The Equilibrium Genetic Algorithm

Ari Juels[*]
RSA Laboratories
20 Crosby Drive
Bedford, MA 01730
tel.: (781) 687-7740
E-mail: ari@rsa.com

June 9, 1998

Abstract

We present a simple mathematical abstraction of the genetic algorithm (GA) which we call the *Equilibrium Genetic Algorithm* (EGA). The EGA provides useful intuitive insight into the mechanics of evolutionary algorithms like the GA. Moreover, the simplicity of the EGA enables us to prove results which would be more difficult to obtain in other models. We prove here that the EGA is able in polynomial time to find the optimum of an elementary non-linear problem which we call MAX_{0s1s} The appeal of the EGA, however, is not strictly theoretical. As we demonstrate through experiments on instances of the widely studied, NP-hard jobshop scheduling problem, the EGA is a competitive optimization tool in its own right, and therefore of potential interest in experimental study.

Key words: genetic algorithm, evolutionary algorithm, jobshop scheduling, optimization

[*]Much of this work was performed at U.C. Berkeley under a UC Regents Fellowship and NSF Grant CCR-9505448

1 Introduction

The problem of rigorously characterizing the behavior of genetic algorithms is a challenging one. The sticking point is the crossover operator: its pairwise ("non-linear") nature yields an algorithm with rather complex behavior, particularly in conjunction with the operations of mutation and selection. The aim of this paper is to address the theoretical challenges of genetic algorithms by investigating a mathematical abstraction which we call the *equilibrium genetic algorithm* (EGA). The results presented here were first elaborated in [15].

The point of embarkation of our approach will be that of Rabinovich, Sinclair, and Wigderson [22]. These authors examine the behavior of the crossover operator in the absence of mutation and selection, investigating its asymptotic properties when it is applied repeatedly to a population of infinite size. Although their results treat a broad range of crossover operators, we shall concern ourselves in this paper exclusively with the special case of k-point or uniform crossover on bitstrings – i.e., sample spaces of form $\{0,1\}^n$. A classical result due to Geiringer [24] characterizes the asymptotic behavior of crossover in this setting. This result states loosely that as crossover is applied repeatedly, bits become mutually independent, so that in the limit, the population assumes a simple and compactly representable form.

We shall consider a genetic algorithm in which the population is of infinite size and always retains the asymptotic form specified by Geiringer. In other words, we shall treat the population as if it is permanently "crossed-over" to convergence or *equilibrium.* This assumption will serve as the basis of the equilibrium genetic algorithm. The equilibrium genetic algorithm is a deterministic algorithm which computes the fitness on all bitstrings in $\{0,1\}^n$. Its behavior may be viewed simply in terms of the trajectory of a point in the interior of an n-dimensional hypercube. Taken as a mathematical and visual model, the EGA will offer a certain degree of intuitive demystification of the behavior of genetic algorithms in general.

Of course, computing the fitness of every bitstring in $\{0,1\}^n$ hardly yields a practical optimization algorithm. For this reason we develop a randomized simulation of the deterministic EGA known as the *stochastic EGA*. Unlike the deterministic algorithm, this form of the EGA may be applied to optimization problems in $\{0,1\}^n$ in a computationally efficient manner. By appealing to the deterministic EGA, it is possible to prove results about the stochastic EGA with greater ease than is normally possible for conventional

GAs.

To date, the only running-time results on GAs have been for linear problems. An important result in this area is that of Baum, Boneh, and Garrett [6], who demonstrate a genetic algorithm which finds the optimum of an n-bit linear function in $O(n \log^2 n)$ time with constant probability. In this paper, we prove that the stochastic EGA is capable of optimizing a simple, *non-linear* function with constant probability in polynomial time. The function in question, referred to as MAX_{0s1s}, outputs the greater of the number of ones and the number of zeroes in a bitstring.

We also conduct an experiment in this paper designed to show that the EGA may be a competitive optimization tool in its own right, and therefore worth of use in experimental study. We examine the performance of the EGA on a widely studied, NP-hard problem known as the jobshop scheduling problem. We find that the performance of the EGA on a standard set of jobshop problem instances is superior to that of a "canonical" GA.

The remainder of this paper is organized as follows. In section 2, we explain the mechanics of optimization on an infinite population, and describe the equilibrium genetic algorithm. We introduce the stochastic EGA in section 3, and prove results about ability to solve the MAX_{0s1s} problem in section 4. Section 5 describes an encoding of the jobshop scheduling problem in $\{0,1\}^n$, and presents an experimental comparison of the EGA and GA on this problem. We offer some concluding remarks in section 6.

2 Infinite populations and the EGA

2.1 Crossover in an infinite population

The finite population maintained by a genetic algorithm is typically specified in the form of a multiset. A population P consisting of four individuals might be described, for instance, by the set $\{111, 000, 000, 000\}$. An alternate way to describe a population, however, is in terms of a probability distribution π on $\{0,1\}^n$: we let $\pi(v)$ denote the probability of obtaining v in a random sampling of P. Thus, the population $P = \{000, 000, 000, 111\}$ may be translated into a distribution in which $\pi(000) = \frac{3}{4}$ and $\pi(111) = \frac{1}{4}$. Observe that the probability distribution π describes the proportions in which individuals are present in the population, but does not specify absolute numbers of individuals. We may therefore think of π as describing a population of *infinite* size.

When an operation such as crossover is applied to a finite population P, the result is a new, randomly-generated population P' (of the same size). Recall that the pairs to be crossed-over in a genetic algorithm are generally selected uniformly at random without replacement. Thus the result of applying 1-point crossover to $P = \{111, 000, 000, 000\}$, for example, will be one of the three following populations P', each with equal probability:

$$\{000, 000, 000, 111\}$$
$$\{000, 000, 100, 011\}$$
$$\{000, 000, 110, 001\}$$

In contrast, the result of applying crossover (or any other operation) to an infinite population π is *deterministic.* Let π' denote the population yielded by applying crossover to π. The probability $\pi'(v)$ on bitstring v in this population is the probability that crossover, when applied to a randomly selected pair from π, yields v. More formally:

$$\pi'(v) = \sum_{abc} \pi(a)\pi(b)\beta_{abcv} \tag{1}$$

where β_{abcv} is the probability that applying crossover to the pair (a, b) yields the pair (c, v). This equation assumes a symmetry condition on the crossover operator in question, namely $\beta_{abcv} = \beta_{abvc}$ for all choices of a, b, c, and v. Both k-point and uniform crossover are easily seen to be symmetric in this sense.

When 1-point crossover is applied to the population π in which $\pi(000) = \frac{3}{4}$ and $\pi(111) = \frac{1}{4}$, by eqn. 1 the resulting population π' is as follows:

$$\begin{array}{lll} \pi'(000) = 5/8 & \pi'(001) = 1/16 & \pi'(011) = 1/16 \\ \pi'(110) = 1/16 & \pi'(100) = 1/16 & \pi'(111) = 1/8 \end{array}$$

Observe that this is not equivalent to a simple condensation to the set of possible populations P' enumerated in the example above. (For example, the fraction of '000' strings among the possible sets P' is $1/2$, while $\pi'(000) = 5/8$.) This difference may be attributed to the fact that pairs are effectively picked *with* replacement in an infinite population.

