Notes on Evolution and Learning

Dimitris Diochnos

University of Oklahoma
CS 5970 – Computational Learning Theory
Fall 2020

Abstract

These notes were created for CS 6501 - Learning Theory at UVA during the Spring of 2017. The primary scope of the notes is the exposition of Valiant's model of evolvability [8].

1 A Naive High-Level Description of Algorithms Inspired by Nature

Nature, and in particular evolution, has inspired different frameworks in computer science. The main model that we will examine is Valiant's model of *evolvability* [8]. However, another framework that is very close to evolvability is that of *evolutionary algorithms* [3, 9]. We note that there is a discussion in the introductory section of Valiant's seminal paper [8] on the comparison between the two models; in our discussion below we will try to illustrate such differences and similarities.

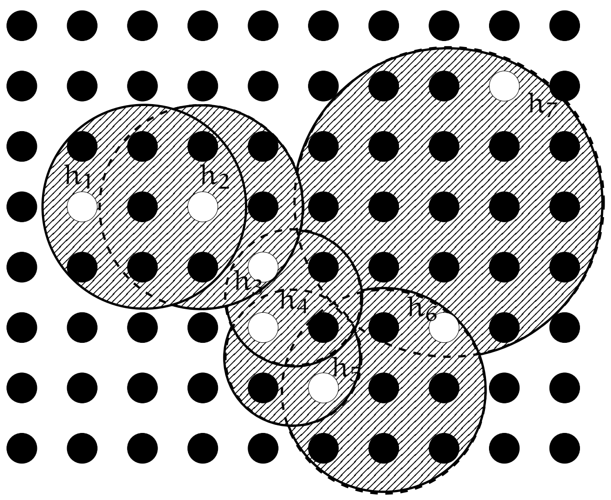

Figure 1: A local search procedure that starts from the hypothesis h_1, follows the hypotheses h_2, h_3, h_4, h_5, h_6 and h_7 in that order. For all the hypotheses apart from h_7 we see the respective neighborhoods with some circles encompassing the neighbors of each hypothesis. Note that both the radius of each circle, as well as the center need not be uniform throughout the whole process (though, typically there is little - if any - variation on the mechanism that generates the neighborhood based on each hypothesis).

From the point of view of computer science and in particular of learning, algorithms that are inspired by natural evolution, are local search procedures [5]; inevitably there is also connection to Markov chains [7]. Hence, in between the different iterations (generations) of these procedures, a pool of candidate hypotheses is generated (offsprings) forming a *neighborhood* where search is being performed. Search is being performed in the sense that a subset of the members of the neighborhood survives to the next iteration (generation). The hypotheses that survive in between the different iterations typically behave well, or better, compared to past hypotheses and in a sense are *naturally* selected. Thus, there is a notion of a *fitness function* that indicates how well a particular functionality (a member of the local search neighborhood between the iterations; as a parallel, think of an organism, a cell, etc.) that is being evolved, is *fit* to its environment. Figure 1 gives an example where only one hypothesis survives after each iteration.

2 An Informal Description of Evolvability

Evolution is seen as a form of learning. The truth values TRUE and FALSE are represented by 1 and -1 respectively. The fitness function that guides the search is called *performance*. For a target c and a fixed distribution $\mathcal{D}_n$ over $\{0,1\}^n$, the performance of a hypothesis h, called the *correlation* of h and c, is,

$$\begin{aligned}\mathrm{Perf}_{\mathcal{D}_n}(h,c) &= \sum_{x\in\{0,1\}^n} h(x)\cdot c(x)\cdot \mathbf{Pr}_{x\sim\mathcal{D}_n}(x) \\ &= 1-2\cdot\mathbf{Pr}_{x\sim\mathcal{D}_n}(h(x)\neq c(x))\,.\end{aligned} \tag{1}$$

Evolution starts with an initial hypothesis h_0, and produces a sequence of hypotheses using a local-search procedure in $\mathcal{H}$. Similarity between h and c in an underlying distribution $\mathcal{D}_n$ is measured by the *empirical performance* $\mathrm{Perf}_{\mathcal{D}_n}(h,c,|S|)$ which is evaluated approximately by drawing a random sample of size $|S|$ and computing

$$\mathrm{Perf}_{\mathcal{D}_n}(h,c,|S|) = \frac{1}{|S|}\sum_{x\in S} h(x)\cdot c(x)\,. \tag{2}$$

Naturally, we want the neighborhood with the candidate hypotheses in each step to have size bounded by $\mathrm{poly}(1/\varepsilon, 1/\delta, n, \mathrm{size}(h))$. Further, we also want this neighborhood to be computed *efficiently*; i.e. within $\mathrm{poly}(1/\varepsilon, 1/\delta, n, \mathrm{size}(h))$ time between each iteration (generation).

