Darwinian Evolution in Parallel Universes: A Parallel Genetic Algorithm for Variable Selection

Mu ZHU

Department of Statistics and Actuarial Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
(*m3zhu@uwaterloo.ca*)

Hugh A. CHIPMAN

Department of Mathematics and Statistics
Acadia University
Wolfville, Nova Scotia B4P 2R6, Canada

The need to identify a few important variables that affect a certain outcome of interest commonly arises in various industrial engineering applications. The genetic algorithm (GA) appears to be a natural tool for solving such a problem. In this article we first demonstrate that the GA is actually *not* a particularly effective variable selection tool, and then propose a very simple modification. Our idea is to run a number of GAs in parallel without allowing each GA to fully converge, and to consolidate the information from all the individual GAs in the end. We call the resulting algorithm the parallel genetic algorithm (PGA). Using a number of both simulated and real examples, we show that the PGA is an interesting as well as highly competitive and easy-to-use variable selection tool.

KEY WORDS: Akaike information criterion; Bayesian information criterion; Central limit theorem; Combinatorial optimization; Early stopping; Majority vote; Stepwise selection; Stochastic search variable selection.

1. INTRODUCTION

In many industrial applications, engineers are interested in identifying the most important factors that affect a certain outcome of interest, y. In Section 4.4 we describe an example in which engineers are interested in determining how the different input bits affect the conversion error in a digital-to-analog converter (Filliben and Li 1997). Let $\mathcal{C} = \{x_1, x_2, \ldots, x_p\}$ be the set containing a total of p possible factors, which are composed of functions of variables, such as main effects and interaction terms. To understand how the outcome y is affected by these factors, a common approach is to collect a number of observations, say n, and build a multiple regression model,

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i, \qquad i = 1, 2, \ldots, n. \tag{1}$$

When p is large, the model (1) can become rather difficult to interpret. Moreover, it is often the case that only a small number of factors actually affect the outcome. Under such circumstances, it is often desirable to work with a subset of $\mathcal{C}$.

Let Ω be the collection of all subsets of $\mathcal{C}$, for example, $\omega = \{x_1, x_4, x_5\} \in \Omega$. Note that ω can be coded as a binary string of length p; for example, $\omega = \{x_1, x_4, x_5\}$ is the same as writing $\omega = (1, 0, 0, 1, 1, 0, \ldots, 0)$. In this case Ω can be considered the space of all possible such binary strings. The goal of variable selection is to find, in some sense, the "best" $\omega \in \Omega$.

Finding the "best" $\omega \in \Omega$ is a typical *combinatorial optimization* problem. There are altogether $2^p - 1$ nontrivial subsets of $\mathcal{C}$. When $p = 100$, we get $2^{100} - 1 \approx 10^{30}$. Stepwise procedures such as forward selection and backward elimination can be used, but these are greedy methods that can be easily trapped at a local suboptimum. After studying the performances of these stepwise procedures, Miller (2002, sec. 3.13) explicitly recommended carrying out an exhaustive search whenever feasible. Some clever algorithms are available to reduce the amount of computation needed for an exhaustive search. One such well-known algorithm is the branch and bound algorithm. Furnival and Wilson (1974) and Miller (2002, sec. 3.8) discussed the application of the branch and bound algorithm to the variable selection problem. However, even the branch and bound algorithm becomes too expensive for truly large p (e.g., $p > 30$).

To find the "best" subset, we need a working definition of what we mean by the "best." This definition has remained highly controversial; common choices include the Akaike information criterion (AIC) (Akaike 1973), the Bayesian information criterion (BIC) (Schwarz 1978), and the C_p (Mallows 1973), among other alternatives. It is well known (e.g., Kou and Efron 2002) that a solution that is "best" for one criterion (e.g., the AIC) is generally not "best" for a different criterion (e.g., the BIC). In particular, the AIC tends to select more variables than is necessary, whereas the BIC tends to select fewer variables. As such, in evaluating the ultimate performance of a variable selection procedure, we will not be asking whether the AIC or the BIC has been optimized. Instead, we focus on the question of whether the correct variables have been selected.

1.1 The Genetic Algorithm

An interesting heuristic search algorithm well suited for the combinatorial optimization problem is the genetic algorithm (GA) (e.g., Goldberg 1989). Although GA is not widely known among statisticians, it has garnered some interest in this community. Chatterjee, Laudato, and Lynch (1996) gave an introductory review and showed how the GA can be applied to a number of classical problems in statistics.

TECHNOMETRICS, NOVEMBER 2006, VOL. 48, NO. 4
DOI 10.1198/004017006000000093

Table 1. An Illustration of Selection

	1	2	3	4	5	⋯	p		Score
ω_1	1	1	0	0	0	⋯	0		s_1
⋮				⋮					⋮
ω_i	0	0	1	1	1	⋯	1	$\stackrel{F}{\Longrightarrow}$	s_i
ω_{i+1}	1	0	0	1	0	⋯	0		s_{i+1}
⋮				⋮					⋮
ω_m	0	1	0	0	0	⋯	1		s_m

The optimization strategy used by the GA is very simple. Start with a number of randomly generated candidates (the initial population). Using the Darwinian principle of "the survival of the fittest," the weaker (less optimal) candidates are gradually eliminated and the stronger ones are allowed to survive and generate offspring. This goes on for a number of generations until, in the end, good solutions are produced.

In a typical setting, each individual ω is represented by a binary string, say of length p, which is treated as the genetic code (DNA) of ω; each position can be regarded as a single gene. Starting with a randomly generated population of size m, $\{\omega_1, \omega_2, \ldots, \omega_m\}$, a new generation is produced with three genetic operations: selection, reproduction, and mutation.

Selection. Each individual is evaluated by a fitness function, F, often the objective function for the underlying optimization problem, and assigned a score. When the goal is to maximize (minimize) F, those with high (low) scores are given higher likelihoods of surviving to the next generation (Table 1).

Reproduction. Two individuals are selected at random to produce a child. Typically, a cross-over position is chosen at random between 1 and p, say j; the child then inherits the first j genes from the father and the rest $p-j$ genes from the mother. The cross-over position in the illustration is between 3 and 4 (Table 2).

Mutation. At birth, an individual is allowed, with a certain small probability (called the mutation rate), to alter its genetic code at a randomly chosen position. In the typical setting where binary codes are used, this amounts to flipping a 0 to a 1 and vice versa (Table 3).

Remark. For the variable selection problem, each ω_i represents a different subset of variables, and hence a different model and the entire population $\{\omega_1, \omega_2, \ldots, \omega_m\}$ together specifies a total of m different models. Our implementation of the aforementioned principles is described in Sections 1.3 and 3.1.

1.2 An Illustrative Example

To test whether the GA is a good variable selection tool, we first create a very simple simulated dataset consisting of $n = 40$ observations and $p = 20$ variables. The variables are generated independently from standard Gaussian distributions, that is, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{20} \sim \mathrm{N}_{40}(\mathbf{0}, \mathbf{I})$. The model is

$$\mathbf{y} = \mathbf{x}_5 + 2\mathbf{x}_{10} + 3\mathbf{x}_{15} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathrm{N}_{40}(\mathbf{0}, \sigma^2\mathbf{I}), \sigma = 1. \quad (2)$$

In other words, only variables 5, 10, and 15 are actually used in generating the response y.

To obtain a model that has good out-of-sample performance, we choose the leave-one-out cross-validated residual sum-of-squares (CVRSS) as the fitness function for the GA, that is,

$$F(\omega) = -\mathrm{CVRSS}(\omega) = -\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i(\omega)}{1 - h_{ii}}\right)^2. \quad (3)$$

Here h_{ii} is the ith diagonal element of the well-known hat matrix $\mathbf{H} = \mathbf{X}_\omega(\mathbf{X}_\omega^T\mathbf{X}_\omega)^{-1}\mathbf{X}_\omega^T$, and the notation $\hat{y}_i(\omega)$ means the fitted value for the ith observation using a model containing only the variables specified by ω. We discuss the fitness function in more detail in Section 3.4.

