The Fundamental Learning Problem that Genetic Algorithms with Uniform Crossover Solve Efficiently and Repeatedly As Evolution Proceeds

Keki M. Burjorjee
Zite, Inc.
487 Bryant St.
San Francisco, CA, USA
kekib@cs.brandeis.edu

Abstract

This paper establishes theoretical bonafides for *implicit concurrent multivariate effect evaluation*—implicit concurrency[1] for short—a broad and versatile computational learning efficiency thought to underlie general-purpose, non-local, noise-tolerant optimization in genetic algorithms with uniform crossover (UGAs). We demonstrate that implicit concurrency is indeed a form of efficient learning by showing that it can be used to obtain close-to-optimal bounds on the time and queries required to approximately correctly solve a constrained version ($k = 7, \eta = 1/5$) of a recognizable computational learning problem: learning parities with noisy membership queries. We argue that a UGA that treats the noisy membership query oracle as a fitness function can be straightforwardly used to approximately correctly learn the essential attributes in $O(\log^{1.585} n)$ queries and $O(n \log^{1.585} n)$ time, where n is the total number of attributes. Our proof relies on an accessible symmetry argument and the use of statistical hypothesis testing to reject a global null hypothesis at the 10^{-100} level of significance. It is, to the best of our knowledge, the first relatively rigorous identification of efficient computational learning in an evolutionary algorithm on a non-trivial learning problem.

1 Introduction

We recently hypothesized [2] that an efficient form of computational learning underlies general-purpose, non-local, noise-tolerant optimization in genetic algorithms with uniform crossover (UGAs). The hypothesized computational efficiency, *implicit concurrent multivariate effect evaluation—implicit concurrency* for short—is broad and versatile, and carries significant implications for efficient large-scale, general-purpose global optimization in the presence of noise, and, in turn, large-scale machine learning. In this paper, we describe implicit concurrency and explain how it can power general-purpose, non-local, noise-tolerant optimization. We then establish that implicit concurrency is a bonafide form of efficient computational learning by using it to obtain close to optimal bounds on the query and time complexity of an algorithm that solves a constrained version of a problem from the computational learning literature: learning parities with a noisy membership query oracle [18, 6].

---

[1] http://bit.ly/YtwdST

2   Implicit Concurrenct Multivariate Effect Evaluation

First, a brief primer on schemata and schema partitions [13]: Let $S = \{0, 1\}^n$ be a search space consisting of binary strings of length n and let $\mathcal{I}$ be some set of indices between 1 and n, i.e. $\mathcal{I} \subseteq \{1, \ldots, n\}$. Then $\mathcal{I}$ represents a partition of S into $2^{|\mathcal{I}|}$ subsets called schemata (singular schema) as in the following example: Suppose $n = 5$, and $\mathcal{I} = \{1, 2, 4\}$, then $\mathcal{I}$ partitions S into eight schemata:

00*0*	00*1*	01*0*	01*1*	10*0*	10*1*	11*0*	11*1*
00000	00010	01000	01000	01010	10000	10010	11010
00001	00011	01001	01001	01011	10001	10011	11011
00100	00110	01100	01100	01110	10100	10110	11110
00101	00111	01101	01101	01111	10101	10111	11111

where the symbol $*$ stands for 'wildcard'. Partitions of this type are called schema partitions. As we've already seen, schemata can be expressed using templates, for example, $10 * 1*$. The same goes for schema partitions. For example $\#\# * \#*$ denotes the schema partition represented by the index set $\{1, 2, 4\}$; the one shown above. Here the symbol $\#$ stands for 'defined bit'. The *fineness order* of a schema partition is simply the cardinality of the index set that defines the partition, which is equivalent to the number of $\#$ symbols in the schema partition's template (in our running example, the fineness order is 3). Clearly, schema partitions of lower fineness order are coarser than schema partitions of higher fineness order.

We define the *effect* of a schema partition to be the variance[2] of the average fitness values of the constituent schemata under sampling from the uniform distribution over each schema in the partition. So for example, the effect of the schema partition $\#\# * \#* = \{00 * 0*, 00 * 1*, 01 * 0*, 01 * 1*, 10 * 0*, 10 * 1*, 11 * 0*, 11 * 1*\}$ is

$$\frac{1}{8} \sum_{i=0}^{1} \sum_{j=0}^{1} \sum_{k=0}^{1} (F(ij * k*) - F(* * **))^2$$

where the operator F gives the average fitness of a schema under sampling from the uniform distribution.

Let $[\![\mathcal{I}]\!]$ denote the schema partition represented by some index set $\mathcal{I}$. Consider the following intuition pump [4] that illuminates how effects change with the coarseness of schema partitions. For some large n, consider a search space $S = \{0, 1\}^n$, and let $\mathcal{I} = [n]$. Then $[\![\mathcal{I}]\!]$ is the finest possible partition of S; one where each schema in the partition has just one point. Consider what happens to the effect of $[\![\mathcal{I}]\!]$ as we remove elements from $\mathcal{I}$. It is easily seen that the effect of $[\![\mathcal{I}]\!]$ decreases monotonically. Why? Because we are averaging over points that used to be in separate partitions. Secondly, observe that the number of schema partitions of order k is $\binom{n}{k}$. Thus, when $k \ll n$, the number of schema partitions with fineness order k will grow very fast with k (sub-exponentially to be sure, but still very fast). For example, when $n = 10^6$, the number of schema partitions of fineness order $2, 3, 4$ and 5 are on the order of $10^{11}, 10^{17}, 10^{22}$, and 10^{28} respectively.

The point of this exercise is to develop the following intuition: when n is large, a search space will have vast numbers of coarse schema partitions, but most of them will have negligible

---

[2]We use variance because it is a well known measure of dispersion. Other measures of dispersion may well be substituted here without affecting the discussion

effects due to averaging. In other words, while coarse schema partitions are numerous, ones with non-negligible effects are rare. Implicit concurrent multivariate effect evaluation is a capacity for scaleably learning (with respect to n) small numbers of coarse schema partitions with non-negligible effects. It amounts to a capacity for efficiently performing vast numbers of concurrent effect/no-effect multivariate analyses to identify small numbers of interacting loci.

2.1 Use (and Abuse) of Implicit Concurrency

Assuming implicit concurrency is possible, how can it be used to power efficient general-purpose, non-local, noise-tolerant optimization? Consider the following heuristic: Use implicit concurrency to identify a coarse schema partition $[\![\mathcal{I}]\!]$ with a significant effect. Now limit future search to the schema in this partition with the highest average sampling fitness. Limiting future search in this way amounts to permanently setting the bits at each locus whose index is in $\mathcal{I}$ to some fixed value, and performing search over the remaining loci. In other words, picking a schema effectively yields a new, lower-dimensional search space. Importantly, coarse schema partitions in the new search space that had negligible effects in the old search space may have detectable effects in the new space. Use implicit concurrency to pick one and limit future search to a schema in this partition with the highest average sampling fitness. Recurse.