2.2 Repeated crossover

The population π' resulting when crossover is applied to a population π can have a complex structure, even if π itself may be characterized simply. This is because of the non-linear (pairwise) nature of the crossover operator. On the other hand, repeated application of the crossover operator, as we shall see, causes convergence to an easily characterized distribution. Let $C(\pi)$ denote the population π' induced by applying either k-point crossover (for any k) or uniform crossover to π, and let $C^t(\pi)$ be the result of t successive applications of C to π. A classical result in population genetics due to Geiringer [24] states the following.

Theorem (Geiringer) *Let π_i denote the probability that a sample bitstring u drawn from the distribution π contains a 1 in position i – i.e., that $u_i = 1$. If $w = lim_{t\to\infty}C^t(\pi)$, then for any bitstring v,*

$$w(v) = \prod_{i=1}^{n}(v_i\pi_i + (1 - v_i)(1 - \pi_i)). \tag{2}$$

We refer to the limiting population w as an *equilibrium population.* Informally, Geiringer's theorem states that bits in an equilibrium population are *independent.* In other words, whether a given bitstring drawn from w contains a 1 in position i is independent of whether it contains a 1 in position j, for $i \neq j$. This accords with our intuitive expectation that the crossover operator, by "severing" bitstrings, should weaken correlations among bits in the population.

As a consequence of Geiringer's Theorem, an equilibrium population w may be completely characterized by a vector $\mathbf{w} = <w_1, w_2, \ldots, w_n>$, where $w_i = \pi_i$ is the probability that a bitstring drawn from w has a 1 in position i. If $\mathbf{w} =<0.7, 0.8, 0.3>$, for example, then $w(110) = 0.7 \times 0.3 \times (1 - 0.8) = 0.035$. We refer to $\mathbf{w}$ as an *equilibrium point.* Throughout the remainder of the paper, we shall also let $\mathbf{w}$ stand also for the corresponding equilibrium population.

It is illuminating to view $\mathbf{w}$ as a point in the unit hypercube of n dimensions. The closer $\mathbf{w}$ lies to a vertex corresponding to a given bitstring v, the higher the probability $w(v)$, and vice versa.

Figure 1 depicts the equilibrium point $\mathbf{w}$ =<0.7, 0.3, 0.8>.

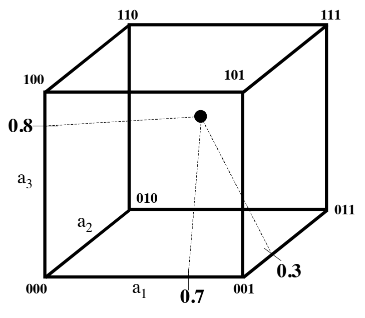

Figure 1. The equilibrium point $\mathbf{w}$ =<0.7, 0.3, 0.8>

2.3 Selection in an infinite population

Like crossover, selection may be viewed as a deterministic operator which, when applied to an infinite population π, yields an infinite population π'. In this case, given finite population P and corresponding infinite population π, there an easily definable relationship between P' and π', the populations yielded by selection on P and π respectively. For a given bitstring v, $\pi'(v)$ is simply the expected number of instances of v in the finite population P'.

Roulette wheel selection [12] is the most commonly used type in didactic descriptions of the GA, although less common in practice. In roulette-wheel selection, a new population P' is generated from P by picking individuals in proportion to their relative fitnesses in the population. The corresponding operation in an infinite population is easily seen to be the following:

$$\pi'(v) = \frac{f(v)}{\overline{f}}\pi(v) \tag{3}$$

where $\overline{f}$ is the weighted average fitness of bitstrings in π. It is easy to verify that π' is a well-defined probability distribution.

A more commonly used type of selection in practice is tournament selection [13]. In k-ary tournament selection, individuals in P' are generated by picking k individuals uniformly at random with replacement from P and discarding all but the fittest one (or an arbitrary fittest individual, if there are multiple ones). Let $\pi_{\leq}(v)$ be the fraction of individuals u in π such that $f(u) \leq f(v)$ and $\pi_{<}(v)$ be the fraction of individuals in u such that $f(u) < f(v)$. Then k-ary tournament selection in an infinite population reduces to the following formula:

$$\pi'(v) = \pi_{\leq}(v)^k - \pi_{<}(v)^k \tag{4}$$

In our construction of the EGA, we shall make use of tournament selection. The reason for this is twofold. First, tournament selection is mathematically simple, particularly when viewed in conjunction with equilibrium populations, as we shall see. Second, tournament selection does not rely on global population characteristics. This is a beneficial quality when dealing with infinite populations, or simulations of infinite populations. Roulette-wheel selection, for instance, requires knowledge of the average fitness $\overline{f}$ over the population. The quantity $\overline{f}$ may be efficiently measured in finite populations, but can effectively only be estimated in infinite populations. In the interest of simplicity, we shall in fact make us of *binary* tournament selection in our construction of the EGA – i.e., tournament selection in which $k = 2$.

2.4 Selection on an equilibrium population: the EGA

We shall construct the equilibrium genetic algorithm using only two operations: crossover and selection. (Mutation may be included, but merely complicates analysis, and, in the case of the EGA, doesn't appear to yield benefits in practice.) In each iteration, binary tournament selection is applied to an equilibrium population π, to yield a new population π', which will not necessarily be in equilibrium. Then, in the interest of obtaining a new equilibrium population, crossover is (loosely speaking) repeatedly applied to π' until convergence to a population π'' . The population π'' will again be in equilibrium, as desired. A single iteration of the EGA may thus be described in two steps that transform a population π into a population π'':

1. Apply binary tournament selection to population π to obtain population π' (not necessarily in equilibrium).

2. Construct population $\pi'' = lim_{t\to\infty} C^t(\pi')$.

By restricting the EGA to operation exclusively on equilibrium populations $\mathbf{w}$, it is possible to condense this description further. Let $f_{i \leftarrow b}$ be a random variable denoting the result of drawing a bitstring v from population $\mathbf{w}$, forcing v_i to b, and applying the objective function f to the resulting bitstring. A single iteration of the deterministic EGA, then, transforms an equilibrium population $\mathbf{w}$ into a new equilibrium population $\mathbf{w}'$ such that:

$$w'_i = w_i^2 + 2(w_i - w_i^2)[z_i] \tag{5}$$

where

$$z_i = Pr[f_{i \leftarrow 1} > f_{i \leftarrow 0}] + \frac{1}{2} Pr[f_{i \leftarrow 1} = f_{i \leftarrow 0}]. \tag{6}$$

Intuitively, the value z_i may be viewed as a global measure in population $\mathbf{w}$ of how much fitter a 1 is in position i than a 0. The EGA seeks to modulate w_i – and hence the proportion of 1s in position i – in accordance with the value of z_i.

Let $\mathbf{w^0}$ denote the equilibrium population at time 0, and $\mathbf{w^t}$ the population after t iterations of the deterministic EGA. Observe that the trajectory $\mathbf{w^0}, \mathbf{w^1}, \mathbf{w^2}, \ldots, \mathbf{w^t}$ is deterministic, and describes a series of points in the unit hypercube of n dimensions. An example of such a trajectory is shown in Figure 2.

