenberg, Austria, January 5-8, 2011, Proceedings*, Hans-Georg Beyer and William B. Langdon (Eds.). ACM, 1–14. https://doi.org/10.1145/1967654.1967656

[6] Yaochu Jin. 2005. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.* 9, 1 (2005), 3–12. https://doi.org/10.1007/s00500-003-0328-5

[7] Per Kristian Lehre and Carsten Witt. 2013. General Drift Analysis with Tail Bounds. https://doi.org/10.48550/ARXIV.1307.2559

[8] Johannes Lengler and Nicholas Spooner. 2015. Fixed Budget Performance of the (1+1) EA on Linear Functions. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII, Aberystwyth, United Kingdom, January 17 - 20, 2015*, Jun He, Thomas Jansen, Gabriela Ochoa, and Christine Zarges (Eds.). ACM, 52–61. https://doi.org/10.1145/2725494.2725506

[9] Phuc Luong, Dang Nguyen, Sunil Gupta, Santu Rana, and Svetha Venkatesh. 2021. Adaptive cost-aware Bayesian optimization. *Knowl. Based Syst.* 232 (2021), 107481. https://doi.org/10.1016/j.knosys.2021.107481

[10] Ole Jakob Mengshoel, Eirik Lund Flogard, Tong Yu, and Jon Riege. 2022. Understanding the cost of fitness evaluation for subset selection: Markov chain analysis of stochastic local search. In *GECCO '22: Genetic and Evolutionary Computation Conference, Boston, Massachusetts, USA, July 9 - 13, 2022*, Jonathan E. Fieldsend and Markus Wagner (Eds.). ACM, 251–259. https://doi.org/10.1145/3512290.3528689

[11] Liang Shi and Khaled Rasheed. 2010. *A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms.* Springer Berlin Heidelberg, Berlin, Heidelberg, 3–28. https://doi.org/10.1007/978-3-642-10701-6_1

[12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[13] Carsten Witt. 2014. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Inform. Process. Lett.* 114, 1 (2014), 38–41. https://doi.org/10.1016/j.ipl.2013.09.013