2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Information Transmission Through Genetic Algorithm Fitness Maps

George D. Montañez
Machine Learning Department
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
gmontane@cs.cmu.edu

***Abstract*—To bound the amount of information transmitted from a fitness map to a genetic algorithm population, we use a method suggested by Abu-Mostafa *et al.* [1] for measuring the information storage capacity of general forms of memory and represent the genetic algorithm as a communication channel. Our results show that a number of bits linear in the size of the search space can be stored in a fitness map, but on average only a logarithmic number of bits can be stored within a genetic algorithm population of bounded size and finite precision representation. Our results place an upper bound on the rate at which information can be transmitted through, or generated by and later extracted from, a genetic algorithm under fairly general conditions.**

***Keywords*-information transmission; fitness function; genetic algorithm; channel capacity; populations;**

I. Introduction

Evolution, when viewed as a process of information generation, implies the ability of organisms to appropriate information from their environment to aid in survival of the species [2], [3], [4], [5]. The method by which (fitness) information from the environment is incorporated into the hereditary memory of the species is natural selection, acting on variants within the population [3], [4]. Genetic algorithms [6], [7], [8] are digital programs that mimic evolution in several ways, including the transmission of information from a fitness function ("environment") to the genetic memory of a digital species [9], [10]. The transmission of information through selection is accomplished by affecting the reproductive probabilities among variants, the quantitative details of which can be modeled using Shannon's theory of information [11], [12], [9]. We concern ourselves with quantifying the transmission of information from a fitness function (alternatively, fitness *map*) to a population of organisms, analyzing the process by reducing it to the transmission of messages through a noisy communication channel [11].

By measuring the information capacity of both fitness maps and output populations, we derive an upper bound on the expected rate of information transfer from a fitness map to a population for genetic algorithms of bounded population size. We compare this to the intrinsic (bit-level) information storage capacity of the fitness map and find that a number of bits linear in the size of the instance space can be stored in a fitness map, but, on average, only a logarithmic number of bits can be extracted from the fitness map through the use of a genetic algorithm. These results place an upper bound on both the fitness value of information [13], [14] and the channel capacity of a genetic algorithm population space, since the fitness value of information is bounded from above by Shannon entropy [13], and Shannon entropy itself is upper bounded by the information capacity (see Appendix, Section A).

The results derived here make progress toward discriminating between the amount of information internally generated by a genetic algorithm and the amount of information provided to it by its external fitness map. If we bound the amount of information stored within a fitness map, then any information exceeding that amount can be unambiguously ascribed to the internal workings of the genetic algorithm itself; any informational output below that amount may be the result of information transfer, rather than information creation. Furthermore, by quantifying the channel capacity of output populations directly, one can bound the expected number of bits per population, thus bounding the information transmission through a genetic algorithm as a whole.

II. Related Work

Many researchers have sought to quantify how evolutionary processes transfer information from environments to populations [3], [4], [9], [12], [13], [15], [14], for biological and digital organisms. We briefly review some of the related work in this area.

Spetner [3], [4] pioneered early work in measuring information transmission through natural selection, considering the amount of information created by adaptive mutations when measured against the severity of environmental constraints, thereby quantifying the average information flow from environment to population through single evolutionary events. Kargupta [12] measured information flow through genetic algorithm operators using Shannon's information theory, modeling genetic algorithm operators as noisy communication channels, in a manner strikingly similar to the work presented here.[1] Kargupta's work explores how error-free communication through noisy channels (as studied in information theory) can guide the search for genetic algorithm operators that best balance the trade-off between generating diversity and preserving message fidelity in the face of selection, recombination and mutation. Schneider [9] used information theory to

978-1-4799-0454-9/13/$31.00 ©2013 IEEE

quantify the reduction of Shannon entropy in digital organism genomes, demonstrating how information can quickly arise in a population when first stored in a researcher-selected fitness function and later mined by a genetic algorithm. As the present manuscript demonstrates, fitness functions can serve as large reservoirs of information (linear in the size of the search space), when an appropriate choice of fitness function is made.

Smith [16] provided an overview of how the concept of information has been applied by biologists since the molecular revolution, discussing Shannon information and the symbolic nature of genetic signaling structures and code-bearing molecules.

Bergstrom *et al.* [13] showed that, when applied to populations evolving by natural selection, two different measures of information (Shannon entropy and the decision-theory value of information) can form a single measure called *the fitness value of information*. Furthermore, their work demonstrates that the fitness value of information is bounded above by Shannon entropy, and that even before the function of a biological signal is known, one can place an upper bound on the possible fitness consequences of that signal. In a related paper, Donaldson-Matasci *et al.* [14] further demonstrated the strong connection between the fitness value of information (which takes biological consequences into account) and mutual information (which is oblivious to consequences, function and meaning) in evolving systems that respond to imperfect environmental cues. In a similar vein, a detailed model integrating the fitness value of information and the directionality of information with Shannon entropy and mutual information was developed by Rivoire and Leibler [15].