Such a heuristic is *non-local* because it does not make use of neighborhood information of any sort. It is *noise-tolerant* because it is sensitive only to the *average* fitness values of coarse schemata. We claim it is *general-purpose* firstly, because it relies on a very weak assumption about the distribution of fitness over a search space—the existence of *staggered conditional effects* [2]; and secondly, because it is an example of a *decimation heuristic*, and as such is in good company. Decimation heuristics such as Survey Propagation [12, 10] and Belief Propagation [11], when used in concert with local search heuristics (e.g. *WalkSat* [17]), are state of the art methods for solving large instances of a number of NP-Hard combinatorial optimization problems close to their solvability/unsolvability thresholds.

The hyperclimbing hypothesis [2] posits that by and large, the heuristic described above *is* the abstract heuristic that UGAs implement—or as is the case more often than not, *misimplement* (it stands to reason that a "secret sauce" computational efficiency that stays secret will not be harnessed fully). One difference is that in each "recursive step", a UGA is capable of identifying *multiple* (some small number greater than one) coarse schema partitions with non-negligible effects; another difference is that for each coarse schema partition identified, the UGA does not always pick the schema with the highest average sampling fitness.

2.2 Needed: The Scientific Method, Practiced With Rigor

Unfortunately, several aspects of the description above are not crisp. (How small is small? What constitutes a "negligible" effect?) Indeed, a formal statement of the above, much less a formal proof, is difficult to provide. Evolutionary Algorithms are typically constructed with biomimicry, not formal analyzability, in mind. This makes it difficult to formally state/prove complexity theoretic results about them without making simplifying assumptions that effectively neuter the algorithm or the fitness function used in the analysis.

We have argued previously [2] that the adoption of the scientific method [15] is a necessary

and appropriate response to this hurdle. Science, *rigorously practiced*, is, after all, the foundation of many a useful field of engineering. A hallmark of a rigorous science is the ongoing making and testing of predictions. Predictions found to be true lend credence to the hypotheses that entail them. The more unexpected a prediction (in the absence of the hypothesis), the greater the credence owed the hypothesis if the prediction is validated [15, 14].

The work that follows validates a prediction that is straightforwardly entailed by the hyper-climbing hypothesis, namely that a UGA that uses a noisy membership query oracle as a fitness function should be able to efficiently solve the learning parities problem for small values of k, the number of essential attributes, and non-trivial values of η, where $0 < \eta < 1/2$ is the probability that the oracle makes a classification error (returns a 1 instead of a 0, or vice versa). Such a result is completely unexpected in the absence of the hypothesis.

2.3 Implicit Concurrency $\neq$ Implicit Parallelism

Given its name and description in terms of concepts from schema theory, implicit concurrency bears a superficial resemblance to *implicit parallelism*, the hypothetical "engine" presumed, under the beleaguered *building block hypothesis* [7, 16], to power optimization in genetic algorithms with strong linkage between genetic loci. The two hypothetical phenomena are emphatically not the same. We preface a comparison between the two with the observation that strictly speaking, implicit concurrency and implicit parallelism pertain to different kinds of genetic algorithms—ones with tight linkage between genetic loci, and ones with no linkage at all. This difference makes these hypothetical engines of optimization non-competing from a scientific perspective. Neverthe-less a comparison between the two is instructive for what it reveals about the power of implicit concurrency.

The *unit of implicit parallel evaluation* is a schema h belonging to a coarse schema partition $[\![\mathcal{I}]\!]$ that satisfies the following *adjacency constraint*: the elements of $\mathcal{I}$ are adjacent, or close to adjacent (e.g. $\mathcal{I} = \{439, 441, 442, 445\}$). The *evaluated characteristic* is the average fitness of samples drawn from h, and the *outcome of evaluation* is as follows: the frequency of h rises if its evaluated characteristic is greater than the evaluated characteristics of the other schemata in $[\![\mathcal{I}]\!]$.

The *unit of implicit concurrent evaluation*, on the other hand, is the coarse schema partition $[\![\mathcal{I}]\!]$, where the elements of $\mathcal{I}$ are unconstrained. The *evaluated characteristic* is the effect of $[\![\mathcal{I}]\!]$, and the *outcome of evaluation* is as follows: if the effect of $[\![\mathcal{I}]\!]$ is non negligible, then a schema in this partition with an above average sampling fitness goes to fixation, i.e. the frequency of this schema in the population goes to 1.

Implicit concurrency derives its superior power viz-a-viz implicit parallelism from the absence of an adjacency constraint. For example, for chromosomes of length n, the number of schema partitions with fineness order 7 is $\binom{n}{7} \in \Omega(n^7)$ [3]. The enforcement of the adjacency constraint, brings this number down to $O(n)$. Schema partition of fineness order 7 contain a constant number of schemata (128 to be exact). Thus for fineness order 7, the number of units of evaluation under implicit parallelism are in $O(n)$, whereas the number of units of evaluation under implicit concurrency are in $O(n^7)$.

4

3 The Learning Model

For any positive integer n, let $[n]$ denote the set $\{1, \ldots, n\}$. For any set K such that $|K| < n$ and any binary string $x \in \{0,1\}^n$, let $\pi_K(x)$ denote the string $y \in \{0,1\}^{|K|}$ such that for any $i \in [|K|], y_i = x_j$ iff j is the ith smallest element of K (i.e. π_K strips out bits in x whose indices are not in K). An *essential attribute oracle with random classification error* is a tuple $\phi = \langle k, f, n, K, \eta \rangle$ where n and k are positive integers, such that $|K| = k$ and $k < n$, f is a boolean function over $\{0,1\}^k$ (i.e. $f : \{0,1\}^k \to \{0,1\}$), and η, the random classification error parameter, obeys $0 < \eta < 1/2$. For any input bitstring x,

$$\phi(x) = \left\{ \begin{array}{l} f(\pi_K(x)) \text{ with probability } 1 - \eta \\ \neg f(\pi_K(x)) \text{ with probability } \eta \end{array} \right.$$

Clearly, the value returned by the oracle depends only on the bits of the attributes whose indices are given by the elements of K. These attributes are said to be *essential*. All other attributes are said to be *non-essential*. The *concept space* $\mathcal{C}$ of the learning model is the set $\{0,1\}^n$ and the *target concept* given the oracle is the element $c^* \in \mathcal{C}$ such that for any $i \in [n]$, $c_i^* = 1 \iff i \in K$. The hypothesis space $\mathcal{H}$ is the same as the concept space, i.e. $\mathcal{H} = \{0,1\}^n$.