The mutator function is responsible for generating the neighborhood $N(h)$ and selecting one hypothesis from the neighborhood as the output for the next iteration (generation). For each hypothesis h_1 in the neighborhood, the mutator function first computes an empirical value of $\nu(h_1) = \mathrm{Perf}_{\mathcal{D}_n}(h,c,s)$ and also associates each hypothesis h_1 with a weight $\mathbf{Pr}_N(h,h_1)$. Then, based on the value of a real constant t called *tolerance*, we obtain[1]

$$\begin{cases}\text{Bene} &= \{h_1\in N(h)\mid \nu(h_1) > \nu(h)+t\}\\ \text{Neut} &= \{h_1\in N(h)\mid \nu(h_1)\geqslant \nu(h)-t\}\setminus \text{Bene}\end{cases}$$

The final output of the mutator function is based on the following rule[2]:

- if Bene $\neq\emptyset$ then output $h_1\in$ Bene with probability $\mathbf{Pr}_N(h,h_1)\,/\sum_{h_1\in\text{Bene}}\mathbf{Pr}_N(h,h_1)$
- if Bene $=\emptyset$ then output $r_1\in$ Neut with probability $\mathbf{Pr}_N(h,h_1)\,/\sum_{h_1\in\text{Neut}}\mathbf{Pr}_N(h,h_1)$.

Ultimately, the goal of the evolution is to produce, within a realistic time period (i.e. within $\mathrm{poly}(1/\varepsilon, 1/\delta, n)$ generations), a hypothesis $h\in\mathcal{H}$ such that

$$\mathbf{Pr}\left(\mathrm{Perf}_{\mathcal{D}_n}(h,c) < \mathrm{Perf}_{\mathcal{D}_n}(c,c)-\varepsilon\right) < \delta\,. \tag{3}$$

3 Monotone Conjunctions under the Uniform Distribution

Given a set of Boolean variables $x_1,\ldots,x_n$, we assume that there is an unknown target $c\in\mathcal{C}$, a monotone conjunction of some of these variables. Let $\mathcal{C}$ be the concept class of all possible monotone conjunctions in their natural representation. For a threshold q, let $\mathcal{C}^{\leqslant q}$ be the set of conjunctions from $\mathcal{C}$ that contain at most q variables. Further, let $\mathcal{C}^{>q}=\mathcal{C}\setminus\mathcal{C}^{\leqslant q}$. Now let,

$$c=\bigwedge_{i=1}^{m} x_i \wedge \bigwedge_{k=1}^{u} y_k \qquad \text{and} \qquad h=\bigwedge_{i=1}^{m} x_i\wedge\bigwedge_{\ell=1}^{r} w_\ell. \tag{4}$$

[1]Note that even if the performance of a hypothesis in the neighborhood is larger compared to the current hypothesis, it is the value of the tolerance t that decides whether this hypothesis in the neighborhood will be characterized as beneficial or neutral. In fact, in the context of evolvability, t should never be below some $\mathrm{poly}(\varepsilon,\delta,1/n)$.

[2]Additional rules can be defined and are discussed in [8], but they are beyond the scope of this class.

Variables in the target c are called *good*, otherwise *bad*. The x's are *mutual* variables, the y's are *undiscovered* (or *missing*) and the w's are *redundant* (or *wrong*). With $|h|$ we denote the *size* (or *length*) of a conjunction h; the number of variables that it contains. Given a size q, a hypothesis h is *short* when $|h| \leqslant q$, *medium* when $|h| = q+1$ and *long* when $|h| > q+1$. For a target c, $\mathcal{H}_{\geqslant 2}$ refers to hypotheses for which $u \geqslant 2$, $\mathcal{H}_{=1}$ refers to hypotheses for which $u = 1$ and $\mathcal{H}_{=0}$ refers to hypotheses for which $u = 0$.