The current work differs from prior related efforts in a few ways. First, we do not concern ourselves with specific genetic algorithms or genetic operators, allowing for general stochastic population-based processes to be evaluated using our method. Second, the results here apply to both fixed and time-varying fitness functions, in contrast to some prior work (e.g. [3], [9]). Finally, by measuring information properties at the level of populations rather than at the level of individual organisms we are able to quantify information capacities regardless of how individuals within the population are composed (e.g. their chromosome structure, allele frequencies, etc.); this departs from some prior work, though not all (for example, see Rivoire and Leibler [15], who take a similar approach).

III. Preliminaries

We will now review some relevant concepts and define some terminology that will prove useful for deriving our results. The definitions, concepts and material in this section are largely reproduced from [17].

A. Genetic Algorithms

Genetic Algorithms are search and optimization systems inspired by biological evolutionary processes that have the following general structure, roughly following [7]:

1) *Initialize* population of individuals, P_0, where the subscript 0 represents the current time step.

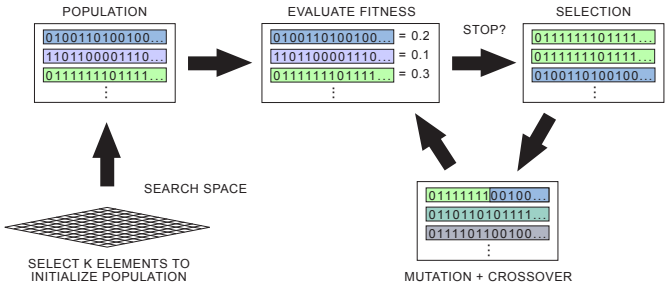

Fig. 1. The control flow structure of a typical genetic algorithm.

2) *Evaluate* initial fitnesses of population (according to **fitness function**.)
3) While the *termination* condition is not met by current population, P_t:
   - *Select* members from P_t for replication (according to **selection operator** that uses fitness evaluations previously calculated), forming temporary population P'_t.
   - *Recombine* members of P'_t (according to **recombination operator**), forming temporary population P''_t.
   - *Mutate* members of P''_t (according to **mutation operator**), forming population P_{t+1}.
   - *Evaluate* fitness of all members of P_{t+1} (according to **fitness function**).

Each of the italicized items represents a point of action where many different methods and parameters can be used. The items in bold represent sub-methods used by the algorithm for carrying out the various procedures. Certain genetic algorithms eschew either recombination or mutation altogether, relying exclusively on one or the other to supply population diversity. The above, therefore, serves only as a general outline of typical design choices for genetic algorithm implementation. Algorithm 1 provides a pseudo-code implementation for the general genetic algorithm described. For those interested, detailed discussion of genetic algorithm implementations can be found in the classic text by Goldberg [6] and the more recent book by Reeves and Rowe [8].

It should be noted that the results derived here in no way depend on the specific implementation of population-based search algorithm used, subject to the conditions that: there is a finite maximum population size, the search space is finite and the composition of a population can only be probabilistically determined by the researcher, otherwise not being directly under his or her control. Most genetic algorithm implementations meet these requirements, and thus the results derived here apply widely.

B. Fitness Functions and Fitness Maps

A *fitness function* is any function $f : \mathcal{X} \to \mathbb{R}$ that maps an individual, $x \in \mathcal{X}$, of a population to a real number [7]. Here, we consider fitness functions that map individuals to binary numbers, $f : \mathcal{X} \to \{0,1\}^l, l \in \mathbb{N}$, and restrict ourselves

Algorithm 1: A Simple Genetic Algorithm

input : Search Space $\mathcal{X}$ and Fitness Function F
output: Population

```
population ← initialize_population(X);
fitnesses ← F(population);
while should_terminate(fitnesses) is False do
    population' ← select_and_replicate(population) ;
    population'' ← recombine(population') ;
    population ← mutate(population'') ;
    fitnesses ← F(population);
end
return population
```

to finite domains of individuals, restrictions which hold for genetic algorithms operating on modern computer hardware.

For our finite domains, a *fitness map* is an exhaustive, uncompressed representation of the mapping between individuals and fitness values generated by a fitness function. It can be defined as the set of all ordered pairs $(i, f(i))$, where i denotes an individual in our domain and $f(i)$ its fitness value.

1) Fitness Map Representations: Finite fitness maps can be represented as a set of point-value pairs (which we denote as *list form*) or as a one-dimensional array of values (*array form*), where position in the array corresponds to the index of an individual. If each individual is encoded using $\log_2 |\mathcal{X}|$ binary digits, then the list form representation ostensibly requires an additional $|\mathcal{X}| \log_2 |\mathcal{X}|$ binary digits to encode compared to array form, since an identifier for each individual must be encoded along with its fitness value. Although the array form appears to require fewer binary digits overall, it assumes knowledge of the ordering used to index individuals. Selecting one ordering from the $|\mathcal{X}|!$ possible requires $O(|\mathcal{X}| \log |\mathcal{X}|)$ bits of information,[2] so both representations are actually equivalent in the number of binary digits required to encode a single map.