Definition 1 (Approximately Correct Learning)**.** *Given some positive integer k, some boolean function $f : \{0,1\}^k \to \{0,1\}$, and some random classification error parameter $0 < \eta < 1/2$, we say that the learning problem $\langle k, f, \eta \rangle$ can be* approximately correctly solved *if there exists an algorithm $\mathcal{A}$ such that for any oracle $\phi = \langle n, k, f, K, \eta \rangle$ and any $0 < \epsilon < 1/2$, $\mathcal{A}^\phi(n, \epsilon)$ returns a hypothesis $h \in \mathcal{H}$ such that $P(h \neq c^*) < \epsilon$, where $c^* \in \mathcal{C}$ is the target concept.*

4 Our Result and Approach

Let $\oplus_k$ denote the parity function over k bits. For $k = 7$ and $\eta = 1/5$, we give an algorithm that approximately correctly learns $\langle k, \oplus_k, \eta \rangle$ in $O(\log^{1.585} n)$ queries and $O(n \log^{1.585} n)$ time. Our argument relies on the use of hypothesis testing to reject two null hypotheses, each at a Bonferroni adjusted significance level of $10^{-100}/2$. In other words, we rely on a hypothesis testing based rejection of a global null hypothesis at the 10^{-100} level of significance. In layman's terms, our result is based on conclusions that have a 1 in 10^{100} chance of being false.

While approximately correct learning lends itself to straightforward comparisons with other forms of learning in the computational learning literature, the following, weaker, definition of learning more naturally captures the computation performed by implicit concurrency.

Definition 2 (Attributewise ϵ-Approximately Correct Learning)**.** *Given some positive integer k, some boolean function $f : \{0,1\}^k \to \{0,1\}$, some random classification error parameter $0 < \eta < 1/2$, and some $0 < \epsilon < 1/2$, we say that the learning problem $\langle k, f, \eta \rangle$ can be* attribute-wise ϵ-approximately correctly solved *if there exists an algorithm $\mathcal{A}$ such that for any oracle $\phi = \langle n, k, f, K, \eta \rangle$, $\mathcal{A}^\phi(n)$ returns a hypothesis $h \in \mathcal{H}$ such that for all $i \in [n]$, $P(h_i \neq c_i^*) < \epsilon$, where $c^* \in \mathcal{C}$ is the target concept.*

Our argument is comprised of two parts. In the first part, we rely on a symmetry argument and the rejection of two null hypotheses, each at a Bonferroni adjusted significance level of $10^{-100}/2$,

$$\overbrace{\hspace{9cm}}^{n}$$

1	0	1	1	0	1	0	0	1	0	1	0	1	0	0	⋯	1	0
0	0	1	0	1	0	0	1	1	0	1	0	0	1	0	⋯	0	0
1	0	0	1	0	0	1	1	1	0	1	0	1	1	0	⋯	1	0
1	1	0	0	1	0	1	0	0	0	1	1	0	1	0	⋯	0	1
1	0	0	0	0	1	0	1	0	0	1	0	0	1	0	⋯	1	0
0	0	0	1	0	1	0	0	1	1	0	1	0	0	0	⋯	0	0
1	1	0	1	0	0	1	1	1	0	1	1	0	1	0	⋯	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮
1	0	0	0	0	1	0	1	0	0	1	0	0	1	0	⋯	1	0
0	0	1	0	1	0	0	1	1	0	1	0	0	1	0	⋯	0	0

(The left brace spans the rows, labelled m.)

Figure 1: A hypothetical population of a UGA with a population of size m and bitstring chromosomes of length n. The 3rd, 5th, and 9th loci of the population are shown in grey.

to conclude that a UGA can attributewise ϵ-approximately correctly solve the learning problem $\langle k = 7, f = \oplus_7, \eta = 1/5 \rangle$ in $O(1)$ queries and $O(n)$ time.

In the second part (relegated to Appendix A) we use recursive three-way majority voting to show that for any $0 < \epsilon < \frac{1}{8}$, if some algorithm $\mathcal{A}$ is capable of attributewise ϵ-approximately correctly solving a learning problem in $O(1)$ queries and $O(n)$ time, then $\mathcal{A}$ can be used in the construction of an algorithm capable of approximately correctly solving the same learning problem in $O(\log^{1.585} n)$ queries and $O(n \log^{1.585} n)$ time. This part of the argument is entirely formal and does not require knowledge of genetic algorithms or statistical hypothesis testing.

5   Symmetry Analysis Based Conclusions

For any integer $m \in \mathbb{Z}^+$, let D_m denote the set $\{0, \frac{1}{m}, \frac{2}{m}, \ldots, \frac{m-1}{m}, 1\}$. Let V be some genetic algorithm with a population of size m of bitstring chromosomes of length n. A hypothetical population is shown in Figure 1.

We define the *1-frequency* of some locus $i \in [n]$ in some generation τ to be the frequency of the bit 1 at locus i in the population of V in generation τ (in other words the number of ones in the population of V at locus i in generation τ divided by m, the size of the population). For any $i \in \{1, \ldots, n\}$, let $\textbf{oneFreq}_{(V,\tau,i)}$ denote the discrete probability distribution over the domain D_m that gives the 1-frequency of locus i in generation τ of V.[3]

---

[3]Note: $\textbf{oneFreq}_{(V,\tau,i)}$ is an unconditional probability distribution in the sense that for any $i \in \{1, \ldots, n\}$ and any generation τ, if $X_0 \sim \textbf{oneFreq}_{(V,0,i)}, \ldots, X_\tau \sim \textbf{oneFreq}_{(V,\tau,i)}$ are random variables that give the 1-frequency of

Let $\phi = \langle n, k, f, K, \eta \rangle$ be an oracle such that the output of the boolean function f is invariant to a reordering of its inputs (i.e., for any $\pi : [n] \to [n]$ and any element $x \in \{0,1\}$, $f(x_1, \ldots, x_k) = f(x_{\pi(1)}, \ldots, x_{\pi(k)})$)) and let W be the genetic algorithm described in algorithm 1 with some population size m, some mutation rate p_m, and that uses ϕ as a fitness function. An appreciation of the algorithmic symmetries in effect (for the purpose at hand, one essential locus is no different from any other essential locus, and a nonessential locus is no different from any other nonessential locus; see Appendix B) yields the following conclusions:

Conclusion 3. $\forall \tau \in \mathbb{Z}_0^+, \forall i,j$ *such that* $i \in K, j \in K, \mathbf{oneFreq}_{(W,\tau,i)} = \mathbf{oneFreq}_{(W,\tau,j)}$

Conclusion 4. $\forall \tau \in \mathbb{Z}_0^+, \forall i,j$ *such that* $i \notin K, j \notin K, \mathbf{oneFreq}_{(W,\tau,i)} = \mathbf{oneFreq}_{(W,\tau,j)}$

For any $i \in K$, $j \notin K$, and any generation τ we define $\mathbf{essential}_{(W,\tau)}$ and $\mathbf{nonessential}_{(W,\tau)}$ to be the probability distributions $\mathbf{oneFreq}_{(W,\tau,i)}$ and $\mathbf{oneFreq}_{(W,\tau,j)}$ respectively. Given the conclusions reached above, these distributions are well defined. A further appreciation of the algorithmic symmetries in effect (the location of the k essential loci is immaterial, each non-essential locus is just "along for the ride" and can be spliced out without affecting the 1-frequency dynamics at other loci) yields the following conclusion:

Conclusion 5. $\forall \tau \in \mathbb{Z}_0^+$, $\mathbf{essential}_{(W,\tau)}$ *and* $\mathbf{nonessential}_{(W,\tau)}$ *are invariant to* n *and* K

6   Statistical Hypothesis Testing Based Conclusions

Fact 6. *Let D be some discrete set. For any subset $S \subseteq D$, and any independent and identically distributed random variables $X_1, \ldots, X_N$, the following statements are equivalent:*

1. $\forall i \in [N], P(X_i \in D \backslash S) \geq \epsilon$

2. $\forall i \in [N], P(X_i \in S) < 1 - \epsilon$

3. $P(X_1 \in S \wedge \ldots \wedge X_N \in S) < (1 - \epsilon)^N$

Let $\oplus_7$ denote the parity function over seven bits, let $\psi = \langle n = 5, k = 7, f = \oplus_7, K = [7], \eta = 1/5 \rangle$ be an oracle, and let G^ψ be the genetic algorithm described in Algorithm 1 with population size 1500 and mutation rate 0.004 that treats ψ as a fitness function. Figures 2a and 2b show the 1-frequency of the first and last loci of G^ψ over 800 generations in each of 3000 runs. Let D_m^* be the set $\{x \in D_m \mid 0.05 < x < 0.95\}$. Consider the following two null hypotheses:

$$H_0^{\mathbf{essential}} : \qquad \sum_{x \in D_{1500}^*} \mathbf{essential}_{(G^\psi, 800)}(x) \geq \frac{1}{8}$$

$$H_0^{\mathbf{nonessential}} : \qquad \sum_{x \in (D_{1500} \backslash D_{1500}^*)} \mathbf{nonessential}_{(G^\psi, 800)}(x) \geq \frac{1}{8}$$

7

Algorithm 1: Pseudocode for a simple genetic algorithm with uniform crossover. The population is stored in an m by n array of bits, with each row representing a single chromosome. RAND() returns a number drawn uniformly at random from the interval [0,1] and RAND(a,b) $< c$ denotes an a by b array of bits such that for each bit, the probability that it is 1 is c.

Input: m: population size
Input: n: number of bits in a bitstring chromosome
Input: τ: number of generations
Input: p_m: probability that a bit will be flipped during mutation

1 pop $\leftarrow$ RAND(m,n) < 0.5
2 **for** $t \leftarrow$ *1 to* τ **do**
3   fitnessVals $\leftarrow$ EVALUATE-FITNESS(pop)
4   **for** $i \leftarrow$*1 to* m **do**
5   totalFitness $\leftarrow$ totalFitness + fitnessVals[i]
6   **end**
7   cumNormFitnessVals[1] $\leftarrow$ fitnessVals[1]
8   **for** $i \leftarrow$*2 to* m **do**
9   cumNormFitnessVals[i] $\leftarrow$ cumNormFitnessVals[$i-1$] +
10   (fitnessVals[i]/totalFitness)
11   **end**
12   **for** $i \leftarrow$ *1 to* $2m$ **do**
13   k $\leftarrow$ RAND()
14   ctr $\leftarrow 1$
15   **while** k $>$ cumNormFitnessVals[ctr] **do**
16   ctr $\leftarrow$ ctr + 1
17   **end**
18   parentIndices[i] $\leftarrow$ ctr
19   **end**
20   crossOverMasks $\leftarrow$ RAND(m,n) < 0.5
21   **for** $i \leftarrow$ *1 to* m **do**
22   **for** $j \leftarrow$ *1 to* n **do**
23   **if** crossMasks[i,j]$= 1$ **then**
24   newPop[i,j]$\leftarrow$ pop[parentIndices[i],j]
25   **else**
26   newPop[i,j]$\leftarrow$ pop[parentIndices[$i+m$],j]
27   **end**
28   **end**
29   **end**
30   mutationMasks $\leftarrow$ RAND(m,n) $< p_m$
31   **for** $i \leftarrow$ *1 to* m **do**
32   **for** $j \leftarrow$ *1 to* n **do**
33   newPop[i,j]$\leftarrow$ XOR(newPop[i,j], mutMasks[i,j])
34   **end**
35   **end**
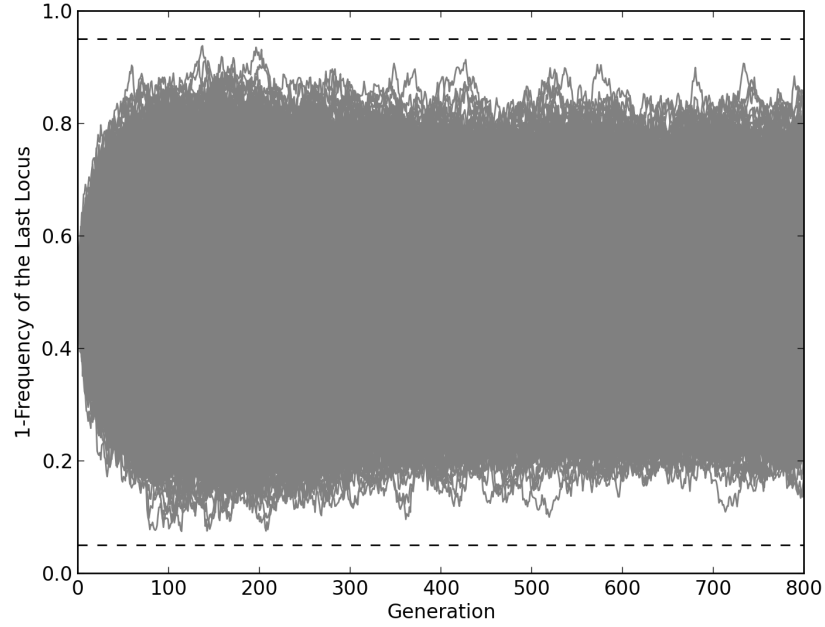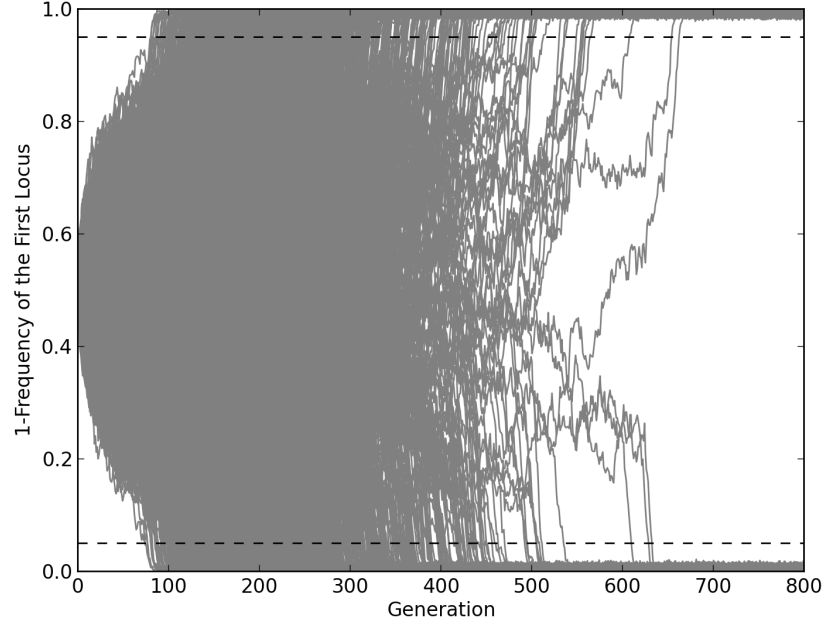36   pop$\leftarrow$newPop
37 **end**