Table 4 shows the results from the GA and a number of other commonly used and readily available variable selection techniques. Our implementation of the GA for variable selection is summarized in Table 5, with details given in Sections 1.3 and 3.1. The terms "stepwise AIC" and "stepwise BIC" refer to a stepwise search combining both forward selection and backward elimination (e.g., Venables and Ripley 1994, p. 175) using either the AIC (Akaike 1973) or the BIC (Schwarz 1978) as the objective function. The terms "exhaustive AIC" and "exhaustive BIC" refer to evaluating all possible subsets with either the AIC or the BIC. An exhaustive search over all possible subsets is feasible here using the `leaps` library in `R` (R Development Core Team 2006) because this problem is of only moderate size ($p = 20$).

Unfortunately, our initial excitement in the GA is quickly tamed by this very simple example. Table 4 clearly shows that the GA is not successful in finding the correct model; it finds a model that includes a number of extra spurious variables. Other researchers have also had similar unsuccessful experiences with the GA as a variable selection tool (e.g., Draper and Fouskakis 2000).

Table 2. An Illustration of Reproduction

	1	2	3	4	5	6	7	8
Father	*1*	*1*	*0*	0	0	1	0	0
Mother	0	0	1	*1*	*1*	*0*	*0*	*1*
Child	1	1	0	1	1	0	0	1

Table 3. An Illustration of Mutation

	1	2	3	4	5	6	7	8
Before	0	0	1	1	1	0	0	1
					↓			
After	0	0	1	1	0	0	0	1

Table 4. Results for the Illustrative Example

Method	*Selected variables*
True model	5, 10, 15
Stepwise AIC	1, 4, 5, 6, 7, 9, 10, 12, 14, 15, 16, 17
Stepwise BIC	5, 6, 10, 15
Exhaustive AIC	2, 5, 6, 8, 9, 10, 15
Exhaustive BIC	5, 6, 10, 15
GA	4, 5, 9, 10, 12, 15, 17

Table 5. The Single-Path Genetic Algorithm for Variable Selection

Inputs
- y: $n \times 1$ vector
- $\mathbf{X}$: $n \times p$ matrix
- π: Prior probability used to initialize the GA; default $= .3$
- m: Population size; default $= [p]_e$ (see Sec. 3.1)
- N: Number of generations to evolve (see Sec. 3.2)
- ν_t: Mutation rate for generation t; default $= 1/p$ for every t (see Sec. 3.1)

Algorithm
1. Initialize a random population of size m, $\mathcal{P}(0)$, using the prior probability π; that is, every individual $\omega_i \in \mathcal{P}(t)$ contains, on average, πp 1's and $(1-\pi)p$ 0's.
2. While $(t < N)$:
   a. For every $\omega_i \in \mathcal{P}(t)$, find its fitness score $s_i = F(\omega_i)$, e.g., (3) or (8).
   b. Let the "survival pool," $\mathcal{S}(t)$, be the set of top $m/2$ individuals (see Sec. 3.1), that is, $\mathcal{S}(t) = \{w_i \in \mathcal{P}(t) : s_i > s_{(m/2)}\}$, where $s_{(m/2)}$ is the median of s_i.
   c. Let $\mathcal{P}(t+1) = \mathcal{S}(t)$; notice that, at this point, $|\mathcal{P}(t+1)| = m/2 < m$.
   d. Repeat:
      - Randomly select a *father* and a *mother* from $\mathcal{S}(t)$.
      - Randomly select a cross-over position and produce a *child* (see Table 2).
      - Randomly select a mutation position and flip the genetic code of the *child* at this position with probability ν_t (see Table 3).
      - Insert this *child* into $\mathcal{P}(t+1)$.

      Until $|\mathcal{P}(t+1)| = m$.
   e. Let $t = t+1$.

   End while.

Output

The last generation population, $\mathcal{P}(N)$.

1.3 Implementations of the GA

There are a number of possible reasons why the GA is unsuccessful in the foregoing illustrative example. First, the fitness function (3) could have been a bad choice. Consequently, we also experimented with a number of other well-known criteria commonly used for the variable selection problem, such as the AIC and the BIC. Second, the control parameters in the GA could have been set inappropriately. To obtain the results in Table 4, we used a total of 200 generations, a population size of $m = 20$, and a mutation rate of $1/m = .05$ (see also Table 5). More discussions about how these control parameters are chosen are given in Section 3.

Here we were able to obtain exactly the same solution (i.e., variables 4, 5, 9, 10, 12, 15, and 17) by starting the GA with different initial populations, indicating that the GA had indeed converged with 200 generations, but it was not clear that the other control parameters (e.g., the mutation rate) were set appropriately. It is widely known that the mutation rate is a very important control parameter for the GA; a too-large mutation rate makes it difficult for the algorithm to converge, whereas a too-small mutation rate often causes the algorithm to converge too soon to a suboptimal local solution. One could also use a more sophisticated strategy by using a large mutation rate in the beginning to move around the search space more aggressively and gradually decreasing the mutation rate over time (e.g., Miyata and Shen 2003), much like how one would set the critical temperature parameter in simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983). We experimented with various different ways for setting the mutation rate (see Table 5). In no experiments were we able to correctly identify the true model using the GA.

1.4 Motivation

A few options exist. We can conclude that the GA, although interesting, is not a good tool for variable selection after all. We can try harder to determine how to set various control parameters in the GA and the best fitness function to use. We are not thrilled by either option. Instead, we are inspired by the recent success stories of algorithms such as bagging (Breiman 1996), where, by consolidating a number of relatively simple-minded models, such as a large tree grown to a maximal size without any pruning, one can often do better than a carefully constructed model such as a tree carefully pruned to the right size. Based on such considerations, we propose a novel parallel evolution strategy, which we find to be both interesting and effective.

Herein we sometimes use the term *parallel genetic algorithm* (PGA) to refer to the parallel evolution strategy. In contrast, we use SGA to refer to the more standard single-path GA.

The article is organized as follows. In Section 2 we outline the main idea behind the PGA and how it can be used for variable selection. In Section 3 we give more details of how to implement the PGA. In Section 4 we present a wide variety of examples using both simulated data and data from real industrial experiments. In Section 5 we present a useful dynamic graphical tool for the PGA. We outline a number of future directions in Section 6, and provide some concluding remarks in Section 7.

2. PARALLEL EVOLUTION

The main idea behind parallel evolution is very simple. Instead of running one long evolutionary path, paying careful attention to the choices of various control parameters, we simply create a number of "parallel universes" and run a relatively mindless and short evolutionary path in each of them. We discuss what we mean by "mindless" and "short" in more detail in Section 3.

Let there be a total of B parallel universes. Start a different evolutionary path in each universe $b = 1, 2, \ldots, B$, each for a total of N generations, where N is relatively small. For simplicity, we take the population size in each universe to be fixed at m in every generation. Let

$$\mathcal{P}(b, t) = \text{generation-}t \text{ population in universe } b.$$

The main idea is to define $0 \le r(j, b) \le 1$ as a measure of the importance for variable j based on N generations of evolution in universe b,

$$r(j, b) = \frac{1}{m}\sum_{i=1}^{m} \omega_i(j) \quad \text{for } \omega_1, \omega_2, \ldots, \omega_m \in \mathcal{P}(b, N). \tag{4}$$

In other words, for each universe b, $r(j, b)$ is the percentage of last-generation candidates that contain variable j. The important variables are then selected by taking a majority vote across all universes. That is, we define

$$\bar{r}_j = \frac{1}{B}\sum_{b=1}^{B} r(j, b) \tag{5}$$

and use $\bar{r}_j$ as an importance measure to rank variable j; the top $q < p$ variables are selected. We discuss the choice of q in more detail later (Secs. 2.1 and 3.3). A summary of this very simple parallel evolution algorithm is given in Table 6.

Table 6. The PGA for Variable Selection

Inputs
All inputs to the SGA algorithm [i.e., y, $\mathbf{X}$, π, m, N, ν_t (see Table 5)] and
B: number of parallel paths (see Sec. 3.3)
Algorithm
1. For $b = 1$ to B:
   a. Let $\mathcal{P}(b, N) =$ result from SGA(y, $\mathbf{X}$, π, m, N, ν_t).
   b. Calculate $r(j, b)$, the percentage of last-generation candidates in universe b that contain variable j, for $j = 1, 2, \ldots, p$; see Section 2, (4);
   End for.
Output
In matrix form, $r(j, b)$ for $j = 1, 2, \ldots, p$ and $b = 1, 2, \ldots, B$.
Although for making the bubble plot we need only $\bar{r}_j$ [see (5), Sec. 2], we need $r(j, b)$ for all b to make the dynamic bubble plot (Sec. 5).