We therefore assume an array form representation, so that a finite fitness map is represented as a binary string of length $n = l \times |\mathcal{X}|$, where l is the number of binary digits required to encode each fitness value [18]. We assume a lexicographic ordering of elements, thereby storing the additional $|\mathcal{X}| \log_2 |\mathcal{X}|$ bits of identifier information in the algorithm used to decode the fitness map, and not in the fitness map itself.

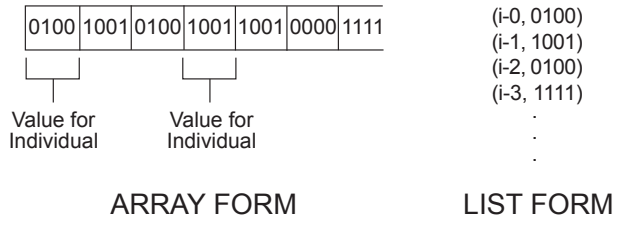

Fig. 2. Two alternative representations of fitness maps.

2) Fitness Value Representations: If the distribution of fitness values in a fitness map is known beforehand, a compression scheme such as Huffman encoding can be used to assign short binary strings to frequently occurring fitness values and longer strings to less frequently occurring values. If distribution information is not available, however, we assume an equiprobable distribution of values, so that each value in the set $\mathcal{Y}$ of possible fitness values has a $\frac{1}{|\mathcal{Y}|}$ probability of occurring [19], [20], [21]. This assumption results in an encoding length of $l = \log_2 |\mathcal{Y}|$ binary digits per value, following Shannon's entropy formula [11].

IV. Information Storage through Object Ranking

Having defined fitness functions and fitness maps, we begin our discussion of information storage in fitness maps by considering how information can be stored using a simple one-dimensional ordering of distinguishable objects. Consider a space of n distinguishable items, under which some canonical ordering holds, such as lexicographical ordering. For concreteness, consider a list of n first and last names, such as "Allen Simon" and "Ronald Mitchell", which can be sorted in alphabetical order. We now consider two people, Alice and Bob, who must communicate a message to one another using nothing but the list of names and a shared knowledge of the encryption method. Alice can communicate one of $n!$ possible messages to Bob, simply by choosing the order in which she lists the n names. Each permutation of values corresponds to a unique message, and since there are $n!$ possible permutations, she can transmit up to $\log_2 n!$ bits of information per message using this communication channel. To decode the message from the list of names, Bob needs to have a mapping from permutations (i.e., possible orderings of the list of the names) to possible messages. This can be done using a look-up table (for small spaces) or an algorithm that transforms list permutations directly into alphabetic strings. (See Appendix A, "Simple Ordinal Communication Channel," of [17] for an implementation of one such algorithm).

Given that the ordering of distinguishable objects can store information, we note that any ranking that assigns real values to distinguishable objects can be used to order a set of objects, either fully (for injective mappings) or partially (for non-injective mappings). Non-injective mappings provide information storage capability bounded by the degeneracy of the mapping, with highly degenerate mappings providing less information storage. Therefore, fitness information as encoded in fitness maps can store information by imposing a partial ordering on the space of individuals in a population. Assigning different fitness values to different individuals serves the purpose of placing an ordering over those individuals, thus allowing an adept Alice to transmit a message through the careful selection of a fitness map. Viewed in this way, the creation of a fitness map becomes the encoding step in message transmission, the genetic algorithm acts as a noisy communication channel, and the examination of an outcome population at time t serves as the decoding step, where the original message is recovered with some probability. Figure 3 illustrates the schematic view of genetic algorithm processes as a communication channel.

With this representation in mind, we will now quantify the amount of information that can be stored in a genetic algorithm fitness map and determine the amount that can be transmitted and extracted at some later time.

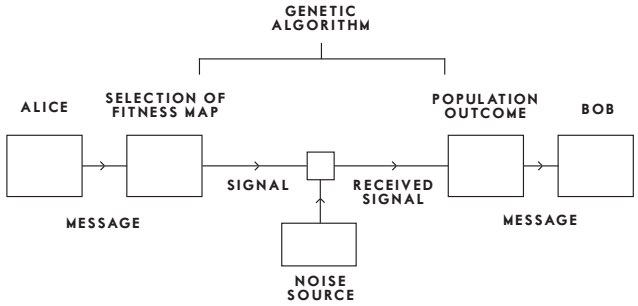

Fig. 3. Genetic algorithm operation as a noisy communication channel. The stochastic nature of the genetic algorithm acts as the noise source. Selecting a fitness map makes some population outcome(s) more likely, but cannot guarantee that the desired population will be received on the terminal end, due to the effects of noise. Examination of the population outcome acts as the decoding step for message transmission. Figure adapted from [11].