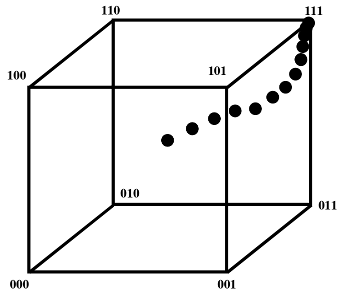

Figure 2. A trajectory of the deterministic EGA

3 The stochastic EGA

The deterministic EGA, as we have seen in the previous section, has a simple mathematical and visual characterization, and is thus a convenient and intuitively natural abstraction of the behavior of genetic algorithms. It is clearly not a feasible optimization tool, however: computing the trajectory of the deterministic EGA requires that we compute the fitness of every bitstring in $\{0,1\}^n$. On the other hand, as we explain in this section, it is possible to simulate the deterministic EGA by means of limited sampling from the population. This simulation, known as the *stochastic EGA*, is an efficiently executable algorithm which, like other GAs, may be studied in the context of combinatorial optimization.

3.1 Description of the stochastic EGA

Let $\mathbf{y}$ denote the equilibrium point maintained by the stochastic EGA in a given iteration. In general, we shall use $\mathbf{y}$ to denote the equilibrium point of a stochastic EGA, and $\mathbf{w}$ to denote that of the deterministic EGA being simulated by the stochastic EGA. A single iteration of the algorithm transforms a population $\mathbf{y}$ into a new population $\mathbf{y}'$ through the following sequence of operations:

1. Sample S elements from distribution $\mathbf{y}$ to obtain a multiset P.
2. Hold $\frac{S}{2}$ binary tournaments in P by selecting pairs uniformly at random without replacement. Let P' be the resulting multiset.
3. Collapse P' into a new equilibrium point $\mathbf{y}'$.

The finite population P' is collapsed to the equilibrium point $\mathbf{y}'$ in this last step by letting y_i be the total fraction of ones in position i in P'.

3.2 How well does the stochastic EGA track the infinite population EGA?

Given a sufficiently large sample size S, the stochastic EGA, as we shall see, provides an accurate simulation of the behavior of the deterministic EGA. To begin with, it is easy to verify that the stochastic EGA provides an *unbiased* simulation of the infinite population. Suppose that $\mathbf{w}$ is the equilibrium point in the deterministic EGA and $\mathbf{y}$ that in the stochastic EGA in a given iteration. Let $\mathbf{w}'$ and $\mathbf{y}'$ be the resulting populations of a single iteration of the respective algorithms. It is easy to see the following:

Claim 1 *The vector $\mathbf{w}'$ is an unbiased estimator of $\mathbf{y}'$. In particular, if $\mathbf{w} = \mathbf{y}$, then $E[y'_i] = w'_i$ for all $1 \leq i \leq n$.*

When S is sufficiently large, a stronger statement than this claim is possible. In particular, we can show that the stepwise behavior of the finite simulation is very likely to be close to that of the deterministic EGA. We shall let the term *w.o.p.* ("with overwhelming probability") denote a probability of $1 - O(n^{-\epsilon})$ for $\epsilon > 0$. Furthermore, we shall make use of the following definition of the L_∞ norm: $\parallel \mathbf{w} - \mathbf{w}' \parallel_\infty = \sum_i w_i - w'_i$. We shall discard the subscript throughout the remainder of this paper, and also abbreviate $\parallel \mathbf{w} - \vec{0} \parallel_\infty$ by $\parallel \mathbf{w} \parallel$.

Lemma 2 *Suppose that $\mathbf{w} = \mathbf{y}$, and that $\parallel \mathbf{w} - \mathbf{w}' \parallel \geq \frac{1}{d}$. Suppose further that $S \geq n^\epsilon d^2$. Then, w.o.p. $\parallel \mathbf{w}' - \mathbf{y}' \parallel < \frac{1}{2d}$.*

Proof. By Claim 1, y'_i may be viewed as the sum of S Bernoulli random variables with mean w'_i. To obtain deviation bounds on this sum, we make use of the following Chernoff-style inequality, which may be found in [1, 17].

Theorem (Chernoff Bound). Let $X_1, X_2, \ldots, X_n$ be independent Bernoulli random variables. Let $X = \frac{1}{n}\sum_{i=1}^{n} X_k$ and p be the expected value of X. Then for $0 < \epsilon < 1$,

1. $pr[X - p \geq \epsilon p] \leq \mathrm{e}^{-\frac{1}{3}\epsilon^2 np}$
2. $pr[X - p \leq -\epsilon p] \leq \mathrm{e}^{-\frac{1}{2}\epsilon^2 np}$. $\square$

The first of these Chernoff-style inequalities implies that $Pr[|y'_i - w'_i| \geq \epsilon w'_i] \leq \mathrm{e}^{-\frac{1}{3}\epsilon^2 w'_i S}$. By letting $\epsilon = \frac{1}{2dw'_i}$, we obtain $Pr[|y'_i - w'_i| \geq \frac{1}{2d}] \leq \mathrm{e}^{-\Omega(n^\epsilon)}$.

Since there are n distinct w'_i, it follows that $Pr[\,\|\,\mathbf{w}' - \mathbf{y}'\,\| > \frac{1}{2d}] \leq n\mathrm{e}^{-\Omega(n^{\epsilon})}$. This proves the lemma. $\square$

As illustrated in Figure 3, the region in which $\mathbf{y}'$ may be found w.o.p. describes a small hypercube around $\mathbf{w}'$. Note that by adjusting the sample size S, the lemma may be modified in such a way to achieve any desired constant size for the length of the sides of the hypercube – e.g., sides of length $\frac{1}{3d}$ instead of $\frac{1}{2d}$.

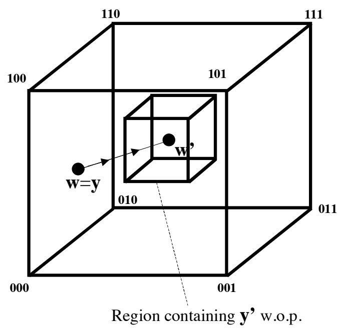

Figure 3. Stepwise tracking of the deterministic EGA

The ability of the stochastic EGA to track the deterministic EGA when S is sufficiently large enables us to construct relatively simple proofs about the behavior of the stochastic EGA on elementary problems. In the next section, we shall show that the stochastic EGA is able to solve a simple, non-linear problem – the MAX_{0s1s} problem – in expected polynomial time with constant probability.

4 The MAX_{0s1s} Problem

Recall that the objective function $f = MAX_{0s1s}$ computes the larger of the number of 1s and the number of 0s in a given bitstring. The function f has two optima: the string $00\ldots0$ and the string $11\ldots1$, which we denote by $\vec{0}$ and $\vec{1}$ respectively. Thus, f is a non-linear function. In particular, to set a given bit optimally, it is necessary to know the values of the other bits in the string. This makes finding the optima of f somewhat more challenging than finding the optima (or optimum) of a simple, linear function.

4.1 Sketch of proof

Our aim now is to show that the stochastic EGA is capable of finding an optimum of f with constant probability in polynomial time. Our proof will begin with by examining the deterministic EGA. We shall initialize the deterministic EGA at a point $\mathbf{w^0}$ lying a small distance from the midpoint of the EGA hypercube – in particular, $\mathbf{w^0}$ will be selected uniformly at random from the small hypercube $[\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon]^n$. We shall show that with constant probability, $\mathbf{w^0}$ is chosen so as to give a "head start" in the direction of the optimum $\vec{1}$: more precisely, we will show that the norm $\parallel \mathbf{w^0} \parallel$ is large. We shall demonstrate further that in later time steps when $\parallel \mathbf{w} \parallel$ is large, the w_i rise appreciably, promoting convergence to $\vec{1}$. Thus the optimum $\vec{1}$ will be sampled w.o.p. in $O(n)$ iterations. shall see.