For the *uniform distribution* $\mathcal{U}_n$ over $\{0,1\}^n$ (each truth assignment has probability 2^{-n}), (1) gives,

$$\mathrm{Perf}_{\mathcal{U}_n}(h, c) = 1 - 2^{1-m-r} - 2^{1-m-u} + 2^{2-m-r-u}. \tag{5}$$

4 Two Algorithms for the Evolution of Monotone Conjunctions

Valiant's Swapping Algorithm. The swapping algorithm for monotone conjunctions was introduced by Valiant in [8], was simplified in [2], subsequently analyzed with drifting targets in [6], and recently was obtained as a special case of a meta-algorithm for binomial distributions in [1]. The neighborhood N of a conjunction h is the set of conjunctions that arise by *adding* a variable (neighborhood N^+), *removing* a variable (neighborhood N^-), or *swapping* a variable with another one (neighborhood $N^\pm$), plus the conjunction itself[3]. Thus, $N = N^- \cup N^+ \cup N^\pm \cup \{h\}$. As an example, let $h = x_1 \wedge x_2$, and $n = 3$. Then, $N^- = \{x_1, x_2\}$, $N^+ = \{x_1 \wedge x_2 \wedge x_3\}$, and $N^\pm = \{x_3 \wedge x_2, x_1 \wedge x_3\}$. Note that $|N| = \mathcal{O}(n\,|h|)$ in general. Algorithm 1 presents the mutator function for the swapping algorithm.

Algorithm 1: Mutator function for the swapping algorithm under the uniform distribution.

```
Input: dimension $n$, $\delta \in (0,1)$, $\varepsilon \in (0,2)$, $\mathcal{H} \in \{\mathcal{C}^{\leqslant q}, \mathcal{C}\}$, $h \in \mathcal{H}$
Output: a new hypothesis $h'$
1  $q \leftarrow \lceil \lg(5/(2\varepsilon)) \rceil$;
2  if $|h| > 0$ then Generate $N^-$ else $N^- \leftarrow \emptyset$;
3  if $|h| < q$ then Generate $N^+$ else $N^+ \leftarrow \emptyset$;
4  if $|h| \leqslant q$ then Generate $N^\pm$ else $N^\pm \leftarrow \emptyset$;
5  Bene $\leftarrow \emptyset$; Neutral $\leftarrow \{h\}$;
6  $t_\ell \leftarrow 2^{-2q}$; $t_u \leftarrow 2^{1-q}$;
7  if ($\mathcal{H} = \mathcal{C}$) and ($|h| > q$) then $t \leftarrow t_u$; $\epsilon_s \leftarrow t_u/4$; $\delta_s \leftarrow \delta/(8n^2 \lg(8n/\delta))$ ;
8  else $t \leftarrow t_\ell$; $\epsilon_s \leftarrow t_\ell$; $\delta_s \leftarrow \delta/(12q^2 n)$ ;
9  SetWeight($h, h, N^-, N^+, N^\pm$); $v_h \leftarrow$ Perf($h, \epsilon_s, \delta_s$);
10 for $x \in N^+, N^-, N^{+-}$ do
       SetWeight($x, h, N^-, N^+, N^\pm$); $v_x \leftarrow$ Perf($x, \epsilon_s, \delta_s$);
       if $v_x > v_h + t$ then Bene $\leftarrow$ Bene $\cup \{x\}$ ;
       else if $v_x \geqslant v_h - t$ then Neutral $\leftarrow$ Neutral $\cup \{x\}$ ;
14 if Bene $\neq \emptyset$ then return Select(Bene) else return Select(Neutral);
```

Tolerance t is normally t_ℓ; however, when $\mathcal{H} = \mathcal{C}$ and $|h| > q$ then $t = t_u$. `Perf` computes the empirical performance of h w.r.t. c over the distribution $\mathcal{U}_n$ within ϵ_s of its true value, with probability at least $1 - \delta_s$. `SetWeight` assigns the same weight to all members of $\{h\} \cup N^- \cup N^+$ so that they add up to $\frac{1}{2}$, and the same weight to all the members of $N^\pm$ so that they add up to $\frac{1}{2}$. `Select` computes the sum of weights W of the conjunctions in the set passed as argument, and returns a hypothesis h' with probability $w_{h'}/W$, where $w_{h'}$ is the weight of h'.