2.1 The Illustrative Example (Continued)

Figure 1(a) shows the result of parallel evolution on our illustrative example; it plots $\bar{r}_j$ for $j = 1, 2, \ldots, p$. The three variables used to generate the data are represented by circles, and all other variables are represented by crosses. A total of $B = 25$ parallel universes are used, each evolving for only $N = 8$ generations. All other control parameters, such as population size and mutation rate, are exactly the same as before. In terms of the amount of computation, a total of $N \times B = 8 \times 25 = 200$ populations are evaluated, the same as using an SGA with 200 generations. But what a difference it makes!

Figure 1(a) shows that the three variables used to generate the data are apparently well separated from the rest. We call such a plot the *bubble plot*, because the important variables float on the top like bubbles.

In general, we must determine q, the number of variables to select. This can be done by plotting $\bar{r}_j$ from the largest to the smallest and looking for a large gap. We call such a plot the *order plot*. We discuss the statistical significance of this gap in more detail in Section 3.3. Figure 1(b) shows the order plot for our illustrative example. The idea behind the order plot is analogous to, for example, standard practices in principal component analysis, where the number of important components is often determined by plotting the ordered eigenvalues of the covariance matrix (see, e.g., Mardia, Kent, and Bibby 1979).

From Figure 1(a), we can also see that $\bar{r}_{15} > \bar{r}_{10} > \bar{r}_5$, which is consistent with the fact that $\beta_{15} > \beta_{10} > \beta_5$.

2.2 Why Does It Work?

In this section we explain why using a number of short evolutionary paths is better than using one long evolutionary path. By a "short path," we mean a path that has not fully converged. This is often referred to as "early stopping." In Section 3.2 we give more details on the convergence criterion. The concept of early stopping is widely used in the neural network community to avoid overfitting (see, e.g., Hastie, Tibshirani, and Friedman 2001, sec. 11.5.2). Here we can consider the convergent solution to be "overfitting" in the sense that it tends to include some spurious variables (see Table 4). The reason why spurious variables are often included is because the objective functions used to evaluate models are often highly asymmetric, in the sense that as far as model performance is concerned, it is often much worse to exclude an important variable than to include a spurious variable.

Therefore, in the evolutionary process, a model that contains all of the important variables will generally have a high fitness score. As a result, all the other spurious variables contained in that model (by random chance) will also be considered important by the evolutionary force. Because the spurious variables are there by chance, when the GA has not fully converged, these variables most likely will be different across the parallel universes. Therefore, by running a number of short paths (i.e., paths that have not fully converged), these spurious variables can be effectively eliminated. More specifically, if a variable j is truly important, then $r(j, b)$ will be high for all or most b, whereas if a variable j is spurious, then $r(j, b)$ will be high (by chance) only for some b. Therefore, when averaged over

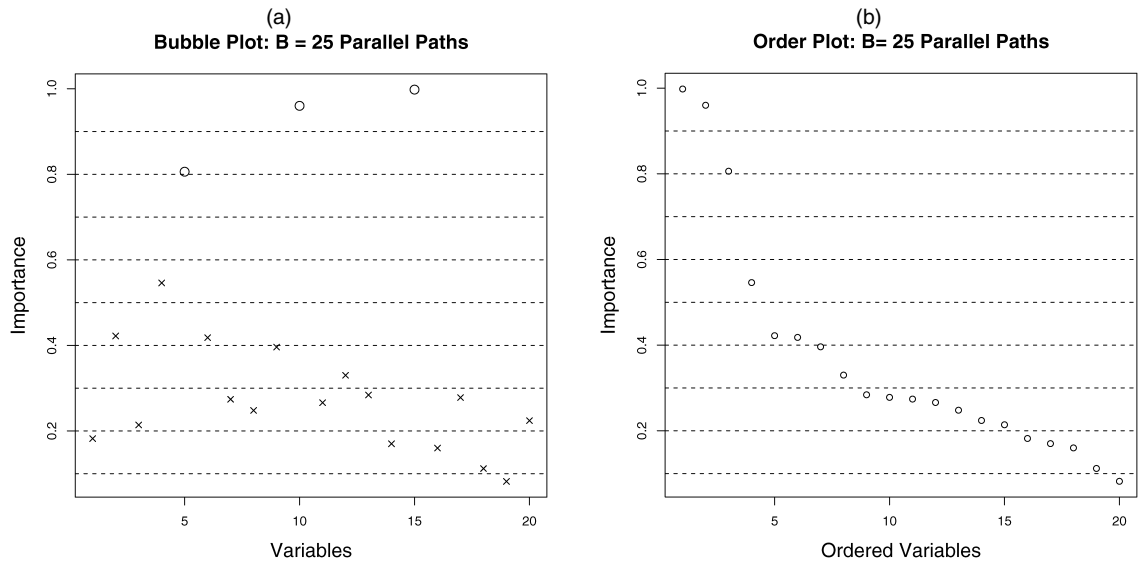

Figure 1. Illustrative Example. (a) The bubble plot. Plotted here are the values of $\bar{r}_j$, for $j = 1, 2, \ldots, p$. The variables used to generate the data are represented by circles; all other variables are represented by crosses. (b) The order plot. Plotted here are the values of $\bar{r}_j$ ordered from the largest to the smallest. The number of variables to select can be determined by looking for a large gap in this plot.

$b = 1, 2, \ldots, B$, $\bar{r}_j$ will be high only for variables that are truly important. This is how parallel evolution can be used to make variable selection choices.

Remark. In numerical optimization, it is common to start an algorithm several times, each time with a different initial value. Here it is crucial to understand that parallel evolution is *not* the same as simply trying out different initial values and picking out the best solution in the end; it actually *consolidates* useful information from *all* parallel paths. By doing so, it discovers something new—something never discovered by a single path alone, namely the correct solution.

3. DETAILS OF THE PARALLEL GENETIC ALGORITHM

We now give more details for how to implement the PGA and discuss how to specify the various control parameters, such as population size, mutation rate, and stopping criterion for each individual evolutionary path, as well as the number of parallel paths needed.

3.1 Individual Paths

There are many different ways to implement the GA. In the context of parallel evolution, it is desirable to keep each individual path as simple and self-contained as possible. Recall that the whole point of parallel evolution is to avoid having to think too hard about how to implement each individual path (Sec. 1.4).

Toward this end, we stick to the following simple scheme. We always start with a population size m that is an even number. The survival pool will always consist of the best $m/2$ individuals. A father and a mother will be chosen randomly from the survival pool to produce an offspring (which is allowed to mutate with a small probability) until a total of $m/2$ offsprings are produced to make up for a new generation of the same size m. Table 5 contains a summary of our implementation of a single evolutionary path.

There are two tuning parameters that we must specify: the population size of each generation and the mutation rate. Whereas the truly optimal values of these tuning parameters are often problem-dependent, in the spirit of keeping each individual path as self-contained and automatic as possible, we prefer to use a fixed set of choices. A number of theoretical and empirical studies on good choices of these tuning parameters are available (e.g., De Jong 1975; Grefenstette 1985; Goldberg 1989, p. 71). The website *http://www.mathworks.com/access/helpdesk/help/toolbox/gads/*, maintained by the producer of Matlab, also provides some general guidelines. These sources suggest that a fairly good and general choice is to set

$$\text{population size} = \text{dimension of the problem}$$

and

$$\text{mutation rate} = \frac{1}{\text{population size}}.$$

So for a total of p candidate variables, each generation will consist of $[p]_e$ individuals, and the mutation rate will be equal to $1/p$. Here the notation $[p]_e$ is used to mean

$$[p]_e = \begin{cases} p & \text{if } p \text{ is even} \\ p+1 & \text{if } p \text{ is odd.} \end{cases}$$

3.2 Stopping Criterion for Individual Paths

In Section 2.2 we stated that each individual evolutionary path must be relatively short for the strategy of parallel evolution to be effective. In this section we specify the appropriate stopping criterion. Given a collection of binary sequences of length p (with each sequence containing p bits), let r_j be the frequency that the jth bit is equal to 1. Then the average "per bit" entropy of this collection (which we simply call "entropy" here) is given by

$$\text{entropy} = -\frac{1}{p}\sum_{j=1}^{p} r_j \log_2(r_j) + (1 - r_j)\log_2(1 - r_j).$$

If the population is pure in the sense that all of the sequences are identical, then r_j is either 1 or 0 for all j and the entropy is 0. Therefore, the GA can be regarded as having converged when the entropy of the population is sufficiently close to 0, that is, below a prespecified threshold δ (e.g., $\delta = .05$). In reality, we require the average entropy over the latest five generations to be below δ so that the procedure is more numerically stable.