We turn then consider the stochastic EGA. Recall Lemma 2, which states that the stochastic EGA performs a close, stepwise tracking of the deterministic EGA when the sample size S is sufficiently large. We invoke this lemma to demonstrate that the stochastic EGA with a sample size $S = \Omega(n^{2+\epsilon})$ follows a trajectory like that of the deterministic EGA, and therefore achieves the optimum $\vec{1}$ w.o.p. in O(n) iterations. We shall prove the following theorem:

Theorem 3 *Suppose that the stochastic EGA is initialized to an equilibrium point selected uniformly at random from the hypercube $[\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon]^n$. Then the EGA will find an optimum of the n-bit objective function $f = MAX_{0s1s}$ in time $O(n^{3+\epsilon})$ with probability $\Omega(1)$.*

4.2 Proof of theorem: the deterministic EGA

Let us first examine the behavior of the deterministic EGA. As an initial condition for the success of the algorithm, we shall require that $\parallel \mathbf{w^0} \parallel$ be relatively large. Thus, we make use of the following lemma:

Claim 4 *With probability $\Omega(1)$, $\parallel \mathbf{w^0} \parallel > \frac{n}{2} + 3\sqrt{n}$.*

Proof. Since the w_i^0 range uniformly over $[\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon]$, $\parallel \mathbf{w^0} \parallel = \sum w_i^0$ is the sum of independent, identically-distributed random variables with mean $\frac{1}{2}$. By the Central Limit Theorem, $\parallel \mathbf{w^0} \parallel$ converges to a normal distribution

N with mean $\frac{n}{2}$ and standard deviation $\frac{\epsilon}{\sqrt{3}}\sqrt{n}$ as $n \to \infty$. Since $\frac{n}{2} + 3\sqrt{n}$ lies only a constant number of standard deviations from the mean of N, the lemma follows. $\square$

We will now show that in time steps when the deterministic EGA has $\| \mathbf{w} \| > \frac{n}{2} + 3\sqrt{n}$, the values w_i rise. In order to do this, we shall need some technical lemmas characterizing the distribution of the number of 1s in a bitstring drawn from $\mathbf{w}$. In particular, we shall examine $g^{\neg i}$, which is defined as follows. Let v be a bitstring drawn from $\mathbf{w}$. Then $g^{\neg i}$ is the density function on $\sum_{j \neq i} v_j$, the total number of 1s in v, excluding the i^{th} bit. We shall drop the subscript in $g^{\neg i}$ in our proofs, and simply write g. Since these next few lemmas are subordinate to the structure of the overall proof, the reader may wish to bypass them and skip to Lemma 9. We shall require the following notation: let $g(x) = Pr[g = x]$, $g([x, y]) = Pr[x \leq g \leq y]$, and $\rho_j = \| \mathbf{w} \| + j\sqrt{n}$.

Lemma 5 *For any value of* $\| \mathbf{w} \|$, $g([\rho_{-2}, \rho_2]) > \frac{9}{10}$.

Proof. Since $g = \sum_{j \neq i}^{n} X_j$, where X_j is a Bernoulli random variable with mean w_j, by Chernoff Bound 2, $g([0, \rho_{-2}]) + g([\rho_2, n]) < 2\mathrm{e}^{-2^2} < \frac{1}{10}$. The lemma follows. $\square$

Lemma 6 *The distribution g is unimodal.*

Proof. Let $g_k = \sum_{j=1}^{k} X_j$ (and treat X_i as having mean 0). Hence $g = g_n$. We shall prove the result by induction.

Clearly the lemma holds for $k = 1$. Assume that it holds for $k < n$ – hence g_{n-1} is unimodal. Since $g_j(-1) = 0$ and $g_j(n+1) = 0$ for any j, if there are an $a, b,$ and c such that $a < b < c$ and $g_j(b) < g_j(a)$ and $g_j(b) < g_j(c)$, then g_j is multimodal. Now, $g_n(x) = w_n(g_{n-1}(x-1)) + (1 - w_n)(g_{n-1}(x))$. Therefore, $g_n(b) < g_n(a)$ and $g_n(b) < g_n(c)$ implies:

1. $(g_{n-1}(a) > g_{n-1}(b))$ or $(g_{n-1}(a-1) > g_{n-1}(b-1))$
2. $(g_{n-1}(c) > g_{n-1}(b))$ or $(g_{n-1}(c-1) > g_{n-1}(b-1))$.

Case by case analysis shows that any combination of these conditions implies the multimodality of g_{n-1}, which is a contradiction. Therefore, $g_n = g$ is unimodal. $\square$

Lemma 7 *For any value of* $\parallel \mathbf{w} \parallel$, $g(x) \leq \frac{1}{10\sqrt{n}}$ *for* $x \leq \rho_{-3}$.

Proof. By Lemma 5, $g([\rho_{-2}, \rho_2]) > \frac{9}{10}$. Therefore, there must exist an $x^* \in [\rho_{-2}, \rho_2]$ such that $g(x^*) > \frac{9}{10(4\sqrt{n})} > \frac{1}{10\sqrt{n}}$. Hence, if $g(x) \geq \frac{1}{10\sqrt{n}}$ for $x \leq \rho_{-3}$, then since $g(-1) = 0$, the unimodality of g implies that $g([\rho_{-3}, \rho_{-2}]) > \frac{1}{10}$ – a contradiction of Lemma 5. $\square$

Let us define g' as follows:

$$g' = \sum_{x=\frac{n}{2}}^{n-1} g^2(x) - \sum_{i=0}^{\frac{n}{2}} g^2(x) + \frac{1}{2}[\sum_{x=\frac{n}{2}}^{n-1} g(x)g(x+1) - \sum_{x=0}^{\frac{n}{2}} g(x)g(x+1)]. \quad (7)$$

Lemma 8 *Suppose that* $\parallel \mathbf{w} \parallel > \frac{n}{2} + 3\sqrt{n}$. *Then* $g' = \Omega(\frac{1}{\sqrt{n}})$.

Proof. Let x^* be the smallest value of x on which g is maximal. We shall consider two cases.

Case 1: $\mathbf{g(x^*) > \frac{1}{10}}$ Let $a = g(\frac{n}{2})$. Since $g(x^*) > \frac{1}{10}$ by assumption, Lemma 7 implies that $x^* > \rho_{-3}$. Therefore, by the monotonicity of g, $\sum_{x=\frac{n}{2}}^{n-1} g^2(x)$ and $\sum_{x=\frac{n}{2}}^{n-1} g(x)g(x+1)$ are strictly bounded below by $\frac{a}{10}$. By Lemma 7, $a < \frac{1}{10\sqrt{n}}$, implying, by the monotonicity of g that $g(x) < \frac{1}{10\sqrt{n}}$ for $x \in [0, \frac{n}{2}]$. By Lemma 5, $g([0, \frac{n}{2}]) < \frac{1}{10}$. Combining these last two observations, it is clear that $\sum_{i=0}^{\frac{n}{2}} g^2(x)$ and $\sum_{x=0}^{\frac{n}{2}} g(x)g(x+1)$ are strictly bounded above by $\frac{a}{10}$. The lemma follows.