The (1+1) EA. For a hypothesis h, the operator `Mutate` generates a hypothesis h' by first initializing h' to h and then flipping each bit with probability $1/n$. Further, when h is short, h' is accepted as a viable neighbor only if h' is also short. When h is long, h' is accepted as a viable neighbor only if $|h'| \leqslant |h|$,

[3]As h will be clear from the context, we write N instead of $N(h)$.

otherwise h is the only candidate in the neighborhood again. Algorithm 2 presents the mutator function. `USelect`, returns uniformly at random one of the elements of the set that is passed as argument.

Algorithm 2: Mutator function of the (1+1) EA under the uniform distribution.

```
Input: dimension $n$, $\delta \in (0,1)$, $\varepsilon \in (0,2)$, $\mathcal{H} \in \{\mathcal{C}^{\leqslant q}, \mathcal{C}\}$, $h \in \mathcal{H}$
Output: a new hypothesis
1 $q \leftarrow \lceil \lg(5/(2\varepsilon)) \rceil$;
2 $h' \leftarrow$ Mutate(h);
3 if $\left((|h| > q) \textit{ and } (|h'| \leqslant |h|)\right)$ or $\left((|h| \leqslant q) \textit{ and } (|h'| \leqslant q)\right)$ then $N \leftarrow \{h'\}$; else $N \leftarrow \emptyset$;
4 $t_\ell \leftarrow 2^{-2q}$; $t_u \leftarrow 2^{1-q}$;
5 if $|h| > q$ then $t \leftarrow t_u$; $\epsilon_s \leftarrow t_u/4$; $\delta_s \leftarrow \delta^2/(72en^3)$; else $t \leftarrow t_\ell$; $\epsilon_s \leftarrow t_\ell$; $\delta_s \leftarrow \delta^2/(544en^2q)$;
6 $v_h \leftarrow$ Perf($h, \epsilon_s, \delta_s$);
7 if $N = \emptyset$ then return $h$; else $v_{h'} \leftarrow$ Perf($h', \epsilon_s, \delta_s$);
8 if $v_{h'} > v_h + t$ then return $h'$; else if $v_{h'} \geqslant v_h - t$ return USelect($\{h\} \cup \{h'\}$); else return $h$;
```

5 On the Local Search Procedure

Local search, when switching to h' from h, is guided by the quantity

$$\Delta = \mathrm{Perf}_{\mathcal{U}_n}(h', c) - \mathrm{Perf}_{\mathcal{U}_n}(h, c) \; . \tag{6}$$

Figure 2 presents the sign of Δ when we deal with hypotheses that are obtained from the current one, by adding, removing, or swapping a variable. Note that while the sign of an arrow may be fully determined, it is the value of the tolerance t that defines the sets Bene and Neut that guide the search.

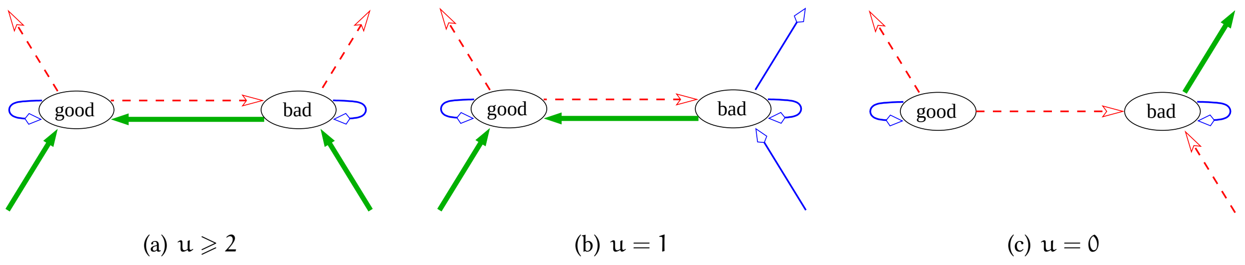

Figure 2: Arrows pointing towards the nodes indicate addition of one variable and arrows pointing away from a node indicate removal of one variable. This is consistent with arrows indicating swapping a pair of variables. Let Δ be as in (6). Thick solid lines indicate $\Delta > 0$. Simple lines indicate $\Delta = 0$. Dashed lines indicate $\Delta < 0$. Figure 2(a) holds when $u \geqslant 2$; Figure 2(b) when $u = 1$; Figure 2(c) when $u = 0$.