V. Information Transmitted from Fitness Maps

Let $f : \mathcal{X} \to \mathcal{Y}$ be a fitness map, represented as a binary string, mapping a space of elements $\mathcal{X}$ to a finite set of fixed-precision numerical values $\mathcal{Y}$. Following Abu-Mostafa and St. Jacques [1], we define the *information capacity*, C, of a memory to be the logarithm of the number of cases it can distinguish between. To assess the information storage capacity of a fitness map and that of an output population, we consider two situations:

1) First, when we have access to the fitness map itself and can directly observe the numerical values of f.
2) Second, when we do not have access to f, but only to a population of k items chosen from $\mathcal{X}$ (with replacement), at time t.

The first situation measures the bit-level information storage capacity of a fitness map, while the second measures the information storage capacity of a finite population, thus providing a means of bounding the quantity of information that can be extracted from a population, and by extension, the maximum expected rate of information transfer through use of a genetic algorithm as a communication channel.

Considering the second situation, we assume the genetic algorithm is stochastic, such that the initial population is randomly selected (and is thus outside of the control of the researcher, i.e. Alice), as is the precise population composition at each time step t. The population composition at each step subsequent to the first is influenced by the fitness map in a probabilistic manner, and thus can be influenced, but not directly controlled, by the researcher. This uncertainty acts as noise within the communication channel.

For Alice to faithfully transmit a message to Bob requires redundancy in the transmission [22], so we assume that Alice is allowed to run the genetic algorithm several times using the same fitness map, allowing Bob to make multiple observations of the final population outcome and average his results. Bob then tries to infer the intended message from the averaged population outcome, which we can represent as the $|\mathcal{X}|$-dimensional average of r vectors, each consisting of $|\mathcal{X}|$ population proportions after t steps on a randomly initialized trial. Formally, let $p_t = [p_1, p_2, \ldots, p_{|\mathcal{X}|}]_t$ be a *population histogram vector*, with $0 \leq p_i \leq 1$ for $i = 1, \ldots, |\mathcal{X}|$, where each $p_i \in \mathbb{R}$ represents the proportion of the population consisting of item $x_i \in \mathcal{X}$ in the final outcome population P_t, and $\sum_i p_i = 1$. We take the limit as t goes to infinity, since during the early steps of the genetic algorithm's operation the population composition may not yet be substantially shaped by the fitness map (reflecting more the randomized initialization than the effects of sustained selection). Putting this together, we define a *population outcome vector*, p^*, as

$$\begin{aligned} p^* &= [p_1^*, p_2^*, \ldots, p_{|\mathcal{X}|}^*] \\ &= \lim_{r\to\infty} \left[\frac{1}{r} \sum_{i=1}^{r} \lim_{t\to\infty} [p_1, p_2, \ldots, p_{|\mathcal{X}|}]_t \right]. \end{aligned} \tag{1}$$

The vector p^* can simply be thought of as the average population outcome for a given fitness map when a genetic algorithm is run for many independent trials using that fitness map. Since more than one fitness map may result in the same $p*$ vector, the mapping from fitness map to averaged population outcome vector (received message) may not be injective. Furthermore, we make no assumption concerning the surjectivity of the mapping from fitness maps to population outcome vectors, so the quantities we derive calculating information capacity C as a function of number of possible population outcomes will represent upper bounds.

For practical purposes, the numbers r and t will be large but finite. Thus, p^* represents an idealization that must be approximated for any realistic algorithm.

A. First Case: Directly Accessing f

If the numerical values of f are directly accessible, and f is encoded as binary string of length n, a total of 2^n different fitness maps can be distinguished. Using the previously given definition of information capacity, the storage capacity of fitness map f is the expected $\log_2(2^n) = n$ bits.

Given that there are $|\mathcal{Y}|^{|\mathcal{X}|}$ unique fitness maps possible given $\mathcal{X}$ and $\mathcal{Y}$

$$\begin{aligned} C &= \log_2(|\mathcal{Y}|^{|\mathcal{X}|}) \\ &\in O(|\mathcal{X}| \log |\mathcal{Y}|). \end{aligned} \tag{2}$$

When $\mathcal{Y}$ is fixed, $\log |\mathcal{Y}|$ becomes a constant and we have $C \in O(|\mathcal{X}|)$.

B. Second Case: Values of f not Directly Observable

In the case where numerical fitness information is not directly available, we must infer fitness information indirectly based on relative reproductive success of items in a population by examining the population outcome vector.