For every problem, then, we first run the GA a couple of times (e.g., 5 or 10 times) using different initial populations and record the average number of generations needed to achieve convergence. We then use half that number of generations for each parallel path.

Remark. Clearly, for a single GA path that has converged in the aforementioned sense, it does not matter whether we average over the entire population or simply take the best individual from the last generation as our solution because every individual will essentially be identical.

3.3 Number of Parallel Paths Needed

Finally, we address the issue of how many parallel paths are needed. Because the parallel evolutionary paths are generated independently of one another, one can think of $r(j, 1), r(j, 2), \ldots, r(j, B)$ as an independent sample from some underlying distribution, say $\mathcal{F}_j$. The fact that $r(j, b)$ is bounded between 0 and 1 means (see, e.g., Ross 1997, problem 11.33) that

$$v(j, b) \equiv \text{var}(r(j, b)) \le \frac{1}{4},$$

and hence

$$\text{var}(\bar{r}_j) \le \frac{1}{4B}.$$

The central limit theorem then implies that for large B, $\bar{r}_j$ is approximately normally distributed with a variance no larger than $\frac{1}{4B}$.

Now let $\mathcal{U}$ be the set consisting of the top q variables (i.e., those that float on top of the bubble plot) and let $\mathcal{L}$ be the set consisting of the remaining variables (i.e., those that sink to the bottom). If we define

$$d = \min(\bar{r}_j : j \in \mathcal{U}) - \max(\bar{r}_j : j \in \mathcal{L})$$

to be size of the gap between these two sets of variables, then we generally can be quite confident in the result produced by parallel evolution if the gap d is larger than a certain multiple

of the maximal standard deviation of $\bar{r}_j$, that is,

$$d \geq \alpha\sqrt{\frac{1}{4B}}. \tag{6}$$

A reasonable and convenient choice here is the commonly used Gaussian critical value of $\alpha = 1.645$, so we get

$$d \geq 1.645\sqrt{\frac{1}{4B}} = .8225B^{-1/2}. \tag{7}$$

This gives us a very simple criterion for choosing B. More specifically, we simply use a sufficient number of parallel paths so that we can be reasonably confident that the gap between the two sets of variables is not due to random fluctuation. The fact that these parallel paths are generated independently means in practice we can start with a fairly small B (say around 20 to 30 paths), check condition (7), and add more paths if needed.

In reality, depending on the specific problem, $v(j, b)$ can sometimes be much less than $1/4$. This means that the criterion (7) sometimes can be relaxed a bit in practice, and we may be willing to accept the answer based on a gap d that is approximately equal to or even slightly less than $.8225B^{-1/2}$.

For the foregoing illustrative example (Secs. 1.2 and 2.1), we can see from the bubble plot that $\bar{r}_{15} > \bar{r}_{10} > \bar{r}_5 > .8$ and $\bar{r}_j < .6$ for all $j \neq 5, 10, 15$. For $B = 25$, we clearly have $d > .2 > .1645 = .8225/\sqrt{25}$. In other words, we can be quite confident in the result based on just 25 parallel paths.

3.4 Fitness Function

If a problem is large and evaluating the fitness function (3) repeatedly becomes computationally expensive, then it is possible to use a generalized cross-validation (GCV) criterion as an approximation without making any tangible difference to the performance of PGA,

$$F(\omega) \equiv -\text{GCVRSS}(\omega) = -\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i(\omega)}{1 - \text{tr}(\mathbf{H})/n}\right)^2$$

$$= -\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i(\omega)}{1 - (q+1)/n}\right)^2, \tag{8}$$

where q is the number of variables contained in the subset ω. In fact, experiments similar to those in Section 4 (not reported here) indicate that we can also use, for example, the AIC or Mallows' C_p as the fitness function for the PGA without affecting its performance.

3.5 Computational Cost

It is easy to quantify the total computational cost for parallel evolution. If a total of B parallel paths are used, each evolving for N generations with a population size of m, then a total of $m \times N \times B$ models are fitted and evaluated before the solution is determined.

3.6 Summary

Table 6 describes the PGA algorithm. Let us summarize our main findings so far. Generally speaking, it is *not* easy to use the GA for variable selection. However, by running a number of GAs in parallel with early stopping and combining the information from all runs, the performance becomes much more encouraging. We emphasize here again that, other than using fewer generations in each path, all other control parameters (e.g., population size, mutation rate, the fitness function) for each individual GA path are exactly the same as before. In our opinion, this finding alone is already quite remarkable in itself and has important practical implications. In the rest of this article we present a number of both real and simulated examples to demonstrate systematically that the PGA is not only an interesting, but also a highly competitive and easy-to-use, variable selection algorithm.

4. EXAMPLES

In this section we present a wide variety of examples, using both real and simulated data. The parameters in PGA—population size m, number of generations for each path N, and total number of parallel paths B—are chosen using the principles outlined in Section 3; the actual numbers used in each example are reported on a case-by-case basis so that the computational cost can be assessed for each case.

4.1 Pollution Data

We first analyze the well-known pollution data, originally from McDonald and Schwing (1973) and used by Miller (2002) throughout as a primary example. In fact, Miller (2002) used a number of different datasets; we choose this one in particular because it is actually the largest ($n = 60$ and $p = 15$) from his collection.

Because $p = 15$, this problem is small enough so that an exhaustive search over all possible subsets is feasible. The `leaps` library in `R` allows us to conduct such a search using the AIC as the objective function; the best subset is found to be $\{1, 2, 3, 6, 9, 14\}$. To be fully comparable, we also use the AIC as the objective function for stepwise selection and the PGA. Table 7 shows the variables selected by these different methods. We can see that, using the same objective function, the PGA is capable of finding the same solution as an exhaustive search, whereas stepwise selection is not.

For the PGA, a total of $B = 25$ parallel paths are used, each evolving for $N = 8$ generations with a population size of $m = 16$ (see also Fig. 2). This simple example confirms that the PGA is capable of conducting a more thorough search over the space of Ω than a greedy algorithm, such as stepwise selection.

4.2 A Hard Problem

We now consider a simulated example taken from George and McCulloch (1993, example 4.2). There are $n = 120$ observations and $p = 60$ variables. In addition, all 60 variables are

Table 7. Results for the Pollution Data

Method	*Selected variables*
Stepwise	1, 2, 3, 4, 5, 6, 9, 12, 13
Exhaustive	1, 2, 3, 6, 9, 14
PGA	1, 2, 3, 6, 9, 14