Figure 2: The 1-frequency dynamics over 3000 runs of the first (*top figure*) and last (*bottom figure*) loci of G^ψ, where G is the genetic algorithm described in Algorithm 1 with population size 1500 and mutation rate 0.004, and ψ is the oracle $\langle n = 8, k = 7, f = \oplus_7, K = [7], \eta = 1/5 \rangle$. The dashed lines mark the frequencies 0.05 and 0.95.

9

where **essential** and **nonessential** are as described in the previous section.

We seek to reject $H_0^{\textbf{essential}} \vee H_0^{\textbf{nonessential}}$ at the 10^{-100} level of significance. If $H_0^{\textbf{essential}}$ is true, then for any independent and identically distributed random variables $X_1, \ldots, X_{3000}$ drawn from distribution $\textbf{essential}_{(G^\psi, 800)}$, and any $i \in [3000]$, $P(X_i \in D_{1500}^*) \geq 1/8$. By Lemma 6, and the observation that $D_{1500} \backslash (D_{1500} \backslash D_{1500}^*) = D_{1500}^*$, the chance that the 1-frequency of the first locus of G^ψ will be in $D_{1500} \backslash D_{1500}^*$ in generation 800 in all 3000 runs, as seen in Figure 2a, is less than $(7/8)^{3000}$. Which gives us a p-value less than 10^{-173} for the hypothesis $H_0^{\textbf{essential}}$.

Likewise, if $H_0^{\textbf{nonessential}}$ is true, then for any independent and identically distributed random variables $X_1, \ldots, X_{3000}$ drawn from distribution $\textbf{nonessential}_{(G^\phi, 800)}$, and any $i \in [3000]$, $P(X_i \in D_{1500} \backslash D_{1500}^*) \geq 1/8$. So by Lemma 6, the chance that the 1-frequency of the last locus of G^ϕ will be in D_{1500}^* in generation 800 in all 3000 runs, as seen in Figure 2b, is less than $(7/8)^{3000}$. Which gives us a p-value less than 10^{-173} for the hypothesis $H_0^{\textbf{nonessential}}$. Both p-values are less than a Bonferroni adjusted critical value of $10^{-100}/2$, so we can reject the global null hypotheses $H_0^{\textbf{essential}} \vee H_0^{\textbf{nonessential}}$ at the 10^{-100} level of significance. We are left with the following conclusions:

Conclusion 7.

$$\sum_{x \in D_{1500}^*} \textbf{essential}_{(G^\phi, 800)}(x) < \frac{1}{8}$$

Conclusion 8.

$$\sum_{x \in (D_{1500} \backslash D_{1500}^*)} \textbf{nonessential}_{(G^\phi, 800)}(x) < \frac{1}{8}$$

Algorithm 2: $\mathcal{G}^\phi$

---

1 $pop \leftarrow$ population of G^ϕ after 800 generations
2 for $i \leftarrow 1$ *to* n **do**
3   **if** $0.05 <$ *1-frequency of locus* i *in pop* < 0.95 **then**
4 $\mathrm{x}[r][i] = 0$
5   **else**
6 $\mathrm{x}[r][i] = 1$
7   **end**
8 end

Conclusion 9. *The learning problem* $\langle k = 7, f = \oplus_7, \eta = 1/5 \rangle$ *can be attributewise* $\frac{1}{8}$*-approximately correctly solved in* $O(1)$ *queries and* $O(n)$ *time.*

Argument. For any oracle $\phi = \langle n, k = 7, f = \oplus_7, K, \eta = 1/5 \rangle$ with target concept c, let h be the hypothesis returned by the algorithm $\mathcal{G}^\phi$ shown in Algorithm 2. It is easily seen that $\mathcal{G}^\phi$ runs in $O(1)$ queries and $O(n)$ time; by Conclusions 7 and 8, for any $i \in [n]$, $P(c_i \neq h_i) < \frac{1}{8}$

$\hfill \square$

That $\langle k = 7, f = \oplus_7, \eta = 1/5 \rangle$ is approximately correctly learnable in $O(\log^{1.585} n)$ queries and $O(n \log^{1.585} n)$ time follows from the conclusion above and from Theorem 14 in the Appendix.