Comparing $h' \in N^+$ with h. We introduce a variable z in the hypothesis h. If z is good, $\Delta = 2^{-|h|} > 0$. If z is bad, $\Delta = 2^{-|h|}(1 - 2^{1-u})$.

Comparing $h' \in N^-$ with h. We remove a variable z from the hypothesis h. If z is good, $\Delta = -2^{1-|h|} < 0$. If z is bad, $\Delta = -2^{1-|h|}(1 - 2^{1-u})$.

Comparing $h' \in N^\pm$ with h. Replacing a good with a bad variable gives $\Delta = -2^{1-|h|-u} < 0$. Replacing a good with a good, or a bad with a bad variable gives $\Delta = 0$. Replacing a bad with a good variable gives $\Delta = 2^{2-|h|-u} > 0$.

6 Structure of Solutions under the Uniform Distribution

Definition 6.1 (Best q-Approximation)**.** *Let h be a hypothesis such that $|h| \leqslant q$ and $\forall h' \neq h, |h'| \leqslant q$: $Perf_{\mathcal{D}_n}(h', c) \leqslant Perf_{\mathcal{D}_n}(h, c)$. We call h a best q-approximation of c.*

Theorem 6.2 (Structure of Best Approximations; [2])**.** *The best q-approximation of a target monotone conjunction c is c if $|c| \leqslant q$, or any hypothesis formed by q good variables if $|c| > q$.*

Lemma 6.3 (Performance Lower Bound, Medium Target)**.** *Let c be a target monotone conjunction of medium size. A best q-approximation h of c has $Perf_{\mathcal{U}_n}(h, c) = 1 - 2^{-q}$.*

Proof. Theorem 6.2 implies $m = q$, $r = 0$, $u = 1$. Using (5) we have: $\mathrm{Perf}_{\mathcal{U}_n}(h, c) = 1 - 2^{1-m-r} - 2^{1-m-u} + 2^{2-m-r-u} = 1 - 2^{1-q} - 2^{-q} + 2^{1-q} = 1 - 2^{-q}$. □

Corollary 6.4. $q \geqslant \left\lceil \lg\left(\frac{5}{2\varepsilon}\right) \right\rceil$, $|h| = q$, $|c| = q+1$, *all variables in h are good* $\Longrightarrow$ *$Perf_{\mathcal{U}_n}(h, c) > 1 - \frac{2\varepsilon}{5}$.*

Lemma 6.5 (Performance Lower Bound, Long Target)**.** *Let h be a hypothesis such that $|h| \geqslant q$ and consider a long target monotone conjunction c. Then, $Perf_{\mathcal{U}_n}(h, c) > 1 - \frac{5}{2} \cdot 2^{-q}$.*

Proof. The setup of the lemma implies $m + r \geqslant q$, and $m + u \geqslant q + 2$. Using (5), $\mathrm{Perf}_{\mathcal{U}_n}(h, c) > 1 - 2^{1-m-r} - 2^{1-m-u} \geqslant 1 - 2^{1-q} - 2^{-1-q} = 1 - \frac{5}{2} \cdot 2^{-q}$. □

Corollary 6.6. $q \geqslant \left\lceil \lg\left(\frac{5}{2\varepsilon}\right) \right\rceil$, $|h| \geqslant q$, $|c| > q + 1 \Longrightarrow$ *$Perf_{\mathcal{U}_n}(h, c) > 1 - \varepsilon$.*

7 Convergence and Complexity of the Swapping Algorithm

Proposition 7.1 ([4])**.** *Let $X_1, \ldots, X_R$ be R independent random variables, each taking values in the range $\mathcal{I} = [\alpha, \beta]$. Let μ denote the mean of their expectations. Then* $\mathbf{Pr}\left(\left|\frac{1}{R}\sum_{i=1}^{R} X_i - \mu\right| \geqslant \epsilon\right) \leqslant 2e^{-2R\epsilon^2/(\beta-\alpha)^2}$.