We assume a bounded maximum population size and a finite precision representation of population outcome vectors. Neither assumption involves a loss of generality for genetic algorithms running on finite-state computer hardware, since such digital representations are necessarily of limited precision and operate under finite resource constraints. Using the information capacity formula instead of the usual channel entropy H involves making an implicit assumption that each population outcome is equally likely; if some population outcomes are more likely than others, the channel entropy will decrease and at the extreme, when only one population outcome is possible regardless of fitness map choice, the channel entropy (and information storage capability) reduces to zero bits. Thus, information capacity serves as an optimistic measure of the actual information transmission possible using a genetic algorithm under the conditions outlined.

We now derive two bounds, one for the information capacity of individual population outcomes at time t, and one for the information capacity of population outcome vectors, which are the averages of several population outcome histograms.

1) Information Capacity of Individual Population Outcomes: In observing output populations of size k, there are exactly $\binom{k+|\mathcal{X}|-1}{k}$ unique populations possible, using the combinatoric formula for combinations with repetition. This number of distinguishable cases denotes the number of possible "messages" that can be received using this communication channel at a time t, based solely on counting the number of each variant in an outcome population. Under the assumption of bounded population size, let k be the maximum population size attainable, and let j be the actual size of the outcome population at time t. We then find that

$$
\begin{aligned}
C &= \log_2 \binom{j+|\mathcal{X}|-1}{j} \\
&\leq \log_2 \binom{k+|\mathcal{X}|-1}{k} \\
&= \log_2 \left[\frac{(k+|\mathcal{X}|-1)!}{k!(|\mathcal{X}|-1)!} \right] \\
&= \log_2 \left[\frac{\prod_{i=|\mathcal{X}|}^{k+|\mathcal{X}|-1} i}{k!} \right] \\
&= \log_2 \left[\prod_{i=|\mathcal{X}|}^{k+|\mathcal{X}|-1} i \right] - \log_2(k!) \\
&= \sum_{i=|\mathcal{X}|}^{k+|\mathcal{X}|-1} \log_2(i) - \log_2(k!) \\
&\leq \sum_{i=|\mathcal{X}|}^{k+|\mathcal{X}|-1} \log_2(k+|\mathcal{X}|-1) - \log_2(k!) \\
&= (k+|\mathcal{X}|-1-|\mathcal{X}|+1)\log_2(k+|\mathcal{X}|-1) - \log_2(k!) \\
&= k \log_2(|\mathcal{X}|+k-1) - \log_2(k!) \\
&\leq k\left[\log_2(|\mathcal{X}|) + \log_2(k-1)\right] - \log_2(k!) \qquad (3) \\
&\in O(\log|\mathcal{X}|) \text{ for fixed } k \qquad (4)
\end{aligned}
$$

where inequality (3) follows from Lemma 1, for reasonable values of $|\mathcal{X}|$ and k (see Appendix). The expression $O(\log|\mathcal{X}|)$ is an asymptotic upper bound on C, under the implicit assumption that every possible population outcome has a fitness map capable of producing it. If the mapping from fitness maps to population outcomes is not surjective, then fewer than all possible population outcomes can be produced by the genetic algorithm. Therefore, the bound of $O(\log|\mathcal{X}|)$ is an optimistic upper bound based on favorable assumptions in relation to the definition of information capacity.

This bound optimistically describes the expected amount of information Alice is able store in a population, using her fitness map and the genetic algorithm as the writing mechanism. Thus, it determines the maximum expected rate at which information can be conveyed using a genetic algorithm population, given a population of bounded size. Since only a finite number of unique populations are possible, there is a corresponding finite number of messages that can be received using this communication channel. Thus, taking the logarithm base two of the number of possible messages gives the information capacity of genetic algorithm populations of bounded size. We find that the information capacity C is logarithmic in the size of the instance space $\mathcal{X}$.

2) Information Capacity of Population Outcome Vectors: We now consider population outcome vectors, which are the averages of many population outcome histograms. We note that each element of $\mathcal{X}$ will comprise some portion of the population in the limit, between 0.0 and 1.0, with the portions summing to one. If we assume that this percentage can be known to only a finite precision, as per our assumptions, then the problem becomes one of assigning s individual percentage points among the elements of $\mathcal{X}$, with repetition. For example, if each element can have 0 to 100 shares of the population, we must assign the one hundred shares among the $|\mathcal{X}|$ elements of $\mathcal{X}$ in some fashion. In this case, we must count the number of ways there are to distribute 100 items among $|\mathcal{X}|$ buckets (with repetition, taking order into account), which is $|\mathcal{X}|^{100}$. More generally, there are $|\mathcal{X}|^s$ ways to assign s items among $|\mathcal{X}|$ elements. However, by taking order into account some assignments become repeated in this set, due to the fact that our s items are actually indistinguishable. Therefore, the true number of distinguishable cases (without regard to assignment order) is less than or equal to $|\mathcal{X}|^s$. Taking the logarithm base 2, we find

$$
\begin{aligned}
C &\leq \log_2(|\mathcal{X}|^s) \\
&= s\log_2(|\mathcal{X}|) \\
&\in O(\log|\mathcal{X}|) \text{ for fixed } s \qquad (5)
\end{aligned}
$$

which is an upper bound on the information capacity of population outcome vectors, since the true number of distinguishable cases is generally less than $|\mathcal{X}|^s$.