Case 2: $\mathbf{g(x^*) \leq \frac{1}{10}}$ By Lemma 5, $g([\rho_{-2}, \rho_2]) > \frac{9}{10}$. Since $\parallel \mathbf{w} \parallel > \frac{n}{2} + 3\sqrt{n}$, observe that the interval $[\rho_{-2}, \rho_2]$ is contained in the interval $[\frac{n}{2}, n]$. Since $g(x^*) \leq \frac{1}{10}$, by the monotonicity of g it is easy to see that a lower bound may be obtained on $\sum_{x=\frac{n}{2}}^{n-1} g^2(x)$ and $\sum_{x=\frac{n}{2}}^{n-1} g(x)g(x+1)$ by assuming a uniform distribution on the interval $[\rho_{-2}, \rho_2]$. This may be seen to yield a lower bound of $\frac{1}{5\sqrt{n}}$ on $\sum_{x=\frac{n}{2}}^{n-1} g^2(x)$ and $\sum_{x=\frac{n}{2}}^{n-1} g(x)g(x+1)$. (In the case of the latter quantity, we are ignoring boundary values for the sake of simplicity, but this does not affect our calculations.) Since, as explained for Case 1, $g(x) < \frac{1}{10\sqrt{n}}$ for $x \in [(0, \frac{n}{2}]$ and $g[(0, \frac{n}{2}] < \frac{1}{10}$, it is easy to obtain an upper bound of $\frac{1}{100\sqrt{n}}$ on $\sum_{i=0}^{\frac{n}{2}} g^2(x)$ and $\sum_{x=0}^{\frac{n}{2}} g(x)g(x+1)$. The lemma follows. $\square$

This next lemma now characterizes the stepwise behavior of the deterministic EGA. In particular, it shows that when 1s are prevalent enough in the population – more precisely, when $\|\mathbf{w}\| > \frac{n}{2} + 3\sqrt{n}$ – the w_i will rise in the next time step.

Lemma 9 *If $\|\mathbf{w}\| > \frac{n}{2} + 3\sqrt{n}$, then $w_i' - w_i = \Omega(\frac{1}{\sqrt{n}})(w_i - (w_i)^2)$ for all i.*

Proof. Recall from eqn. 5 that:

$$w_i' = 2(w_i)(1 - w_i)[z_i], \quad \text{where } z_i = Pr[f_{i\leftarrow 1} > f_{i\leftarrow 0}] + \frac{1}{2}Pr[f_{i\leftarrow 1} = f_{i\leftarrow 0}].$$

Observing by symmetry that

$$Pr[f_{i\leftarrow 0} > f_{i\leftarrow 0}] + \frac{1}{2}Pr[f_{i\leftarrow 0} = f_{i\leftarrow 0}] = \frac{1}{2},$$

we can write:

$$\begin{aligned} z_i = \; & \tfrac{1}{2} + \tfrac{1}{2}[Pr[(f_{i\leftarrow 0} = f_{i\leftarrow 0}) \bigwedge (f_{i\leftarrow 1} > f_{i\leftarrow 0})] - Pr[(f_{i\leftarrow 0} = f_{i\leftarrow 0}) \bigwedge (f_{i\leftarrow 1} < f_{i\leftarrow 0})] + \\ & (Pr[(f_{i\leftarrow 0} < f_{i\leftarrow 0}) \bigwedge (f_{i\leftarrow 1} = f_{i\leftarrow 0})] - Pr[(f_{i\leftarrow 0} > f_{i\leftarrow 0}) \bigwedge (f_{i\leftarrow 1} = f_{i\leftarrow 0})]] \end{aligned} \tag{8}$$

For the MAX_{0s1s} problem, algebraic manipulation enables us to rewrite the above as:

$$\begin{aligned} z_i = \; & \tfrac{1}{2} + \textstyle\sum_{x=\frac{n}{2}}^{n-1} (g^{\neg i}(x))^2 - \sum_{i=0}^{\frac{n}{2}} (g^{\neg i}(x))^2 + \\ & \tfrac{1}{2}[\textstyle\sum_{x=\frac{n}{2}}^{n-1} g^{\neg i}(x) g^{\neg i}(x+1) - \sum_{x=0}^{\frac{n}{2}} g^{\neg i}(x) g^{\neg i}(x+1)] \end{aligned} \tag{9}$$

By Lemma 8, it follows that $z_i = \frac{1}{2} + \Omega(\frac{1}{\sqrt{n}})$. Therefore, by eqn. 5:

$$w_i' = \; w_i + \Omega(\tfrac{1}{\sqrt{n}})(w_i - (w_i)^2). \tag{10}$$

□

4.3 Proof of theorem: the stochastic EGA

This last lemma shows that when $\parallel \mathbf{w} \parallel$ is large enough, the equilibrium point of the deterministic EGA will move bitwise in the direction of the optimum $\vec{1}$. By applying this knowledge of the behavior of the deterministic EGA to the stochastic EGA, we are equipped to prove our theorem.

Theorem 10 *Let $\mathbf{y^0}$ be chosen uniformly at random from $[\frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon]$ for some $\epsilon > 0$. The stochastic EGA initialized at $\mathbf{y^0}$ will find an optimum of the n-bit objective function $f = MAX_{0s1s}$ in time $O(n^{3+\epsilon})$ with probability $\Omega(1)$.*

Proof. Let us choose $S = n^{2+\epsilon}$, and consider the first step of the stochastic EGA. With some constant probability c, we will have $\parallel \mathbf{y^0} \parallel \geq \frac{n}{2} + 3\sqrt{n}$. If we let $\mathbf{w^0} = \mathbf{y^0}$, then by the previous lemma, $w_i^1 = w_i^0 + \Omega(\frac{1}{n})(w_i^0 - (w_i^0)^2)$. Since $w_i^0 < \frac{1}{2} + \epsilon$, this means that $w_i^1 - w_i^0 = \Omega(\frac{1}{n})$ for all i. Now, by Lemma 2, since $S = n^{2+\epsilon}$, we will have $|y_i^1 - w_i^1| < \frac{w_i^1 - w_i^0}{2} = \Omega(\frac{1}{n})$ w.o.p. Therefore, $y_i^1 - y_i^0 = \Omega(\frac{1}{n})$ w.o.p. for all i.

By similar work, we can show that w.o.p. for all i, the sequence $y_i^0, y_i^1, \ldots, y_i^t$ will be monotonically increasing, with $y_i^t > 1 - \frac{1}{2n}$ for some $t = O(n)$. It is also easy to see that for all $s = t + poly(n)$, $y_i^s > 1 - \frac{1}{n}$ w.o.p. for all i. Therefore, w.o.p. in time $O(n)$, $y_i > 1 - \frac{1}{n}$ for all i.

When $y_i > 1 - \frac{1}{n}$ for all i, the probability that the bitstring $\vec{1}$ will be obtained in a single sample is $> (1 - \frac{1}{n})^n > \frac{1}{4}$. Therefore, the probability that we will obtain $\vec{1}$ in $S = n^{2+\epsilon}$ samples is $> 1 - (\frac{3}{4})^{n^{2+\epsilon}}$. Thus, $\vec{1}$ is sampled w.o.p.