Within $2q$ generations we will form a solution satisfying the following: when the target is short, it will be identified precisely; when the target is medium, a best q-approximation will be returned; when the target is long, a hypothesis of size q will be returned (not necessarily a best q-approximation).

Theorem 7.2 (Evolution starting in $\mathcal{C}^{\leqslant q}$ under $\mathcal{U}_n$)**.** *Let the distribution be $\mathcal{U}_n$ over $\{0, 1\}^n$. Let $q = \lceil \lg(5/(2\varepsilon)) \rceil$. Starting with a short hypothesis and considering hypotheses in $\mathcal{C}^{\leqslant q}$, the swapping algorithm, in at most $2q$ generations, will evolve a hypothesis h such that $\mathbf{Pr}\left(Perf_{\mathcal{U}_n}(h, c) > 1 - \varepsilon\right) \geqslant 1 - \delta/2$, using total sample size $\mathcal{O}\left(\frac{n}{\varepsilon^4} \cdot \left(\ln \frac{1}{\varepsilon}\right)^2 \cdot \left(\ln n + \ln \frac{1}{\delta} + \ln \ln \frac{1}{\varepsilon}\right)\right)$.*

Proof. Let $h \in \mathcal{C}^{\leqslant q}$. Then, $|N| = \left|N^+ \cup N^- \cup N^{\pm} \cup \{h\}\right| = |N^+| + |N^-| + \left|N^{\pm}\right| + 1 = (n - |h|) + |h| + |h|(n - |h|) + 1 = (|h| + 1)n - |h|^2 + 1 \leqslant (|h| + 2)n \leqslant (q + 2)n$. Our solution strategy implies that evolution requires no more than $2q$ generations for the convergence. As a consequence, evolution needs to compute the empirical performance of at most $2q(q+2)n = 2q^2n + 4qn \leqslant 6q^2n$ hypotheses. Tolerance is $t = t_\ell = 2^{-2q} \geqslant \varepsilon^2/25$. Requiring $R \geqslant \left\lceil \frac{2}{t_\ell^2} \ln\left(\frac{24q^2n}{\delta}\right) \right\rceil$ samples for estimating the empirical performance of each hypothesis, it follows by Hoeffding's bound (Proposition 7.1), using $\alpha = -1$ and $\beta = 1$, that the empirical performance of each hypothesis is estimated within $\epsilon_s = t_\ell$ of its exact value with probability at least $1 - \delta/(12q^2n)$. As the number of different hypotheses is not more than $6q^2n$, by the union bound, the performance of *every* hypothesis in this phase is computed within $\epsilon_s = t_\ell$ of its exact value with probability at least $1 - \sum_{i=1}^{6q^2n} \delta/(12q^2n) = 1 - \delta/2$. □

8 Convergence and Complexity of the (1+1) EA

Lemma 8.1. $h_1 \in \mathcal{H}_{\geqslant 2}, h_2 \in \mathcal{H}_{\geqslant 2}, |h_2| < |h_1| \leqslant q \Longrightarrow \mathit{Perf}_{\mathcal{U}_n}(h_1, c) \geqslant \mathit{Perf}_{\mathcal{U}_n}(h_2, c) + 2^{1-2q}$.

Lemma 8.2. $h_1 \in \mathcal{H}_{=1}, h_2 \in \mathcal{H}_{\geqslant 2}, |h_2| \leqslant q \Longrightarrow \mathit{Perf}_{\mathcal{U}_n}(h_1, c) \geqslant \mathit{Perf}_{\mathcal{U}_n}(h_2, c) + 2^{-q}$.

Lemma 8.3. $h_1 \in \mathcal{H}_{=0}, h_2 \in \mathcal{H}_{=1}, |h_1| \leqslant q \Longrightarrow \mathit{Perf}_{\mathcal{U}_n}(h_1, c) \geqslant \mathit{Perf}_{\mathcal{U}_n}(h_2, c) + 2^{1-q}$.