---

locus i in generations $0, \ldots, \tau$ of V in some run. Then $P(X_\tau \mid X_{\tau-1}, \ldots, X_0) = P(X_\tau)$

7 Discussion and Conclusion

We used an empirico symmetry analytic proof technique with a 1 in 10^{100} chance of error to conclude that a genetic algorithm with uniform crossover can be used to solve the noisy learning parities problem $\langle k = 7, f = \oplus_7, \eta = 1/5 \rangle$ in $O(\log^{1.585} n)$ queries and $O(n \log^{1.585} n)$ time. These bounds are marginally higher than the known optimal bounds for a more difficult version of the problem in which the oracle is queried *non-adaptively*: $O(\log n)$ and $O(n)$ respectively [6, 8]. Tighter bounds on the query complexity than the one obtained can be achieved using recursive majority voting with a higher branching factor (e.g. 5 instead of 3). However, for our purposes the bounds obtained suffice. The finding that a genetic algorithm with uniform crossover can straightforwardly be used to obtain close-to-optimal bounds on the queries and running time required to solve to solve $\langle 7, \oplus_7, 1/5 \rangle$ is wholly unexpected and lends support to the hypothesis that implicit concurrent multivariate effect evaluation powers general-purpose, non-local, noise-tolerant optimization in genetic algorithms with uniform crossover.

References

[1] Keki Burjorjee. Sufficient conditions for coarse-graining evolutionary dynamics. In *Foundations of Genetic Algorithms 9 (FOGA IX)*, 2007.

[2] Keki M. Burjorjee. Explaining optimization in genetic algorithms with uniform crossover. In *Proceedings of the twelfth workshop on Foundations of genetic algorithms XII*, FOGA XII '13, pages 37–50, New York, NY, USA, 2013. ACM.

[3] T. H. Cormen, C. H. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[4] Daniel C Dennett. *Intuition Pumps and Other Tools for Thinking*. WW Norton & Company, 2013.

[5] L.J. Eshelman, R.A. Caruana, and J.D. Schaffer. Biases in the crossover landscape. *Proceedings of the third international conference on Genetic algorithms table of contents*, pages 10–19, 1989.

[6] Vitaly Feldman. Attribute-efficient and non-adaptive learning of parities and dnf expressions. *Journal of Machine Learning Research*, 8(1431-1460):101, 2007.

[7] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[8] Thomas Hofmeister. An application of codes to attribute-efficient learning. In *Computational Learning Theory*, pages 101–110. Springer, 1999.

[9] E.T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2007.

[10] Lukas Kroc, Ashish Sabharwal, and Bart Selman. Survey propagation revisited. In Ronald Parr and Linda C. van der Gaag, editors, *UAI*, pages 217–226. AUAI Press, 2007.

[11] Elitza Maneva, Elchanan Mossel, and Martin J. Wainwright. A new look at survey propagation and its generalizations. *J. ACM*, 54(4), July 2007.

[12] M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

[13] Melanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, MA, 1996.

[14] Karl Popper. *Conjectures and Refutations*. Routledge, 2007.

[15] Karl Popper. *The Logic Of Scientific Discovery*. Routledge, 2007.

[16] C.R. Reeves and J.E. Rowe. *Genetic Algorithms: Principles and Perspectives: a Guide to GA Theory*. Kluwer Academic Publishers, 2003.

[17] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, 26:521–532, 1993.

[18] Uehara, Tsuchida, and Wegener. Identification of partial disjunction, parity, and threshold functions. *TCS: Theoretical Computer Science*, 230, 2000.

Appendix

A   Formal Analysis

Algorithm 3: RECURSIVE-3-WAY-MAJ$(\ell, (x_1, \ldots, x_{3^\ell}))$

---

1 **if** $\ell = 1$ **then**
2   **return** $M(x_1, x_2, x_3)$
3 **else**
4      $y_1 = $ RECURSIVE-3-WAY-MAJ$(\ell - 1, (x_1, \ldots, x_{3^{\ell-1}}))$
5      $y_2 = $ RECURSIVE-3-WAY-MAJ$(\ell - 1, (x_{3^{\ell-1}+1}, \ldots, x_{2 \times 3^{\ell-1}}))$
6      $y_3 = $ RECURSIVE-3-WAY-MAJ$(\ell - 1, (x_{2 \times 3^{\ell-1}+1}, \ldots, x_{3^\ell}))$
7   **return** $M(y_1, y_2, y_3)$
8 **end**

Let $M : \{0,1\}^3 \to \{0,1\}$ denote the three way majority function that returns the mode of its three arguments. So, $M(1,1,1) = M(0,1,1) = M(1,0,1) = M(1,1,0) = 1$, and $M(0,0,0) = M(1,0,0) = M(0,1,0) = M(0,0,1) = M(0,0,0) = 0$.

Lemma 10. *Let X_1, X_2, X_3 be independent binary random variables such that for any $i \in \{1,2,3\}$, $P(X_i = 0) < \epsilon$. Then $P(M(X_1, X_2, X_3) = 0) < 4\epsilon^2$*

Proof:

$$
\begin{aligned}
P(M(X_1, X_2, X_3) = 0) &= P(X_1 = 0 \wedge X_2 = 0 \wedge X_3 = 0) + \\
&\quad P(X_1 = 1 \wedge X_2 = 0 \wedge X_3 = 0) + \\
&\quad P(X_1 = 0 \wedge X_2 = 1 \wedge X_3 = 0) + \\
&\quad P(X_1 = 0 \wedge X_2 = 0 \wedge X_3 = 1) \\
&< \epsilon^3 + 3\epsilon^2 \\
&< 4\epsilon^2
\end{aligned}
$$

Consider Algorithm 3.

Theorem 11 (Recursive 3-way majority voting)**.** *For any $\epsilon > 0$ and any $\ell \in \mathbb{Z}^+$, let $X_1, \ldots, X_{3^\ell}$ be independent binary random variables such that $\forall i \in [3^\ell], P(X_i = 0) < \epsilon$. Then $P($RECURSIVE-3-WAY-MAJ$(\ell, (X_1, \ldots, X_{3^\ell})) = 0) < 4^{2^{\ell-1}} \epsilon^{2^\ell}$*

Proof. The proof is by induction on ℓ. The base case, when $\ell = 1$, follows from lemma 10. We assume the inductive hypothesis for $\ell = k$ and prove it for $\ell = k + 1$. Let $Y_1, \ldots, Y_3$ be random binary variables that give the values of y_1, y_2, y_3 in the first (top-level/non-recursive) pass of Algorithm. Three applications of the inductive hypothesis for $\ell = k$ gives us that $\forall i \in \{1,2,3\}$, $P(Y_i = 0) < 4^{2^k-1} \epsilon^{2^k}$. Since $X_1, \ldots, X_{3^{k+1}}$ are independent, $Y_1, \ldots, Y_3$ are also independent. So,