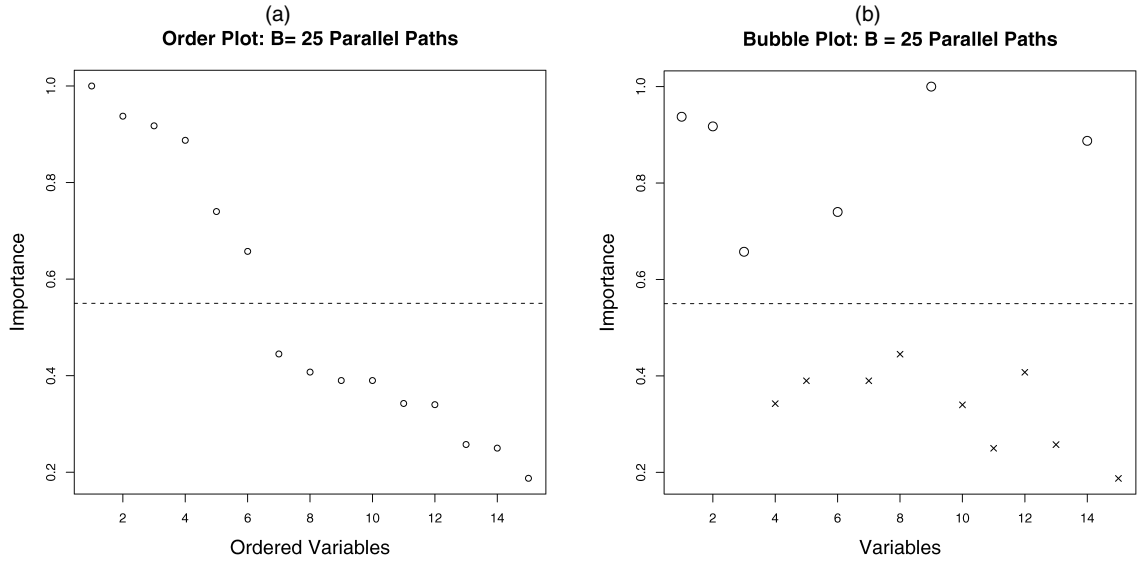

Figure 2. Pollution Data Example. Results from parallel evolution: order plot (a) and bubble plot (b). A total of 25 parallel paths are used, each evolving for 8 generations. The variables selected by exhaustive search are represented by circles; all other variables are represented by crosses.

made to be correlated; they are generated as

$$\mathbf{x}_j = \mathbf{z} + \boldsymbol{\epsilon}_j, \qquad j = 1, 2, \ldots, 60, \boldsymbol{\epsilon}_j, \mathbf{z} \overset{\text{iid}}{\sim} \mathrm{N}_{120}(\mathbf{0}, \mathbf{I}). \tag{9}$$

This implies the pairwise correlation among all of the variables is .5. The resulting problem is generally considered a hard variable selection problem both because of its size ($p = 60$) and because of the correlation among the variables (e.g., Liang and Wong 2000). The response variable $\mathbf{y}$ is generated as

$$\mathbf{y} = \beta_1 \mathbf{x}_1 + \cdots + \beta_{60} \mathbf{x}_{60} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathrm{N}_{120}(\mathbf{0}, \sigma^2 \mathbf{I}), \sigma = 2, \tag{10}$$

where $\beta_1 = \cdots = \beta_{15} = 0$, $\beta_{16} = \cdots = \beta_{30} = 1$, $\beta_{31} = \cdots = \beta_{45} = 2$, and $\beta_{46} = \cdots = \beta_{60} = 3$. In other words, the variables form four distinctive groups. The first group (1–15) is useless, with coefficients 0. The second group (16–30) is relatively weak (small nonzero coefficients relative to the noise σ). The third group (31–45) is stronger, and the fourth group (46–60) is the strongest.

Figure 3 shows a sequence of bubble plots from the PGA, for $B = 25, 50, 100$, and 500 parallel paths; each path is evolved for $N = 15$ generations with a population size of $m = p = 60$ (see Sec. 3.3). We can see that even when B is as small as 25, all of the correct variables are already ranked at the top and ahead of the rest of the variables. However, there are a few ambiguous gaps in the bubble plot, which can make selecting variables somewhat difficult for an unexperienced data analyst. Even when B is increased to 50 or 100, there are still two ambiguous gaps in the bubble plot, indicated in the plot by dashed horizontal lines. In fact, neither of these gaps is significant according to the criterion (7). As mentioned earlier (Sec. 3.3), when this happens we may consider running yet a few more parallel paths. When we increase B to 500, for example, we can see that there is only one unambiguous (and significant) gap remaining in the bubble plot—one that will give us exactly the right solution. In addition, the fact that a few variables belonging to the second group (variables 16–30) are floating a bit below all of the other important variables in the bubble plot is also consistent with fact that these are all variables with the smallest nonzero coefficient relative to the noise σ. This example demonstrates the PGA's ability to solve a large and difficult variable selection problem.

4.3 A Simulation Study

We now conduct a repeated simulation study to evaluate the behavior and performance of the PGA more systematically. We also compare the PGA against a number of readily available variable selection methods, such as stepwise selection and exhaustive search using both the AIC and the BIC as the selection criteria. We also consider a (completely different) Bayesian stochastic variable selection method known as stochastic search variable selection (SSVS) (George and McCulloch 1993).

4.3.1 Simulation Settings. Our simulation is based on the illustrative example in Section 1.2. In Sections 1.2 and 2.1 we have already presented some evidence that parallel evolution is useful for this example. Here we repeat this simulation 100 times and also consider four variations (Table 8).

In the first three variations, we make $\mathbf{x}_{20}$ highly correlated with one of the three useful variables. That is, for $j = 5, 10, 15$, we generate $\mathbf{x}_{20}$ as

$$\mathbf{x}_{20} = \mathbf{x}_j + .25\mathbf{z}, \qquad \mathbf{z} \sim \mathrm{N}_{40}(\mathbf{0}, \mathbf{I}).$$

The resulting correlation, $\mathrm{corr}(\mathbf{x}_{20}, \mathbf{x}_j) \approx .97$, is quite high. Hence these variations illustrate the behavior of parallel evolution when the problem is highly collinear.

In the fourth variation, we adopt the same data-generating mechanism as in Section 4.2 to create high pairwise correlations among *all* of the variables instead of introducing just one correlated variable as is done in the first three variations. In particular, the response is still generated as

$$\mathbf{y} = \mathbf{x}_5 + 2\mathbf{x}_{10} + 3\mathbf{x}_{15} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathrm{N}_{40}(\mathbf{0}, \sigma^2 \mathbf{I}),$$