Thus, we find that the messages output by our communication channel provide at most $O(\log|\mathcal{X}|)$ bits of information per message, whether each message consists of a single population outcome or an averaged population outcome vector. Returning

to our motivating question of how much information can be transmitted from a fitness map to a population, we find that because each population conveys, at most, a logarithmic number of bits on average, the quantity of information that can be successfully transferred to an output population is necessarily limited to the same amount. Thus, genetic algorithms can appropriate no more than $O(\log |\mathcal{X}|)$ bits of information from their fitness maps, on average, per population. Furthermore, this places a logarithmic bound on how much information can be expected to accumulate within a population, irrespective of the length of time during which a genetic process is in operation.

VI. Time-Varying Fitness Maps

Our analysis has thus far focused on bounding the information transmission from a fixed fitness map to an output population, but we now consider the case of time-varying fitness maps as well.

Since our bounds are derived by bounding the maximum number of unique output populations and calculating the information storage capacity given this number of possible populations, we find that introducing time-varying fitness maps does not alter either our bounds or their derivations. Time-varying fitness maps do not produce additional unique output populations, and thus cannot increase the information capacity of a genetic algorithm output population. Our derivations are agnostic concerning the precise processes that produce the final outcome population, under the assumptions that the output population is of bounded size and has a composition not under the direct control of the researcher, both of which hold for stochastic genetic algorithms with time-varying fitness functions implemented on finite-state computer hardware. When calculating the information capacity of an output population, the only relevant detail is how many distinct outcomes are possible, and introducing time-varying fitness maps does nothing to increase this number.

Thus, our logarithmic information bounds hold for both fixed and time-varying fitness maps, covering a large variety of genetic algorithm implementations.

VII. Relevance of Results

To review, our analysis demonstrates that while a fitness map allows for up to $O(|\mathcal{X}|)$ bits of storage, only $O(\log |\mathcal{X}|)$ bits can typically be recovered by a genetic algorithm with a bounded population size and finite precision. We now consider some practical consequences of these results.

First, a large quantity of information is lost in using a fitness map as an information storage device within a genetic algorithm. On average, far less information can be recovered from the population of a genetic algorithm than can be encoded in the fitness map, under the conditions outlined. In choosing (or constructing) a fitness map that transfers information to a population, one must provide a linear quantity of information to recover only a logarithmic amount of information at a later time. From a practical standpoint, this process is inefficient.

Second, we note that the bounds derived here place an upper bound on the rate at which functional [24] or semantic information can be transmitted through a genetic algorithm with bounded population size and fixed precision. This is in agreement with prior work [13], [14] that bounds semantic notions of information (such as the fitness value of information) using Shannon entropy and mutual information.

Third, these results bound not only the rate of information transmission through a genetic algorithm, but also bound the rate at which information can be exported by such an algorithm. Since the final product of the genetic algorithm process is a population, the rate at which information can be output by the genetic algorithm is also logarithmic in the size of the input space. Even for genetic algorithms capable of generating more than a logarithmic number of bits internally, the form of output serves as a bottleneck that restricts the overall rate of export.

Lastly, our analysis urges caution when stating claims concerning the information generation capabilities of various genetic algorithms [9], [23], since demonstrating actual *generation* of information requires one to first identify and account for information stored within the fitness map. If the quantity of information output by the algorithm does not exceed $O(\log |\mathcal{X}|)$ bits, then one cannot determine whether the information was transfered extrinsically from the preexisting storage of the fitness map, or was generated intrinsically by the algorithm itself. Thus, such analysis and careful discrimination should feature in all future research into the information generation capabilities of genetic algorithms.

VIII. Conclusion

To address the question of how much information can be transmitted through a genetic algorithm from a fitness map to an output population, we have represented the genetic algorithm as a noisy communication channel and measured the information storage capacity of both the input (fitness map) and the output (population) of that system. Following the method of Abu-Mostafa *et al.* [1] for measuring the information storage capacity of general forms of memory, we found the information capacity C of fitness maps is linear in the size of the input space, while the information capacity of population outcomes (for populations of bounded size or population vectors of fixed precision) is logarithmic in the size of the input space. These results apply to genetic algorithms using static and time-varying fitness functions.

Thus, under fairly general conditions, the maximum expected number of bits per population is logarithmic in the size of the input space, bounding the information transmission rate for genetic algorithms. Furthermore, these results urge caution and careful analysis when attempting to quantify the information generation capabilities of genetic algorithms, or any other population based algorithm in general, since the information present in an output population may reflect information provided to the algorithm by the researcher's choice of fitness function rather than information generated by the algorithm itself.