13

by Lemma 10,

$$P(\text{RECURSIVE-3-WAY-MAJ}(\ell, (X_1, \ldots, X_{3^{k+1}})) = 0) < 4\left(4^{2^k - 1}\epsilon^{2^k}\right)^2$$

$$= 4\left(4^{2^k - 1}\right)^2 \left(\epsilon^{2^k}\right)^2$$

$$= 4\left(4^{2(2^k - 1)}\right)\left(\epsilon^{2(2^k)}\right)$$

$$= 4\left(4^{2^{k+1} - 2}\right)\left(\epsilon^{2^{k+1}}\right)$$

$$= 4^{2^{k+1} - 1}\epsilon^{2^{k+1}}$$

$\square$

Corollary 12. *For any $\ell \in \mathbb{Z}^+$, let $X_1, \ldots, X_{3^\ell}$ be independent binary random variables such that for any $i \in [3^\ell], P(X_i = 0) < \frac{1}{8}$. Then,*

$P(\text{RECURSIVE-3-WAY-MAJORITY}(\ell, (X_1, \ldots, X_{3^\ell})) = 0) < 1/2^{2^\ell}$

Proof. Follows from Theorem 12 and the observation that

$$4^{2^\ell - 1}\left(\frac{1}{8}\right)^{2^\ell} = 4^{2^\ell - 1}\left(\frac{1}{4}\right)^{2^\ell}\left(\frac{1}{2}\right)^{2^\ell}$$

$$= \frac{1}{4}\left(\frac{1}{2}\right)^{2^\ell}$$

$$< \frac{1}{2^{2^\ell}}$$

$\square$

Noting that 0 and 1 are just labels in the above gives us the following:

Corollary 13. *For any $\ell \in \mathbb{Z}^+$, let $X_1, \ldots, X_{3^\ell}$ be independent binary random variables such that for all $i \in [3^\ell], P(X_i = 1) < \frac{1}{8}$. Then,*

$P(\text{RECURSIVE-3-WAY-MAJORITY}(\ell, (X_1, \ldots, X_{3^\ell})) = 1) < 1/2^{2^\ell}$

Theorem 14. *For any $0 < \epsilon < 1/8$, if the learning problem $\langle k, f, \eta \rangle$ can be attributewise ϵ-approximately correctly solved in $O(1)$ queries and $O(n)$ time, then $\langle k, f, \eta \rangle$ can be approximately correctly solved in $O(\log^{1.585} n)$ queries and $O(n \log^{1.585} n)$ time*

Proof Let $\mathcal{A}$ be an algorithm that attributewise ϵ-approximately correctly solves $\langle k, f, \eta \rangle$ in $O(1)$ queries and $O(n)$ time. Let $\phi = \langle n, k, f, K, \eta \rangle$ be some oracle. Consider Algorithm 4 that executes $\ell = \lceil \log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) \rceil$ runs of the algorithm $\mathcal{A}^\phi$. Let c be the target concept, and for any $i \in [n]$, let H_i be a binary random variable that gives the value of h[i] (set in line 6), and let H be the random variable that give the value of h.

Claim 15. *For all $i \in [n]$, $P(H_i \neq c_i) < 1/2^{2^\ell}$.*

14

Algorithm 4: $\mathcal{B}^\phi(n, \epsilon)$

   Input: n

   Input: ϵ

1 $\ell \leftarrow \lceil \log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) \rceil + 1$

2 for $r \leftarrow 1$ *to* 3^ℓ **do**

3 | $\mathrm{x}[r] = \mathcal{A}^\phi(n)$

4 end

5 for $i \leftarrow 1$ *to* n **do**

6 | $\mathrm{h}[i] = \textsc{Recursive-3-Way-Majority}(\ell, (\mathrm{x}[0][i], \ldots, \mathrm{x}[3^\ell][i]))$

7 end

8 return h

Proof of Claim 15. For any $r \in [3^\ell]$, let $X_{(r,i)}$ be a binary random variable that gives the value of $\mathrm{x}[r][i]$ where $\mathrm{x}[r]$, set in line 3, is the hypothesis returned by $\mathcal{A}$ in run r. Clearly, $X_{(1,i)}, \ldots, X_{(3^\ell,i)}$ are independent. The claim follows from Corollaries 12 and 13, and the premise that $\langle 7, \oplus_7, 1/5 \rangle$ is attributewise $\frac{1}{8}$-approximately correctly learnable by $\mathcal{A}$. $\qquad\square$

Claim 16. *If* $n/2^{2^\ell} < \epsilon$, *then* $P(H \neq c) < \epsilon$

Proof of Claim 16. The claim follows from Claim 15 and the union bound. $\qquad\square$

Claim 17. $n/2^{2^\ell} < \epsilon$

Proof of Claim 17.

$$
\begin{aligned}
\frac{n}{2^{2^\ell}} &= \frac{n}{2^{2^{\lceil \log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) \rceil + 1}}} \\
&< \frac{n}{2^{2^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon})}}} \\
&= \frac{n}{2^{\log_2 n + \log_2 \frac{1}{\epsilon}}} \\
&= \frac{n}{n/\epsilon} \\
&= \epsilon
\end{aligned}
$$

$\qquad\square$

By claims 16 and 17, $\mathcal{B}$ approximately correctly solves the learning problem $\langle 7, \oplus_7, 1/5 \rangle$. We now consider the query complexity of $\mathcal{B}$. Let $q_\mathcal{A}(n)$, $q_\mathcal{B}(n)$ give the number of queries made by algorithms $\mathcal{A}$ and $\mathcal{B}$ respectively. Likewise, let $t_\mathcal{A}(n)$, $t_\mathcal{B}(n)$ give the running time of algorithms $\mathcal{A}$ and $\mathcal{B}$ respectively. The following two claims complete the proof.

Claim 18. $q_\mathcal{B}(n) \in O(\log^{1.585} n)$:

Claim 19. $t_\mathcal{B}(n) \in O(n \log^{1.585} n)$:

Proof of Claim 18. By the premise of the theorem, there exist constants $n_0, c_{\mathcal{A}}$ such that for all $n \geq n_0$, $q_{\mathcal{A}}(n) \leq c_{\mathcal{A}}$. Thus, for all $n \geq n_0$,

$$
\begin{aligned}
q_{\mathcal{B}}(n) &\leq c_{\mathcal{A}}.3^{\ell} \\
&= c_{\mathcal{A}}.3^{\lceil \log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) \rceil + 1} \\
&\leq c_{\mathcal{A}}.3^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) + 2} \\
&= 9 c_{\mathcal{A}}.3^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon})}
\end{aligned}
$$

Taking logs to the base 2 on both sides gives

$$
\log_2(q_{\mathcal{B}}(n)) \leq \log_2(9 c_{\mathcal{A}}) + \log_2(\log_2 n + \log_2 \frac{1}{\epsilon}). \log_2 3
$$

$$
\Leftrightarrow \quad q_{\mathcal{B}}(n) \leq 9 c_{\mathcal{A}}.(\log_2 n + \log_2 \frac{1}{\epsilon})^{\log_2 3}
$$

$$
\Leftrightarrow \quad q_{\mathcal{B}}(n) \leq 9 c_{\mathcal{A}}.(\log_2 n + \log_2 \frac{1}{\epsilon})^{1.585}
$$

$\square$

Proof of Claim 19. Let $\tau_1(n)$, $\tau_2(n)$ be the time taken to execute lines 1—4 and one iteration of the for loop in lines 5–8 of $\mathcal{B}$, respectively. Clearly, $t_{\mathcal{B}}(n) = \tau_1(n) + n.\tau_2(n)$.