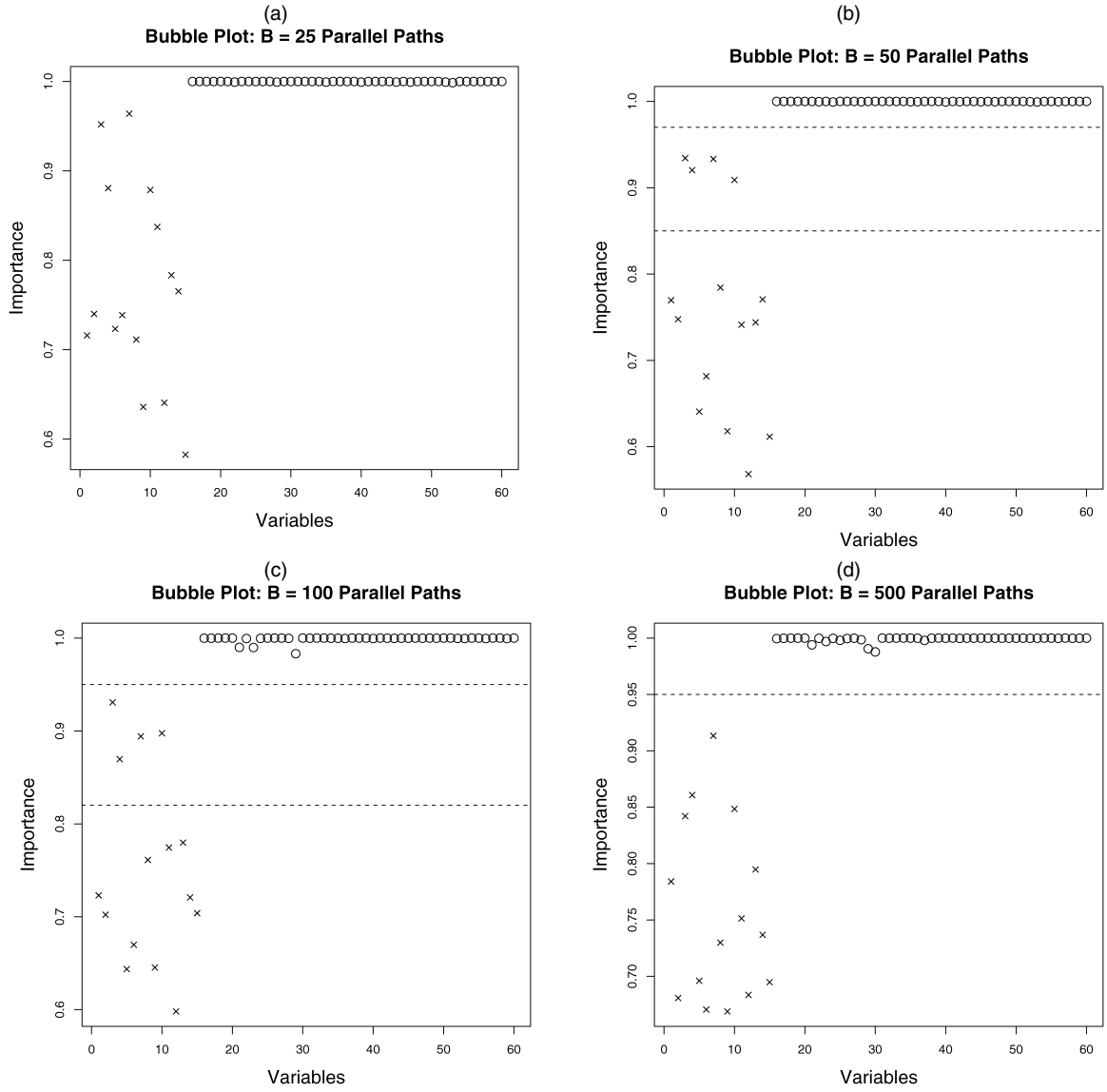

Figure 3. A Hard Problem. Shown here are four bubble plots, for B = 2 (a), 50 (b), 100 (c), and 500 (d) parallel paths.

except $\sigma = 2$ now instead of 1 and the predictors are generated not independently but according to

$$\mathbf{x}_j = \mathbf{z} + \boldsymbol{\epsilon}_j, \qquad j = 1, 2, \ldots, 20, \boldsymbol{\epsilon}_j, \mathbf{z} \overset{\text{iid}}{\sim} \mathrm{N}_{40}(\mathbf{0}, \mathbf{I}). \quad (11)$$

A single PGA run here consists of a total of $B = 25$ parallel paths each evolving for 8 generations with a population size of $m = p = 20$. This amounts to evaluating 4,000 linear models, which is a tiny fraction of 2^{20}, the number of all possible models.

Table 8. Simulation Study: Summary of All Variations

Scenario	σ	*Correlation structure*
Variation 0:	1	$\rho(\mathbf{x}_j, \mathbf{x}_k) = 0$ for all $j \neq k$
Variation 1:	1	$\rho(\mathbf{x}_j, \mathbf{x}_k) \approx .97$ for $j = 5$ and $k = 20$
Variation 2:	1	$\rho(\mathbf{x}_j, \mathbf{x}_k) \approx .97$ for $j = 10$ and $k = 20$
Variation 3:	1	$\rho(\mathbf{x}_j, \mathbf{x}_k) \approx .97$ for $j = 15$ and $k = 20$
Variation 4:	2	$\rho(\mathbf{x}_j, \mathbf{x}_k) = .80$ for all $j \neq k$

4.3.2 Performance Evaluation. In each variation, we repeat the same simulation experiment 100 times, each time generating a slightly different dataset with the same mechanism. Let $T(M)$ be the number of times that the correct subset of variables is identified by method M over the 100 simulations; our performance metric for method M is simply $T(M)/100$ or the proportion of times that the correct model is identified.

For both the PGA and SSVS, we report two different ways of calculating the performance metric—one that we call the "hard" metric and another one that we call the "soft" metric. We first state how the "hard" and "soft" metrics are calculated, and then discuss why the "soft" metric is introduced. To calculate the "hard" metric for the PGA, we simply find the largest gap in the bubble plot using B parallel paths and select the variables above the gap. If these variables coincide exactly with the true variables used to generate the model, then the PGA is considered to have found the right model. To calculate the "soft" metric, we focus instead on how variable j is ranked by $\bar{r}_j$; the PGA is considered to have found the right model if all of the correct variables are ranked ahead of the rest using $\bar{r}_j$. To calculate the

"hard" metric for SSVS, we simply rely on the most probable model found by SSVS. If this model coincides exactly with the true model, then SSVS is considered to have found the right answer. To calculate the "soft" metric, we rely on the marginal posterior probability that variable j is active. SSVS is considered to have found the right model if all of the correct variables are ranked ahead of the rest using the posterior probability.

We now explain why the "soft" performance metric is introduced and how it can be used together with the "hard" metric to evaluate the potential of the PGA as a variable selection tool. We argue that the "hard" metric can be considered a lower bound and that the "soft" metric can be considered an upper bound.

Consider again the case depicted in Figure 3. If we use $B = 50$ parallel paths, then selecting all variables above the largest gap in the bubble plot actually includes four extra variables. Using the "hard" metric, the PGA is considered a failure, but this clearly understates the PGA's capability and potential for identifying the correct set of variables in this particular case. Therefore, the "hard" performance metric can be considered a lower bound on the performance.

One possibility here is to use a very large B in our simulation experiments to be conservative, but this not only slows down our simulation experiments considerably (especially considering that we are running 100 repetitions), but also creates a misleading impression that B must be very large for the PGA to work. We have already seen in Section 4.2 that, even for a very hard variable selection problem, the PGA can produce very meaningful information with as few as 25 parallel paths.

The "soft" performance metric is introduced so that the readers can better evaluate the potential of the PGA. If, with $B = 25$ parallel paths, the PGA is already ranking the correct variables ahead of everything else, this can be, to some extent, considered an indication that the PGA will eventually be able to identify the right answer when a larger B is used. Therefore, the "soft" performance metric can be considered an upper bound on the performance.

4.3.3 Results. Table 9 presents the results. Keep in mind that for 100 independent Bernoulli trials, the maximal standard error is $\sqrt{.5 \times .5/100} = .05$, corresponding to a success rate of .5; if the success rate is .95, then the standard error drops to $\sqrt{.95 \times .05/100} \approx .02$. This information is helpful for evaluating the amount of variation in the simulation results.

Recall that SSVS is a Bayesian method. To run SSVS, we use four different sets of hyperparameters for the prior distribution. The ones that we use are exactly the same as those used in the original SSVS article (George and McCulloch 1993): $(\hat{\sigma}_\beta/\tau, c) = (1, 5), (1, 10), (10, 100)$, and $(10, 500)$; refer to the original article for more details about SSVS. In fact, a more efficient version of SSVS (George and McCulloch 1997) is used, with an equivalent specification of the parameters τ and c.

As expected, variation 1 is the hardest case for all methods. There $\mathbf{x}_{20}$ is made to be highly correlated with $\mathbf{x}_5$, the variable that has the smallest nonzero coefficient ($1 = \beta_5 < \beta_{10} < \beta_{20}$) relative to the noise σ in the true model. It is hard to imagine that any method could avoid being "fooled" and not treat $\mathbf{x}_{20}$ as an important variable at least some of the time.

Generally speaking, Table 9 leads to the following conclusions: The SGA is hopeless, but the PGA is consistently better than both the AIC and the BIC regardless of whether an exhaustive search or a greedy stepwise search is used. Variation 0 and variation 3 are relatively easy problems, whereas variation 4 is somewhat more difficult because all of the variables are correlated with each other; in these cases we see that SSVS and the PGA are largely comparable. For the most difficult cases (i.e., variations 1 and 2), we see that SSVS is competitive against the PGA only if the hyperparameters for the prior distribution are set "correctly"; otherwise, it performs worse. In reality, of course, one does not generally know how to best set these hyperparameters in advance. In contrast, the PGA is relatively easy to use. This simulation study offers strong empirical evidence that the PGA can be a highly competitive variable selection tool.

