A Genetic Algorithm Application In Nonparametric Functional Estimation

Cezary Z. Janikow and Haiyan Cai [a]

[a]Mathematics and Computer Science Department, University of Missouri – St. Louis, St. Louis, MO 63121, USA

Abstract
Genetic algorithms model the natural processes of information inheritance and selective pressure. As general domain independent tools, they were shown to be applicable to a number of numerical optimization problems, but the applications are often limited by constraints. In this paper we show that appropriately enhanced genetic algorithms can be effectively used to solving the complex, multi–dimensional, and highly constrained problem of statistical functional estimation. Our approach deals with the constraints by utilizing them to reduce the search space and defining only operators closed in the reduced space. We describe here the algorithm and present few experimental results.

1. INTRODUCTION

Recent problem solving ideas tend to lean to utilization of active agents which evolve by interactions with the environment and the surrounding world rather than by isolated operations. These ideas are derived from nature where organisms both cooperate and compete for resources of the environment in the quest for a better adaptation. Because the nature is highly parallel, these observations led to the design of algorithms that could both provide the desired characteristics and be applicable to parallel architectures. One of the most successful such algorithms is the genetic algorithm (GA) [4].

GAs operate by a simulation in which a population of agents compete for survival and cooperate to achieve a better adaptation. The competition is stochastic but with survival chances of an agent proportional to its current level of adaptation. This process simulates the Darwinian selection. Here, the environment is the problem at hand, and the agents are judged by their quality as potential solutions. The cooperation is achieved by merging information from few (often two) agents to produce a new one, with the hope of producing more adapted individuals (a better solution). For this process to be successful, we must provide for two mechanisms. Firstly, we need a method that ensures that good agents have higher chances of being selected as donors (parents). This is achieved by the selective mechanisms which promotes survival of such individuals. Secondly, we must provide for mechanisms aimed at selecting and merging the information in an intelligent way. This is achieved by means of crossover operators modeling the natural DNA inheritance. Mutation operators are aimed at introducing extra variability. Algorithms utilizing these mechanisms were shown to be very robust and applicable for parallel processing [3].

GAs have been quite successfully applied to a number of problems, but by far they are famous as numerical optimization algorithms. The reason for this success is that numerical solutions can be easily represented as artificial agents and that the quality assessment of such agents is reduced to function evaluation.

A serious problem that arises in such applications is that of processing constraints. A natural approach is to impose penalties for constraint violations. In other words, a constrained problem is transformed into an unconstrained problem by associating a penalty with all constraint violations and the penalties are included in the function evaluation. However, though the evaluation function is usually well defined, there is no accepted methodology for combining it with the penalty [2]. Many improvements have been proposed. For example, a common approach is to start with smaller penalties in order to allow deeper space exploration, and subsequently tighten them in order to achieve a feasible solution [7]. However, such approaches are highly problem specific [7, 8] and generally do not guarantee a feasible solution for highly constrained problems. The penalty approach is suitable only for weak (non–essential) constraints often present in constraint–based design problems. Unfortunately, numerical problems normally have strong constraints. Another approach is to devise special repair algorithms that guarantees that each solution be moved into the feasible solution space. However, for extensive set of constraints, this method introduces large computational overhead. Moreover, it does not reduce the search space to that feasible. Yet another approach is to define the operators so that their actions are closed in the search space. It is difficult to design such operators for general uses.

Since the genetic approach is basically an accelerated search of the feasible solution space, introducing constraints can be potentially advantageous and can improve the behavior of the technique by limiting the space to be searched. However, traditional GA approaches do not use this fact and rather apply techniques aimed at minimizing the negative effect of such constraints. This, in turn, often increases the search space by allowing some infeasible solutions outside the constrained solution space. In [6] we provided a general set of operators that are closed in the feasible space for numerical optimization problems with arbitrary linear constraints. In this paper, we describe a specific application of that method to the problem of survival analysis in biostatistics. We start by presenting the problem and the problem specific method for handling equality constraints in section 2. In section 3 we present the specific closed operators utilized here to handle the problem specific inequality constraints. Finally, we present some experiments in section 4.