Lemma 8.4. $h_1 \in \mathcal{H}_{=0}, h_2 \in \mathcal{H}_{\geqslant 2}, |h_1| \leqslant q, |h_2| \leqslant q \Longrightarrow \mathit{Perf}_{\mathcal{U}_n}(h_1, c) \geqslant \mathit{Perf}_{\mathcal{U}_n}(h_2, c) + 3 \cdot 2^{-q}$.

Lemma 8.5 (Phase I)**.** *Let $\mathcal{D}_n = \mathcal{U}_n$. Starting with a short hypothesis h_0, the (1+1) EA, assuming all the empirical estimates behave as expected, within $\lceil 16enq/\delta \rceil$ generations, with probability at least $1 - \delta/16$, will evolve a hypothesis h such that, either $h \in \mathcal{H}_{=1} \cup \mathcal{H}_{=0}$, or it is the case that $h \in \mathcal{H}_{\geqslant 2}$ and $|h| = q$.*

Lemma 8.6 (Phase II)**.** *Let $\mathcal{D}_n = \mathcal{U}_n$. Unless the target is long, starting with a hypothesis h_0 such that $h_0 \in \mathcal{H}_{\geqslant 2}$ and $|h_0| = q$, the (1+1) EA, assuming all the empirical estimates behave as expected, within $\lceil 16en^2q/\delta \rceil$ generations, with probability at least $1 - \delta/16$, will evolve a hypothesis $h \in \mathcal{H}_{=1} \cup \mathcal{H}_{=0}$.*

Lemma 8.7 (Phase III)**.** *Let $\mathcal{D}_n = \mathcal{U}_n$. Let $h_0 \in \mathcal{H}_{=1}$ such that $|h_0| \leqslant q$. The (1+1) EA, assuming all the empirical estimates behave as expected, within $\lceil 16en^2/\delta \rceil$ generations, with probability at least $1 - \delta/16$, will, either maintain a best q-approximation for a target of size $q + 1$, or evolve a hypothesis $h \in \mathcal{H}_{=0}$.*

Lemma 8.8 (Phase IV)**.** *Let $\mathcal{D}_n = \mathcal{U}_n$. For an initial hypothesis $h_0 \in \mathcal{H}_{=0}$ such that $|h_0| \leqslant q$, the (1+1) EA, assuming all the empirical estimates behave as expected, within $\lceil 16enq/\delta \rceil$ generations, with probability at least $1 - \delta/16$, will evolve to the target c.*

Theorem 8.9 (Evolution Starting in $\mathcal{C}^{\leqslant q}$ under $\mathcal{U}_n$)**.** *Let $\mathcal{D}_n = \mathcal{U}_n$. Let $q = \lceil \lg(5/(2\varepsilon)) \rceil$. Starting with a short hypothesis, the (1+1) EA, in $\mathcal{O}\left(n^2q/\delta\right)$ generations, will evolve a hypothesis h such that* **Pr** $\left(\mathit{Perf}_{\mathcal{U}_n}(h, c) > 1 - \varepsilon\right) \geqslant 1 - \delta/2$*, using total sample size* $\mathcal{O}\left(\frac{n^2}{\delta\varepsilon^4} \cdot \ln\frac{1}{\varepsilon} \cdot \left(\ln n + \ln\frac{1}{\delta} + \ln\ln\frac{1}{\varepsilon}\right)\right)$.

References

[1] Dimitrios I. Diochnos. On the Evolution of Monotone Conjunctions: Drilling for Best Approximations. In *ALT*, pages 98–112, 2016.

[2] Dimitrios I. Diochnos and György Turán. On Evolvability: The Swapping Algorithm, Product Distributions, and Covariance. In *SAGA*, pages 74–88, 2009.

[3] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[4] Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[5] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004.

[6] Varun Kanade, Leslie G. Valiant, and Jennifer Wortman Vaughan. Evolution with Drifting Targets. In *COLT*, pages 155–167, 2010.

[7] John George Kemeny and James Laurie Snell. *Finite Markov Chains*. Springer, 1983.

[8] Leslie G. Valiant. Evolvability. *Journal of the ACM*, 56(1):3:1–3:21, 2009.

[9] Ingo Wegener. Methods for the Analysis of Evolutionary Algorithms on Pseudo-Boolean Functions. In *Evolutionary Optimization*, pages 349–369. Springer, 2003.