APPENDIX
SUPPLEMENTARY MATERIAL

Lemma 1. $\log_2(|\mathcal{X}|+k-1) \leq \log_2(|\mathcal{X}|)+\log_2(k-1)$ *for* $|\mathcal{X}| \geq 2, k = 3, 4, \ldots$

Proof: First we note that, given our conditions on $|\mathcal{X}|$ and k,

$$|\mathcal{X}| \geq \frac{k-1}{k-2},$$

since the expression on the right side asymptotically approaches 1 with increasing k, and achieves a maximum value of 2 when $k = 3$.

Continuing,

$$\begin{aligned}
|\mathcal{X}| &\geq \frac{k-1}{k-2} \\
&= \frac{1-k}{2-k} \\
|\mathcal{X}|(2-k) &\leq 1-k \qquad \text{(since } (2-k)<0\text{)} \\
|\mathcal{X}|(1-(k-1)) &\leq 1-k \\
|\mathcal{X}| - (k-1)|\mathcal{X}| &\leq 1-k \\
|\mathcal{X}| &\leq (k-1)|\mathcal{X}| + 1 - k \\
|\mathcal{X}| &\leq (k-1)|\mathcal{X}| - (k-1) \\
\frac{|\mathcal{X}|}{k-1} &\leq |\mathcal{X}| - 1 \\
\frac{|\mathcal{X}|}{k-1} + 1 &\leq |\mathcal{X}|.
\end{aligned}$$

We next take the logarithm, base 2, of both sides, yielding

$$\begin{aligned}
\log_2\left(\frac{|\mathcal{X}|}{k-1}+1\right) &\leq \log_2(|\mathcal{X}|) \\
\log_2\left(\frac{|\mathcal{X}|}{k-1}+\frac{k-1}{k-1}\right) &\leq \log_2(|\mathcal{X}|) \\
\log_2\left(\frac{|\mathcal{X}|+k-1}{k-1}\right) &\leq \log_2(|\mathcal{X}|) \\
\log_2(|\mathcal{X}|+k-1) - \log_2(k-1) &\leq \log_2(|\mathcal{X}|) \\
\log_2(|\mathcal{X}|+k-1) &\leq \log_2(|\mathcal{X}|) + \log_2(k-1).
\end{aligned}$$

■

A. Shannon Entropy and Information Capacity.

Shannon entropy H is bounded from above by information capacity C. To see this relationship, note that the Shannon entropy over a discrete space of possible populations $\mathcal{P}$ has the form

$$H(\mathcal{P}) = -\sum_{p\in\mathcal{P}} \Pr(p)\log_2\Pr(p),$$

where p is a population, and $H(\mathcal{P})$ is maximized when $\Pr(p) = \frac{1}{|\mathcal{P}|}$. The information capacity C can be re-written as

$$\begin{aligned}
C &= -\log_2\frac{1}{|\mathcal{P}|} \\
&= -\sum_{p\in\mathcal{P}} \frac{1}{|\mathcal{P}|}\log_2\frac{1}{|\mathcal{P}|} \\
&\geq H(\mathcal{P}),
\end{aligned}$$

thereby establishing the relationship.

B. Compression of Fitness Maps[3]

Although we have represented fitness maps as simple binary strings mapping elements of $\mathcal{X}$ to fitness values, this linear representation is rare in genetic algorithms, which compute, rather than store in explicit form, fitness information. The reason is simple: if you have a search space consisting of binary strings of length 512, there are 2^{512} such strings, roughly 10^{154}; yet there are only an estimated 10^{80} atoms in the observable universe. Therefore, we cannot represent such maps in their entirety within computer memory. Generating fitness information allows us to circumvent this difficulty by not having to store fitness information for strings in our space.

Fitness functions form a compressed representation of a fitness map. Take, for example, a Hamming distance fitness function, which calculates the Hamming distance between two strings of any finite length and can be represented as follows

```
hamming(a, b, length):
    difference = 0
    for i from 0 to length:
        difference += (a[i] != b[i])
    return difference
```

This pseudo-code (along with accompanying interpreter code), while not as compact as possible, is orders of magnitude more compressed than an uncompressed fitness map for all strings of length 512, yet computes the same function. This compression raises a difficulty when attempting to measure the amount of information necessary to represent arbitrary fitness mappings; it would appear some fitness maps are greatly compressible and, therefore, require much less than $|\mathcal{X}|\log_2|\mathcal{Y}|$ bits of information to encode.

Although true, this compression is not possible in general, since incompressible strings of every length exist.[4] Therefore, to be able to represent an arbitrary fitness map over elements in $\mathcal{X}$ a minimum number of binary digits is required, which is of order $|\mathcal{X}|\log|\mathcal{Y}|$.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 0946825 and the Ford Foundation Predoctoral Fellowship program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author alone and do not necessarily reflect the views of the National Science Foundation, the Ford Foundation, Carnegie Mellon University or any other organization.