2. THE STATISTICAL ESTIMATION PROBLEM

In clinical trials, the survival time T of an experimental subject usually follows the distribution such that, for a given time t, the chance that the subject will survive after t is

$$P(T > t) = \exp[-\int_0^t (1 - \lambda(s) + \lambda(s)e^{\beta x})\phi_0(s)ds] \tag{1}$$

In above formula, x takes values either 1 or 0, depending on whether or not the experimental subject is receiving certain treatment. The function $\lambda(t)$ is a nondecreasing function defined on $[0, \infty)$, with $\lambda(0) = 0$ and $\lambda(\infty) = 1$. The values of $\lambda(t)$ reflect the percentage of the full effect of the treatment achieved at time t. Function $\phi_0(t)$ is called controlled hazard rate and can be a rather arbitrary positive function. Suppose β is given and the functions $\lambda(t)$ and $\phi_0(t)$ are unknown. One needs to estimate the function $\lambda(t)$.

The usual parametric approach to the estimation problem is to assume a functional form for $\lambda(t)$ so that the function depends on some unknown parameters. Then a "partial likelihood function" can be defined based on data as a function of those unknown parameters. Maximizing the likelihood function will provide estimates for the parameters. A major difficulty of such approaches is that, in many practical problems, the information about the function $\lambda(t)$ is very limited. Therefore, a good form for $\lambda(t)$ is hard to guess. The algorithm proposed in this paper provides an alternative, GA–based, approach, which needs fewer information from the distribution and is rather robust. No knowledge about the form of $\lambda(t)$ is needed in estimation. We decided to estimate $\lambda(t)$ by computing discrete points yielding a vector $\vec{\lambda(t)}$, which subsequently may be interpolated.

First, let us observe that, since $\lambda(t)$ is a nondecreasing function, the following inequalities must hold for k_0 discretized points: $0 \leq \lambda_0 \leq \cdots \leq \lambda_{k_0-1} \leq 1$.

Since our idea is to use the available equality constraints to reduce the search space, we are interesting in detection of all such possible constraints. It turns out that a number of them can be generated. Suppose independent observations from n individuals are collected. The data consist of the pairs (t_i, x_i), $i = 1, ..., n$, where t_i is the survival time of the ith individual and x_i is his treatment status. The values of the function $\lambda(t)$ can be estimated at some of the time points t_i as follows. Suppose T_i's are arranged in the way so that $t_1 < \ldots < t_n$. Let

$$s_i = \text{sign}\left(\sum_{j=i}^{n}(e^{\beta x_i} - e^{\beta x_j})\right) \tag{2}$$

In the sequence $s_1, s_2, ..., s_n$, consider the subsequences starting with consecutive '+' signs then following by consecutive '−' signs. We may easily show that, for each k-th sequence, the following equality constraints must hold: $\lambda_{k^i} = \cdots = \lambda_{k^j}$. Then, we suggest to remove all these equality constraints by replacing all these λ values by a single value η_k. Furthermore, let τ_k be the location of the first '+' sign in the subsequence. Define

$$f_k(\eta) = \sum_{i=\tau_k}^{\tau_{k+1}-1} \{\log[1 + \eta(e^{\beta x_i} - 1)] - \log \sum_{j=i}^{n}[1 + \eta(e^{\beta x_j} - 1)]\} \tag{3}$$

and

$$F(\eta_0, \ldots, \eta_{k_{0-1}}) = \sum_{k=0}^{k_{0-1}} f_k(\eta_k) \tag{4}$$

assuming that we have k_0 such sequences. Then, it can be shown that good estimates λ_k of $\lambda(T_{\tau_k})$ will maximize the function F under the constraint [9]: $0 \leq \eta_0 \leq \cdots \leq \eta_{k_0-1} \leq 1$.

In other words, we remove the troublesome equality constraints and, at the same time, reduce the search space by reducing the number of estimate parameters. Note that $k_0 \leq n/2$, but experiments show that most often $k_0 \approx n/4$. This process constitutes the problem specific data reduction by removing the detected equality constraints. In addition, for some specific cases we may detect additional constraints of the form: $\eta_0 = 0$ and $\eta_{k_0-1} = 1$. We discuss dealing with these additional equalities after presenting our convex–search–space–closed operators.

An interpolation of the points (η_k, t_{τ_k}), which may easily be converted to (λ_i, t_i), will provide an estimate for $\lambda(t)$. This problem is basically a nonlinear programming problem with linear constrains. Typical algorithms for solving such problems are limited to functions with small number of variables and the computations become more and more unreliable when the the number of variables increases. In our application, however, to obtained adequate estimates for the function $\lambda(t)$ the number of the variables should be as large as possible. It is desirable, for example, to have $n \geq 200$. Therefore, the usual algorithms do not provide practical solutions to this problem.