Since the algorithm locates the optimum $\vec{1}$ in $O(n)$ iterations w.o.p., and the sample size of the stochastic EGA is $S = n^{2+\epsilon}$, the total number of function evaluations will be $O(n^{3+\epsilon})$ w.o.p. This proves the theorem. $\square$

5 The Jobshop Scheduling Problem

An obvious objection to the EGA's suitability as a combinatorial optimization tool is the simplistic structure of the equilibrium population. The foundation of genetic algorithms' optimization ability is their supposed maintenance of a pool of highly fit, overlapping schemata [12]. The EGA, being unable to maintain correlations among bits in its population, cannot include such schemata.

Consider the case of the MAX_{0s1s} problem. A GA might arrive at a population like $\{000, 000, 111, 111\}$, in which only the two optima are represented. The EGA cannot maintain a population with this type of structure: if crossed over to equilibrium, the population $\{000, 000, 111, 111\}$ would become $<\frac{1}{2}, \frac{1}{2}, \frac{1}{2}>$ – the uniform distribution over $\{0, 1\}^n$.

Two questions arise, then. First, is a more complex population structure than that of the equilibrium population necessary for sophisticated optimization tasks? Second, do genetic algorithms in general maintain a population with such complex structure? We will not take up either of these rather involved questions. Instead, we shall confine ourselves to an experimental demonstration of the fact that the EGA may be competitive with a "canonical" GA on hard, naturally arising combinatorial optimization problems.

We wish to choose for our comparison a problem on which the genetic algorithm represents a suitable choice of black-box optimization method. Juels and Wattenberg [16, 15] demonstrated a range of combinatorial optimization problems for which genetic algorithms were inappropriate given the proposed encodings, in the sense that simple hillclimbing algorithms were more effective optimization tools. Here we shall compare the EGA and the GA on the jobshop scheduling problem. At the end of [16, 15] it was demonstrated that a carefully constructed neighborhood structure for this problem renders the genetic algorithm more effective than hillclimbing.

5.1 Jobshop problem description

The jobshop problem is widely studied in the field of management science. It is a notoriously difficult NP-complete problem [11] that is hard to solve even for small instances. A great deal of effort over the course of thirty years has gone into finding efficient approximation algorithms for it. See, for example, [4, 5, 9, 18, 8, 21, 10, 20].

In this problem, a collection of J jobs are to be scheduled on M machines (or processors), each of which can process only one task at a time. Each job is a list of M tasks which must be performed in order. Each task must be performed on a specific machine, and no two tasks in a given job are assigned to the same machine. Every task has a fixed (integer) processing time. The problem is to schedule the jobs on the machines so that all jobs are completed in the shortest overall time. This time is referred to as the *makespan*.

Three instances formulated in [20] constitute a standard benchmark for this problem: a 6 job, 6 machine instance, a 10 job, 10 machine instance, and a 20 job, 5 machine instance. The 6x6 instance is now known to have an optimal makespan of 55. This is very easy to achieve. While the optimum value for the 10x10 problem is known to be 930, this is a difficult problem which remained unsolved for over 20 years [2]. A great deal of research has also been invested in the similarly challenging 20x5 problem, for which an optimal value of 1165 has been achieved, and a lower bound of 1164 [8].

A number of papers have considered the application of GAs to scheduling problems. In particular, Nakano and Yamada [21], Davidor et al. [9], and Fang et al. [10] have described GAs designed to address the three benchmark instances for the jobshop problem. We compare our results with those obtained in Fang et al., one of the more recent of these articles.

5.2 Jobshop problem encoding

Juels and Wattenberg [15, 16] proposed the following encoding of the jobshop scheduling problem. A schedule is encoded in the form of an ordering $\sigma_1, \sigma_2, \ldots, \sigma_{JM}$ of JM markers. These markers have colors associated with them: there are exactly M markers of each color of $1, \ldots, J$. To construct a schedule, σ is read from left to right. Whenever a marker with color k is encountered, the next uncompleted task in job k is scheduled in the earliest plausible time slot. Since there are exactly M markers of each color, and since every job contains exactly M tasks, this decoding of σ yields a complete schedule. Observe that since markers of the same color are interchangeable, many different ordering σ will correspond to the same scheduling of tasks.

Hillclimbing algorithm Juels and Wattenberg first demonstrate a hillclimbing algorithm using this encoding. We present this algorithm as first step in presenting the GA for the problem. To generate a neighbor of σ in this algorithm, a marker σ_i is selected uniformly at random and moved to a new position j chosen uniformly at random. To achieve this, it is necessary to shift the subsequence of markers between σ_i and σ_j (including σ_j) one position in the appropriate direction. If $i < j$, then $\sigma_{i+1}, \sigma_{i+2}, \ldots, \sigma_j$ are shifted one position to the left in σ. If $i > j$, then $\sigma_j, \sigma_{j+1}, \ldots, \sigma_{i-1}$ are shifted one position to the right. (If $i = j$, then the generated neighbor is of course identical to σ.)

Suppose, for example, that $J = 2$ and $M = 3$. One possible ordering σ corresponds to the sequence of colors 111222. (Note that in the above formulation, it is only the colors of the markers that are significant.) If we choose to move σ_1 (here, the first marker of color 1) to position 6, then we obtain the sequence 112221.

GA encoding The basic step in the crossover operator for a GA as applied to a pair (σ, τ) of orderings is as follows. A label i is chosen uniformly at random from $\{1, 2, \ldots, JM\}$. In σ, the marker with label i is moved to the position occupied by i in τ; conversely, the marker with label i in τ is moved to the position occupied by that marker in σ. In both cases, the necessary shifting is performed as before. Hence the idea is to move a single marker in σ (and in τ) to a new position as in the hillclimbing algorithm; instead of moving the marker to a random position, though, we move it to the position occupied by that marker in τ (and σ, respectively). The full crossover operator picks two labels $j \leq k$ uniformly at random from $\{1, 2, \ldots, JM\}$, and performs this basic operation first for label j, then $j+1$, and so forth, through k. (By analogy with the GA above for MDAP.) The mutation operator in our GA performs exactly the same operation as that used to generate a neighbor in the hillclimbing algorithm. A marker σ_i is chosen uniformly at random and moved to a new position j, chosen uniformly at random. The usual shifting operation is then performed. Observe how closely the crossover and mutation operators in this GA for the jobshop problem are based on those in the corresponding hillclimbing algorithm.

Binary encoding Since the EGA operates on bitstrings, however, we must re-encode the jobshop scheduling problem as an optimization task on $\{0, 1\}^n$. Using a trick proposed in [23], we may obtain a neighborhood structure in $\{0, 1\}^n$ which looks substantially like that for the integer-based encoding described above. A solution to a problem instance with J jobs and M machines is encoded as a bitstring v of length JMT, where T is an encoding parameter. We translate v into a schedule as follows. To each marker i, we assign a *tag*, consisting of the i^{th} subsequence of T bits in v. We obtain the sequence of markers $\sigma_1, \sigma_2, \ldots, \sigma_{JM}$ by letting σ_1 be the marker whose tag has the smallest integral value, letting σ_2 be the marker whose tag has the second smallest integral value, and so on. Ties are broken arbitrarily; the parameter T determines the length of v, and consequently the likelihood of ties occurring. (Hence T should be made large enough to avoid an undesirable bias arising in the translation of v into a schedule.) Once the sequence $\sigma_1, \sigma_2, \ldots, \sigma_{JM}$ is obtained, it is translated into a schedule exactly as explained above in this beginning of this subsection.