A significant portion of this manuscript is drawn from material first presented (but not yet publicly released) in [17].

The author would like to thank Greg J. Hamerly and Robert J. Marks II of Baylor University for their feedback and comments regarding the contents from which this manuscript was developed, and Cosma Shalizi, for his helpful suggestions concerning related prior work.

NOTES

[1]This similarity was brought to the attention of the author by Cosma Shalizi, after the completion of the present work, but prior to its final publication.

[2]Although a lexicographic ordering of the elements in $\mathcal{X}$ is usually assumed, it is not the only ordering possible for binary strings. Other orderings, such as Binary Reflective Gray codes [8], produce a different interpretation of the positions in the array. Therefore, we must specify which ordering is being used.

[3]The material in this section is largely reproduced from [17].

[4]Although one could imagine a short program outputting a string representing all binary strings of a given length, one must still split the resulting output into individual strings, which can be done in many possible ways. Selecting one particular split from the set of possibilities, therefore, incurs an additional information cost that must be accounted for. Although an extensive investigation of compression is beyond the scope of this manuscript, the simple point that not all strings are compressible suffices here for our purposes.

REFERENCES

[1] Y. Abu-Mostafa and J. St Jacques, "Information capacity of the Hopfield model," *Information Theory, IEEE Transactions on*, vol. 31, no. 4, pp. 461–464, 1985.

[2] H. Simon, "The architecture of complexity," *Proceedings of the American philosophical society*, vol. 106, no. 6, pp. 467–482, 1962.

[3] L. Spetner, "Natural selection: An information-transmission mechanism for evolution," *Journal of Theoretical Biology*, vol. 7, no. 3, pp. 412–429, 1964.

[4] ——, "Information transmission in evolution," *Information Theory, IEEE Transactions on*, vol. 14, no. 1, pp. 3–6, 1968.

[5] A. Klyubin, D. Polani, and C. Nehaniv, "Organization of the information flow in the perception-action loop of evolved agents," in *Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on*. IEEE, 2004, pp. 177–180.

[6] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley, 1989.

[7] T. Bäck and H. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. a, no. 1, pp. 1–23, 1993.

[8] C. Reeves and J. Rowe, *Genetic algorithms: principles and perspectives: a guide to GA theory*. Kluwer Academic Pub, 2002.

[9] T. D. Schneider, "Evolution of biological information," *Nucleic Acids Res.*, vol. 28, no. 14, pp. 2794–2799, 2000.

[10] C. Ofria and C. Wilke, "Avida: A software platform for research in computational evolutionary biology," *Artificial Life*, vol. 10, no. 2, pp. 191–229, 2004.

[11] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[12] H. Kargupta, "Information transmission in genetic algorithm and shannon's second theorem," in *Proceedings of the 5th International Conference on Genetic Algorithms*, S. Forrest, Ed. San Francisco: Morgan Kaufmann, 1993, p. 640. [Online]. Available: http://illigal.org/pub/papers/IlliGALs/93003.ps.Z

[13] C. T. Bergstrom and M. Lachmann, "Shannon information and biological fitness," in *Proceedings of the IEEE Workshop on Information Theory*, 2004. [Online]. Available: http://goo.gl/dIHzf

[14] M. C. Donaldson-Matasci, C. T. Bergstrom, and M. Lachmann, "The fitness value of information," *Oikos*, vol. 119, pp. 219–230, 2010. [Online]. Available: http://goo.gl/c7pn9

[15] O. Rivoire and S. Leibler, "The value of information for populations in varying environments," *Journal of Statistical Physics*, vol. 142, pp. 1124–1166, 2011. [Online]. Available: http://arxiv.org/abs/1010.5092

[16] J. M. Smith, "The concept of information in biology," *Philosophy of science*, pp. 177–194, 2000.

[17] G. Montañez, "Information storage capacity of genetic algorithm fitness maps," Master's thesis, Baylor University, 2011.

[18] T. English, "Optimization is easy and learning is hard in the typical function," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 2. IEEE, 2000, pp. 924–931.

[19] E. Jaynes, "Information theory and statistical mechanics," *Statistical Physics. Brandeis Lectures*, vol. 3, pp. 160–185, 1963.

[20] ——, "Prior probabilities," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 3, pp. 227–241, 1968.

[21] W. Dembski and R. Marks II, "Bernoulli's principle of insufficient reason and conservation of information in computer search," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, oct. 2009, pp. 2647 –2652.

[22] R. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[23] D. Floreano and L. Keller, "Evolution of adaptive behaviour in robots by means of darwinian selection," *PLoS Biol*, vol. 8, no. 1, 2010.

[24] R. M. Hazen, P. L. Griffin, J. M. Carothers, and J. W. Szostak, "Functional information and the emergence of biocomplexity," *Proceedings of the National Academy of Sciences*, vol. 104, no. Suppl 1, pp. 8574–8581, 2007.