4.4 A q-Bit D/A Converter

Finally, we illustrate the use of the PGA with a relatively large real industrial dataset studied by Filliben and Li (1997). A q-bit digital-to-analog (D/A) converter accepts q digital inputs, $D_1, D_2, \ldots, D_q$, and outputs an analog signal; each $D_j \in \{-1, +1\}$ represents an "on" $(+1)$ or "off" (-1) digital switch. To test the performance of the D/A converter, different digital signals are given to the device, and the conversion

Table 9. Simulation Study: Performance of Various Methods

	Scenario				
Method	*Variation 0*	*Variation 1*	*Variation 2*	*Variation 3*	*Variation 4*
Stepwise AIC	.01	.01	.01	.01	.02
Stepwise BIC	.20	.14	.18	.21	.20
Exhaustive AIC	.03	.01	.01	.01	.02
Exhaustive BIC	.28	.20	.23	.25	.27
SGA	.03	.02	.01	.00	.02
PGA (hard)	.80	.24	.64	.85	.49
PGA (soft)	.97	.63	.88	.95	.81
SSVS (1, 5; hard)	.81	.06	.51	.70	.47
SSVS (1, 10; hard)	.88	.04	.49	.82	.49
SSVS (10, 100; hard)	.58	.48	.61	.61	.56
SSVS (10, 500; hard)	.84	.62	.82	.86	.62
SSVS (1, 5; soft)	.97	.23	.77	.96	.76
SSVS (1, 10; soft)	.97	.23	.78	.97	.77
SSVS (10, 100; soft)	1.00	.72	.88	.98	.84
SSVS (10, 500; soft)	1.00	.72	.92	.98	.86

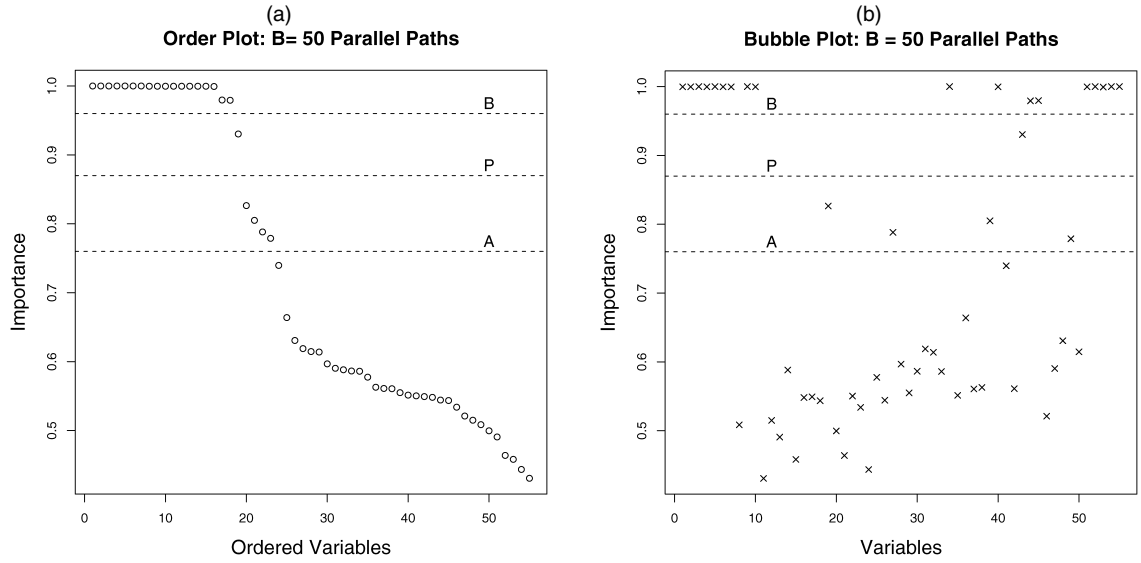

Figure 4. D/A Converter Example. Results from parallel evolution: order plot (a) and bubble plot (b). The PGA selects all variables above line P. Stepwise search using the BIC selects all variables above line B, whereas stepwise search using the AIC selects all variables above line A.

errors (Y) are recorded. It is desirable to understand how the conversion errors are related to the input bits. The dataset studied by Filliben and Li (1997) is for $q = 10$ and it is a 2^q full factorial design ($n = 1{,}024$). We limit our attention to models with at the most second-order interaction terms, that is,

$$Y = \beta_0 + \sum_{j=1}^{10} \beta_j D_j + \sum_{k=2}^{10} \sum_{j<k} \beta_{jk} D_j D_k + \epsilon.$$

For $q = 10$, there are altogether 10 main effects and 45 two-way interaction terms, giving rise to a total of $p = 55$ candidate variables.

Figure 4 shows the results from the PGA. A total of $B = 50$ parallel paths are used, each evolving for $N = 20$ generations. Based on the largest gap in the bubble plot, the PGA selects all the variables above line P. Altogether 19 variables are selected; the gap at P is also significant according to the criterion (7).

Both SSVS and a stepwise search using the BIC select 18 variables; they both turn out to be the same as those above line B in Figure 4. On the other hand, a stepwise search using the AIC selects 23 variables; they turn out to be those above line A (Fig. 4). Table 10 summarizes these results and lists the variables selected by different methods; the notation "j:k" denotes the two-way interaction term $D_j D_k$.

4.4.1 Evaluation of Competing Solutions. It is well known that, unless n is very small, the AIC has a tendency to select more variables, whereas the BIC has a tendency to select fewer variables (e.g., Kou and Efron 2002). The fact that the PGA has produced a solution somewhere in the middle is, in some sense, a comfortable position for us to be in. This said, we hasten to add that this fact alone does *not* mean that the PGA solution here is in any sense superior to the AIC or BIC solutions or that the PGA solution will always be somewhere between the two.

For a real problem like this, it is both impossible and quite meaningless to judge, without obtaining extra information, which solution is superior when the solutions are so similar and differ by just a few terms. To further evaluate the relative "goodness" of these different subsets, we perform a *crude* two-fold cross-validation experiment. First, the dataset is randomly divided into two groups, g_1 and g_2. Let $\mathcal{S}_A$, $\mathcal{S}_B$, and $\mathcal{S}_P$ be the subset of variables identified by the AIC, BIC, and PGA. If we use $|\mathcal{S}|$ to denote the size of the subset $\mathcal{S}$, then $|\mathcal{S}_A| = 23$, $|\mathcal{S}_B| = 18$, and $|\mathcal{S}_P| = 19$. For a given subset of variables $\mathcal{S}$, we evaluate its "goodness" by an estimate of its prediction error (PE),

$$\widehat{\mathrm{PE}(\mathcal{S})} = \sum_{k=1}^{2} \sum_{i \in g_k} \big(y_i - \hat{y}_i(\mathcal{S}; -g_k)\big)^2,$$

where $\hat{y}_i(\mathcal{S}; -g_k)$ means the predicted response for observation i using a model that includes just the variables in the subset $\mathcal{S}$ but whose coefficients are estimated using all observations *except* those in group g_k. To account for the fact that the groups g_1 and g_2 are created randomly, this experiment is repeated 10 times. The mean prediction errors (MPEs) for the three competing subsets ($\mathcal{S}_A$, $\mathcal{S}_B$, and $\mathcal{S}_P$) over these 10 repeated experiments are listed in Table 11. Clearly, they are quite comparable

Table 10. D/A Converter Example: Variables Selected by Different Methods

Method	*Selected variables*
Stepwise BIC	1, 2, 3, 4, 5, 6, 7, 9, 10, 3:10, 4:10, 5:9, 7:9, 7:10, 8:9, 8:10, 9:10
SSVS	Stepwise BIC solution
PGA	Stepwise BIC solution + 5:8
Stepwise AIC	PGA solution + 1:10, 2:10, 4:9, 6:10

Table 11. D/A Converter Example: MPE of Different Subsets

Subset	MPE (SE)
$\mathcal{S}_B$	$11.9695_{(.076)}$
$\mathcal{S}_P$	$11.9378_{(.076)}$
$\mathcal{S}_A$	$11.9686_{(.065)}$

statistically, with the subset $\mathcal{S}_P$ identified by the PGA having a slightly better mean estimate.

4.4.2 An Extra Insight From the PGA. The AIC and BIC solutions differ by five variables. Using the bubble plot from parallel evolution (Fig. 4), we can even conclude that one of these five variables is more important than the other four. To check whether this conclusion is sensible, we conduct an exhaustive search using the `leaps` library in `R` to find the best subset of size 19. Note that if the subset size is fixed a priori, then it does not matter whether the AIC or the BIC is used as the selection criterion. In this case, the exhaustive search does in fact find exactly the same 19 variables as those found by the PGA. This example shows that the PGA can be a useful practical tool for fairly large industrial applications where the primary goal is to identify the key factors that affect a certain outcome.