5.3 The EGA and GA

The EGA In practice, a slight modification to the stochastic EGA used in our analyses in this paper yields a substantially more effective optimization tool. We call this modified EGA the *point-push EGA*. In the stochastic EGA described above, S binary tournaments are performed and the $\frac{S}{2}$ victors of these tournaments are collapsed into a new equilibrium point. In the point-push EGA, S bitstrings are selected at random from the distribution specified by $\mathbf{y}$, and the *fittest* of these S bitstrings, v^* is then determined (with ties broken randomly). The new equilibrium point is obtained by pushing $\mathbf{y}$ some fractional distance δ in the direction of v^*, where δ is an implementation parameter. Letting $max_f(P)$ denote the set of elements of P which maximize the objective function f, we may describe the point-push EGA by the following piece of pseudocode:

```
LET $\mathbf{y} = <\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}>$;
DO FOR $t$ ITERATIONS
        SELECT A SET $P$ OF $S$ BITSTRINGS FROM $\mathbf{y}$;
        SELECT $v^*$ AT RANDOM FROM $\max_f(P)$;
        LET $\mathbf{y} = \mathbf{y} + \delta(\mathbf{y} - v^*)$;
```

We let $\delta = \frac{1}{20}$ in our experiments, a setting which appears to yield good results over a broad range of problems.

Remarks. It should be noted that Theorem 10 applies also to the point-push version of the EGA. The selection mechanism used in the point-push EGA may be viewed as a form of S-ary tournament selection. Therefore, the following claim, when traced through the proof of Theorem 10 shows that the theorem applies to the point-push EGA for any constant $\delta > 0$. $\square$

Claim 11 *Suppose that for some objective function f and some equilibrium point $\mathbf{w}$, an deterministic EGA employing binary tournament generates equilibrium point $\mathbf{w}'$ in a single time step. Let $\mathbf{w}''$ be the equilibrium point generated by the same EGA employing k-ary tournament selection for some $k > 2$. If $w_i' > w_i$, then $w_i'' > w_i'$. Similarly, if $w_i' < w_i$, then $w_i'' < w_i'$. Finally, $w_i' = w_i$ implies $w_i'' = w_i'$.*

Proof. Suppose that $w'_i > w_i$. Let us assume, for the sake of simplicity, that $k = 2^j$ for some integer j. (The proof is similar for values of k which are not powers of two.) It is possible to hold a k-ary tournament among k bitstrings by recursively holding j successive binary tournaments. Let A_j be probability distribution on bitstrings resulting from the j^{th} round of binary selection, and let u^j_i be the proportion of bitstrings in A_j with a 1 in position i. Clearly, $u^i_1 = w'_i$. Similarly, since $w'_i > w_i$, clearly $z_i > \frac{1}{2}$, so it follows from eqn. 1 that $u^t_i > u^{t-1}_i > \ldots > u^i_1$. Therefore $w''_i > w'_i$. Proofs for the other cases are similar. $\square$

The GA As there are many different forms of GA, and many different parameter settings, we strove as best as possible to create an efficient "canonical" GA, using a collection of parameter settings which are to be more or less standard in the literature.

Our GA included, in order, the following phases: evaluation, elitist replacement, selection, crossover, and mutation. In the evaluation phase, the fitnesses of all members of the population were computed. Elitist replacement substitutes the fittest permutation from the evaluation phase of the previous iteration for the least fit permutation in the current population (except, of course, in the first iteration, in which there is no replacement). We chose to use binary stochastic tournament selection [13]. In this type of selection, P pairs, where P is the size of the population, are selected uniformly at random with replacement from the population. A new population of size P is constituted by selecting the fitter permutation from each of these pairs (with ties broken randomly). The crossover step in our GA selected $P/2$ pairs uniformly at random without replacement from the population and applied the mating operator to each of these pairs independently with probability 0.6. For this probability, referred to generally as the *crossover rate*, a value of 0.6 is fairly standard in the literature. It is, for instance, the default value proposed in Grefenstette's GENESIS software package [14]. In accordance with results in [3] and [19] regarding the optimal *mutation rate*, we set this parameter to $1/n$. More precisely, the number of mutations performed on a given permutation in a single iteration was binomial with parameter $p = 1/n$. The population in our GA was initialized by selecting every individual uniformly at random from the set of all possible jobshop schedules.

5.4 Experimental Results

We conducted experiments on five instances of the jobshop scheduling problem. These include the 10×10 and 20×5 instances given in [20], and explored in [16, 15]. Also considered here are three larger problem instances obtained from the Imperial College Operations Research Library [7]. These are a 15×15 instance, a 20×15 instance, and a 20×20 instance.

We ran each algorithm for 1000 iterations with a population (or sample) size of 100; each run thus included 100,000 function evaluations. We performed 100 runs for each problem instance. The figure below shows the average percentage above the best known upper bound achieved by the two algorithms for each problem instance. Indicated by the vertical lines is the average percentage above the best known upper bound achieved by the hillclimbing algorithm proposed in [16, 15]. This algorithm was executed for 5 repetitions of 20,000 iterations – thus 100,000 iterations in total. We ran the hillclimbing algorithm 100 times on the integer encoding described above.[1]

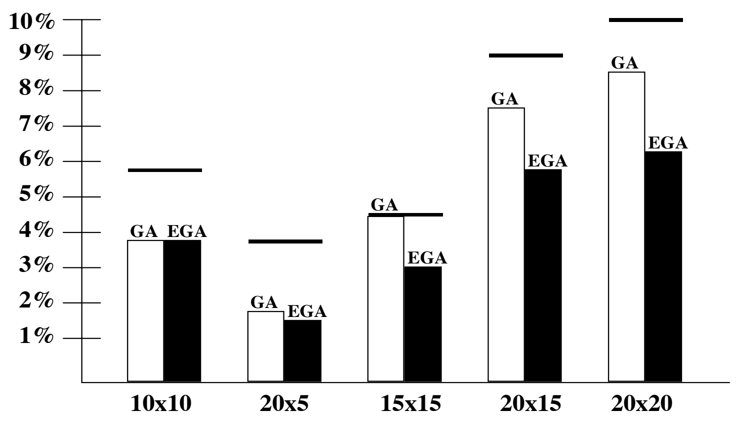

Figure 4. Average percentage above best-known makespan

[1]We applied hillclimbing to the original problem encoding for the following reason. If we had used a hillclimbing algorithm on the new, bitstring-based encoding, a perturbation would have had to consist of a certain number of random bit flips. It is not clear how many such flips would be optimal, nor is it clear how best to perform them. Moreover, we may reasonably expect better results from the original encoding, as it is somewhat more natural to the problem's combinatorial structure.

On all problem instances, the point-push EGA yielded better average results than the GA. The difference in performance was most striking on the larger three instances. Moreover, at the end of most experimental runs, the EGA appeared to be continuing to find improved solutions more rapidly than the GA. We see this in the following graph, depicting the evolution over time of the best makespan achieved by the EGA and GA on the 15×15 jobshop problem.