Sub Claim 20. $\tau_1(n) \in O(n \log^{1.585} n)$

Proof of Sub Claim 20. The proof closely mirrors the proof of Claim 18. We include it for the sake of completeness. By the premise of the theorem, there exist constants $n_0, c_{\mathcal{A}}$ such that for all $n \geq n_0$, $t_{\mathcal{A}}(n) \leq c_{\mathcal{A}}$. Thus, for all $n \geq n_0$,

$$
\begin{aligned}
\tau_1(n) &\leq c_{\mathcal{A}}.n.3^{\ell} \\
&= c_{\mathcal{A}}.n.3^{\lceil \log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) \rceil + 1} \\
&\leq c_{\mathcal{A}}.n.3^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) + 2} \\
&= 9 c_{\mathcal{A}}.n.3^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon})}
\end{aligned}
$$

Taking logs to the base 2 on both sides gives

$$
\log_2(\tau_1(n)) \leq \log_2(9 c_{\mathcal{A}}.n) + \log_2(\log_2 n + \log_2 \frac{1}{\epsilon}). \log_2 3
$$

$$
\Leftrightarrow \quad \tau_1(n) \leq 9 c_{\mathcal{A}}.n.(\log_2 n + \log_2 \frac{1}{\epsilon})^{\log_2 3}
$$

$$
\Leftrightarrow \quad \tau_1(n) \leq 9 c_{\mathcal{A}}.n.(\log_2 n + \log_2 \frac{1}{\epsilon})^{1.585}
$$

$\square$

16

Sub Claim 21. $\tau_2(n) \in O(\log^{1.5} n)$

Proof of Sub Claim 21. Observe that $\tau_2(n) \leq T(3^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) + 2})$, where T is given by the following recurrence relation:

$$T(x) = \begin{cases} 3T(x/3) + x & \text{if } x > 3 \\ 1 & \text{if } x = 3 \end{cases}$$

A simple inductive argument (omitted) gives us $T(x) \in O(x \log x)$. Thus,

$$\tau_2(n) \in O(3^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) + 2}(\log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) + 2))$$

$$= O(3^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon})}(\log_2(\log_2 n + \log_2 \frac{1}{\epsilon}) + 2))$$

$$= O(3^{\log_2(\log_2 n + \log_2 \frac{1}{\epsilon})} \log_2(\log_2 n + \log_2 \frac{1}{\epsilon}))$$

$$= O((\log_2 n + \log_2 \frac{1}{\epsilon})(\log_2(\log_2 n + \log_2 \frac{1}{\epsilon})))$$

$$\subset O((\log_2 n + \log_2 \frac{1}{\epsilon})^{1.5})$$

Where the last equation follows from the observation that $\log(x) < \sqrt{x}$ for all positive reals. $\square$

Claim 19 follows from the observation that

$$t_{\mathcal{B}}(n) = \tau_1(n) + \tau_2(n) \Rightarrow t_{\mathcal{B}}(n) \in O(n \log^{1.585} n + n \log^{1.5} n)$$

$$\Rightarrow t_{\mathcal{B}}(n) \in O(n \log^{1.585} n)$$

Where the first implication follows from Sub Claims 20 and 21, and the fact that for any functions $f_1 \in O(g_1), f_2 \in O(g_2)$, we have that $f_1 + f_2 \in O(|g_1| + |g_2|)$ and $f_1.f_2 \in O(g_1.g_2)$ $\square$

B On Our Use of Symmetry

A homologous crossover operation between two chromosomes of length ℓ can be modeled by a vector of ℓ random binary variables $\langle X_1, \ldots, X_\ell \rangle$ from which a crossover mask is sampled. Likewise, a mutation operation can be modeled by a vector of ℓ random binary variables $\langle Y_1, \ldots, Y_\ell \rangle$ from which a mutation mask is sampled. Only in the case of uniform crossover are the random variables $X_1, \ldots, X_\ell$ independent and identically distributed. This absence of *positional bias* [5] in uniform crossover constitutes a symmetry. Essentially, permuting the bits of all chromosomes using some permutation π before crossover, and permuting the bits back using π^{-1} after crossover has no effect on the dynamics of a UGA. If, in addition, the random variables $Y_1, \ldots, Y_\ell$ that model the mutation operator are independent and identically distributed (which is typical), and (more crucially) independent of the value of ℓ, then in the event that the values of chromosomes at some locus i are immaterial during fitness evaluation, the locus i can be "spliced out" without affecting allele dynamics at other loci. In other words, the dynamics of the UGA can be *coarse-grained* [1].

These conclusions flow readily from an appreciation of the symmetries induced by uniform crossover and length independent mutation. While the use of symmetry arguments is uncommon in EC research, symmetry arguments form a crucial part of the foundations of physics and chemistry. Indeed, according to the theoretical physicist E. T. Jaynes "almost the only known exact results in atomic and nuclear structure are those which we can deduce by symmetry arguments, using the methods of group theory" [9, p331-332]. Note that the conclusions above hold true regardless of the selection scheme (fitness proportionate, tournament, truncation, etc), and any fitness scaling that may occur (sigma scaling, linear scaling etc). "The great power of symmetry arguments lies just in the fact that they are not deterred by any amount of complication in the details", writes Jaynes [9, p331]. An appeal to symmetry, in other words, allows one to cut through complications that might hobble attempts to reason within a formal axiomatic system.

Of course, symmetry arguments are not without peril. However, when used sparingly and only in circumstances where the symmetries are readily apparent, they can yield significant insight at low cost. It bears emphasizing that the goal of foundational work in evolutionary computation is not pristine mathematics within a formal axiomatic system, but insights of the kind that allow one to a) explain optimization in current evolutionary algorithms on real world problems, and b) design more effective evolutionary algorithms.