5. MAKING DYNAMIC BUBBLE PLOTS

For any B, we can create a bubble plot by plotting $\bar{r}_j$ against j. In our experience, it has proven very useful to create a series of such bubble plots as information from more and more parallel universes are consolidated. Treating each individual bubble plot as a single frame, we can create a dynamic movie showing how the values $\bar{r}_j$ $(j = 1, 2, \ldots, p)$ change with B. For obvious reasons, we are unable to include examples of such dynamic plots in the article; a number of examples are available from our websites.

If is often very apparent from such dynamic plots which variables are truly important; these are often the ones that quickly float to the top and stay there with relatively little fluctuation, whereas the rest tend to fluctuate quite a bit, especially when B is relatively small. This is very convenient because for relatively small B, even meaningful gaps in the bubble plot generally would be insignificant according to the criterion (7). But by looking at how the variables move in the dynamic bubble plot, one can often distinguish the important variables from the rest quite easily.

6. DISCUSSION

Before closing, we briefly discuss a number of possibilities for the PGA that we have not explored in this article.

Parallel Computing. First, there is the possibility of implementing the parallel evolution method using a parallel computing architecture. Although we did not explore this option, it should be fairly obvious how this can be achieved quite easily in practice. If one has access to, say, a 40-processor system, then one can invoke 40 parallel evolutionary paths simultaneously by sending one path to each processor because the individual paths are completely independent of one another.

Variable Selection for More Complex Models. Variable selection for models that are more complex than multiple linear regression is, of course, a much harder problem. Conceptually it is clear how the PGA can be adapted quite easily to solve these more difficult problems. The associated computational cost will of course be more substantial, because a total of $m \times N \times B$ models must be fitted, and this can become quite significant if, for example, each model is a generalized additive model (GAM) (Hastie and Tibshirani 1990) that is typically fitted with a rather expensive back-fitting algorithm, but it will still be much easier to implement than, say, SSVS, because we can simply embed a functional call to existing GAM software in the PGA algorithm, whereas the MCMC-based SSVS would require entirely different implementations for different types of models. With increasing computing power, a conceptually easy approach such as the PGA will become more and more attractive and useful in practice.

Variable Selection With the Effect Heredity Principle. In many industrial applications, models with two-factor interaction terms without at least one of the corresponding main effects are considered unreasonable, a principle sometimes referred to as "effect heredity" (see, e.g., Chipman, Hamada, and Wu 1997). It is possible to take such principles into account in GA/PGA by, for example, penalizing models that violate the principle in the fitness function or increasing the mutation probability for an inactive interaction term whenever one or both of its parents are active.

7. CONCLUSION

We now summarize the main contributions of this article. First, we demonstrated that the GA, although perfectly natural for the variable selection problem, is actually *not* easy to use or terribly effective. Toward this end, we devised a very simple and relatively mindless strategy of parallel evolution and turned the otherwise hard-to-use GA into a much more effective variable selection tool. We demonstrated with a systematic simulation study that parallel evolution or PGA is competitive in its ability to recover the correct model. We also illustrated the strength and usefulness of parallel evolution with both simulated and real datasets and pointed out its general applicability as a variable selection tool for more complex statistical models.

The PGA is a stochastic search method. In this regard, it is similar to SSVS, and they are both better than greedy stepwise methods. But as we have demonstrated in Section 4.3, the success of SSVS is heavily dependent on being able to select the correct set of prior parameters. In this regard, the PGA is somewhat more accessible and easier to use than SSVS in practice.

The apparent success of our simple strategy also reinforces a very important idea in modern statistical computing, that is, for some problems, consolidating information from a number of mediocre solutions may actually be better than conducting a greedy search for a single best solution. This important idea is sometimes referred to as "majority vote" (e.g., James 1998). The case of PGA further testifies to the importance of such an idea and demonstrates that there is more to be gained from parallel computation than simply reducing the amount of computational time. The value of the PGA as another compelling case in support of the "majority vote" idea is also significant in our

opinion, because we believe that such an idea will continue to have far-reaching consequences for scientific computation beyond the scope of the GA or the variable selection problem alone.

ACKNOWLEDGMENTS

Both authors are partially supported by the Natural Science and Engineering Research Council of Canada, the Mathematics of Information Technology and Complex Systems network, and the Canada Foundation for Innovation. The authors thank the editor, Randy Sitter, an associate editor, and two referees for their encouragement and constructive comments. The first author thanks Jason Loeppky for his discussion of a related problem.

[Received January 2004. Revised September 2005.]

REFERENCES

Akaike, H. (1973), "Information Theory and an Extension of the Maximum Likelihood Principle," in *Second International Symposium on Information Theory*, pp. 267–281.

Breiman, L. (1996), "Bagging Predictors," *Machine Learning*, 24, 123–140.

Chatterjee, S., Laudato, M., and Lynch, L. A. (1996), "Genetic Algorithms and Their Statistical Applications: An Introduction," *Computational Statistics and Data Analysis*, 22, 633–651.

Chipman, H. A., Hamada, H., and Wu, C. F. J. (1997), "A Bayesian Variable Selection Approach for Analyzing Designed Experiments With Complex Aliasing," *Technometrics*, 39, 372–381.

De Jong, K. (1975), "An Analysis of the Bahavior of a Class of Genetic Adaptive Systems," unpublished doctoral dissertation, University of Michigan, Department of Computer and Communication Sciences.

Draper, D., and Fouskakis, D. (2000), "A Case Study of Stochastic Optimization in Health Policy: Problem Formulation and Preliminary Results," *Journal of Global Optimization*, 18, 399–416.

Filliben, J. J., and Li, K. C. (1997), "A Systematic Approach to the Analysis of Complex Interaction Patterns in Two-Level Factorial Designs," *Technometrics*, 39, 286–297.

Furnival, G. M., and Wilson, R. W. (1974), "Regression by Leaps and Bounds," *Technometrics*, 16, 499–511.

George, E. I., and McCulloch, R. E. (1993), "Variable Selection via Gibbs Sampling," *Journal of the American Statistical Association*, 88, 881–889.

——— (1997), "Approaches for Bayesian Variable Selection," *Statistica Sinica*, 7, 339–374.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley.

Grefenstette, J. (1985), "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transaction on Systems, Man and Cybernetics*, 16, 122–128.

Hastie, T. J., and Tibshirani, R. J. (1990), *Generalized Additive Models*, New York: Chapman & Hall.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001), *The Elements of Statistical Learning: Data-Mining, Inference and Prediction*, New York: Springer-Verlag.

James, G. (1998), "Majority-Vote Classifiers: Theory and Applications," unpublished doctoral dissertation, Stanford University, Department of Statistics.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983), "Optimization by Simulated Annealing," *Science*, 220, 671–680.

Kou, S. C., and Efron, B. (2002), "Smoothers and the C_p, Generalized Maximum Likelihood, and Extended Exponential Criteria: A Geometric Approach," *Journal of the American Statistical Association*, 97, 766–782.

Liang, F., and Wong, W. H. (2000), "Evolutionary Monte Carlo: Applications to C_p Model Sampling and Change Point Problem," *Statistica Sinica*, 10, 317–342.

Mallows, C. L. (1973), "Some Comments on C_p," *Technometrics*, 15, 661–675.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979), *Multivariate Analysis*, New York: Academic Press.

McDonald, G. C., and Schwing, R. C. (1973), "Instabilities of Regression Estimates Relating Air Pollution to Mortality," *Technometrics*, 15, 463–482.

Miller, A. J. (2002), *Subset Selection in Regression* (2nd ed.), New York: Chapman & Hall.

Miyata, S., and Shen, X. (2003), "Adaptive Free-Knot Splines," *Journal of Computational and Graphical Statistics*, 12, 197–213.

R Development Core Team (2006), *R: A Language and Environment for Statistical Computing*, Vienna, Austria: R Foundation for Statistical Computing.

Ross, S. M. (1997), *Introduction to Probability Models* (6th ed.), New York: Academic Press.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *The Annals of Statistics*, 6, 461–464.

Venables, W. N., and Ripley, B. D. (1994), *Modern Applied Statistics With S–PLUS*, New York: Springer-Verlag.