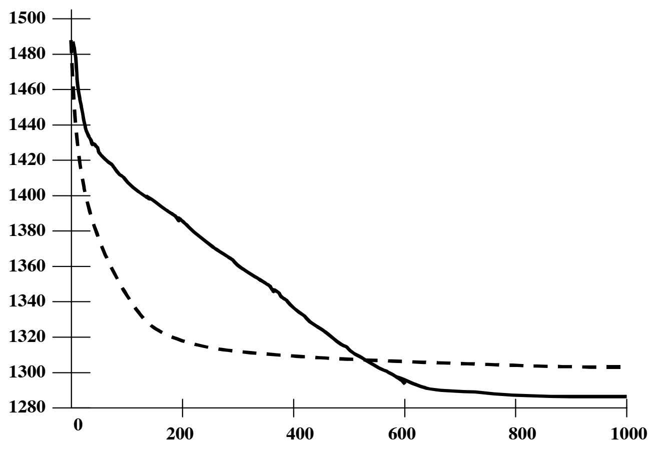

Figure 5. Evolution of best makespan over time

Additional experimental data on the performance of the GA, EGA, and hillclimbing on all five problem instances are available in the appendix of this paper.

6 Concluding remarks and open problems

We have demonstrated how the equilibrium genetic algorithm may be viewed as a simple, mathematical abstraction of conventional genetic algorithms. Using the deterministic version of the EGA, we were able to prove that the computationally efficient stochastic EGA solves the MAX_{0s1s} problem in polynomial time with constant probability. We showed further that on the

NP-hard jobshop scheduling problem, this stochastic EGA is competitive with a "canonical" genetic algorithm, finding solutions of superior average quality on a range of problem instances.

Our exploration in this paper has been confined to the search space $\{0,1\}^n$. Results in [22], however, hold forth the promise of characterizing more sophisticated crossover operators than those on bitstrings. By means of these results, it may be possible to construct EGAs on other, more complex combinatorial search spaces. As many GAs used in practice rely on encodings other than bitstrings, this represents an important line of future research.

Rigorous results on non-linear problems in $\{0,1\}^n$ other than MAX_{0s1s} would still be desirable. Especially appealing would be a proof of the ability of the EGA to solve a non-trivial instance of a naturally-arising combinatorial optimization problem such as maximum cut in polynomial time. A useful tool for proving such results might be a demonstration that the stochastic EGA tracks the deterministic version not only stepwise, but for a significant portion of the latter's trajectory.

There is much room also for experimental investigation of the EGA. One potentially interesting result would be the exhibition of a problem on which the simplistic population structure of the EGA is demonstrably insufficient to make the algorithm an effective optimization tool. Such a result would reveal the distinctive advantages of conventional genetic algorithms as optimization tools. A related result would show how good a model the EGA is in practice – in other words, how much the structure of genetic algorithms' populations resembles that of the EGA. As demonstrated in e.g., [16, 15], however, there still remains for any global optimization method like the GA or EGA the question of how effective it is relative to local search methods on real-world problems.

Appendix

This appendix contains tables detailing the performance statistics of the GA, EGA, and hillclimbing on the five jobshop instances explored in our experiments in section 5.

Problem instance: $J \times M$	*avg.*	*S.D.*	*low*	*high*	Best Known
10x10	965.50	13.38	937	997	930
20x5	1185.92	11.22	1165	1235	1165
15x15	1306.36	20.93	1272	1260	1252
20x15	1484.67	24.28	1437	1576	1381
20x20	1805.92	29.23	1748	1884	1663

Table 1. GA performance on the jobshop scheduling problem

Problem instance: $J \times M$	*avg.*	*S.D.*	*low*	*high*	Best Known
10x10	965.06	12.31	940	982	930
20x5	1183.17	11.73	1165	1219	1165
15x15	1289.79	12.44	1276	1346	1252
20x15	1459.85	14.90	1425	1501	1381
20x20	1767.02	17.85	1715	1807	1663

Table 2. EGA performance on the jobshop scheduling problem

Problem instance: $J \times M$	*avg.*	*S.D.*	*low*	*high*	Best Known
10x10	983.08	19.81	939	1041	930
20x5	1208.60	19.76	1178	1264	1165
15x15	1308.71	21.27	1266	1369	1252
20x15	1504.27	27.41	1443	1580	1381
20x20	1826.04	29.73	1754	1908	1663

Table 3. Hillclimbing performance on the jobshop scheduling problem

References

[1] D. Angluin and L.G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *J. Computer and System Sciences*, 18:155–193, 1979.

[2] D. Applegate and W. Cook. A computational study of the job-shop problem. *ORSA Journal of Computing*, 3(2), 1991.

[3] T. Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 85–94. Elvezier, 1992.

[4] E. Balas. Machine sequencing via disjunctive graphs: An inplicit enumeration algorithm. *Operations Research*, 17:941–857, 1969.

[5] J. Barker and C. McMahon. Scheduling the general jobshop. *Management Science*, 31(5):594–598, 1985.

[6] E.B. Baum, D. Boneh, and C. Garrett. On genetic algorithms. In *COLT '95: Proceedings of the Eight Annual Conference on Computational Learning Theory*, pages 230–239. ACM, 1996.

[7] J.E. Beasley. OR-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[8] J. Carlier and E. Pinson. An algorithm for solving the jobshop problem. *Management Science*, 35(2):164–176, 1989.

[9] Y. Davidor, T. Yamada, and R. Nakano. The ECOlogical framework II: Improving GA performance at virtually zero cost. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 171–176. Morgan Kaufmann, 1993.

[10] H. Fang, P. Ross, and D. Corne. A promising genetic algorithm approach to job-shop scheduling. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1993.

[11] M. Garey and D. Johnson. *Computers and Intractability.* W.H. Freeman and Co., 1979.

[12] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[13] D. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In L.D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 69–93. Morgan Kaufmann, 1991.

[14] J. Grefenstette. A user's guide to GENESIS, 1987.

[15] A. Juels. *Topics in Black-box Combinatorial Optimization.* PhD thesis, U.C. Berkeley, 1996.

[16] A. Juels and M. Wattenberg. Stochastic hillclimbing as a baseline method for the evaluation of genetic algorithms. In M. Hasselmo, editor, *Neural Information Processing Systems (NIPS) 8*, pages 430–436. M.I.T. Press, 1996.

[17] C. McDiarmid. On the method of bounded differences. *London Mathematical Society Lecture Note Series*, 141:148–188, 1989.

[18] G. McMahon and M. Florian. On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research*, 23(3):475–482, 1975.

[19] H. Mühlenbein. How genetic algorithms really work: I. Mutation and hillclimbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 15–25. Elvezier, 1992.

[20] J. Muth and G. Thompson. *Industrial Scheduling.* Prentice Hall, 1963.

[21] R. Nakano and T. Yamada. Conventional genetic algorithm for job shop problems. In Belew and Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 474–479. Morgan Kaufmann, 1991.

[22] Y. Rabinovich and A. Wigderson. An analysis of a simple gneetic algorithm. In Belew and Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 215–221. Morgan Kaufmann, 1991.

[23] C. Shaefer and S. Smith. The Argot strategy II - combinatorial optimization. Technical Report RL90-1, Thinking Machines, 1990.

[24] H. Geiringer von Mises. On the probability theory of linkage in Mendelian heredity. *Annals of Mathematical Statistics*, 15:25–37, 1944.