3. THE PROPOSED APPROACH

The methodology we proposed in [6] provides a way of handling strong linear constraints that is both general and problem independent. Its main idea is to use the equalities to limit the problem's dimensionality, and then to use the inequalities to further limit the search space by providing a set of closed operators to explore the feasible space only. In this problem of survival analysis, we already have removed the detected equalities, thus reduced the search space, and are left with the following constraints: $0 \leq \eta_0 \leq \cdots \leq \eta_{k_0-1} \leq 1$.

3.1. Representation

The most often used agent representation, for numerical problems, is binary vector made of individual genes coding individual variables. A potential problem with this coding is that distance from the problem space is not preserved in the representation space. This problem can be avoided by special coding techniques as Grey [1]. In [5] we showed that a floating point representation also removes the same bias as it also reduces run–to–run variance and is generally faster without of any precision sacrifice. Therefore, for this algorithm, we decided to use this representation.

In the floating point representation, each chromosome vector is coded as a vector of floating point numbers which is of the same length as the sought solution vector. For example, assuming a problem of k_0 variables, a single solution has the form: $\eta = \langle \eta_0, \ldots, \eta_{k_0-1} \rangle$. The precision of such an approach depends on the underlying machine, but is generally much better than that of the binary representation. Of course, we can always extend the precision of the binary representation by introducing more bits, but this considerably slows down the algorithm. In addition, the floating point representation is capable of representing quite large domains (or cases of unknown domains). On the other hand, the binary representation must trade off the precision and domain size.

3.2. Closed Operators

Our operators are closed in the convex search space. In other words, they are context–dependent, meaning that the feasible value of η_n gene is restricted to $[\eta_{n-1}, \eta_{n+1}]$, with the first and the last genes restricted by 0 and 1. We also extensively use dynamic operators, that is those whose performance is quided by the age of the population [5].

Mutations are quite different from the traditional one with respect to the actual mutation: a gene, being a floating point number, is mutated in a context–dependent range.

- **Plain mutation** selects a random gene v_k of the chromosome $\eta_v^t = \langle \eta_0, \ldots, \eta_{k_0-1} \rangle$. The result is a vector $\eta_v^{t+1} = \langle \eta_0, \ldots, \eta'_k, \ldots, \eta_{k_0-1} \rangle$, where η'_k is a random value (with uniform probability distribution) from the range $[\eta_{k-1}, \eta_{k+1}]$. If $k = 0$ or $k = k_0 - 1$, then the appropriate boundary is taken to be 0 or 1, respectively.

- **Dynamic mutation** is the operator designed for fine tuning capabilities aimed at achieving high precisions [5]. It is defined as follows: if $\eta^t = \langle \eta_0, \ldots, \eta_{k_0-1} \rangle$ is the parent and the element η_k is selected for this mutation, the result is a vector $\eta^{t+1} = \langle \eta_0, \ldots, \eta'_k, \ldots, \eta_{k_0-1} \rangle$, with

  $$\eta'_k = \eta_k + \triangle(t, \eta_{k+1} - \eta_k) \text{ or } \eta_k - \triangle(t, \eta_k - \eta_{k-1}) \text{ selected randomly.} \qquad (5)$$

  The function $\triangle(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\triangle(t, y)$ being close to 0 increases as t increases. This property causes this operator to search the space uniformly initially (when t is small), and very locally at later stages. We have used the following function:

  $$\triangle(t, y) = y \cdot \left(1 - r^{(1-\frac{t}{T})^b}\right), \qquad (6)$$

  where r is a random number from $[0..1]$, T is the maximal generation number, and b is a system parameter determining the degree of non–uniformity. Here again, the used boundary value is 0 or 1 if k is 0 or $k_0 - 1$, respectively.

- **Shift range** is designed to adjust a whole range of parameters at a time. This operator is designed especially for this problem, whose individual genes are so closely related. It is designed as follows: if $\eta^t = \langle \eta_0, \ldots, \eta_n, \ldots, \eta_m, \ldots, \eta_{k_0-1} \rangle$ is selected as the parent, with randomly selected n, m such that $0 \leq n < m < l$ (what follows is that $\eta_n \leq \eta_m$), then the offspring is of the form: $\eta^t = \langle \eta_0, \ldots, \eta_n + \delta, \ldots, \eta_m + \delta, \ldots, \eta_{k_0-1} \rangle$. δ is a random value from a feasible range, that is from the range that will preserve the inequality constraints. This range is easily computed as follows: $\delta \in [\eta_{n-1} - \eta_n, \eta_{m+1} - \eta_m]$. Again, we deal with the boundary cases separately: if $n = 0$ then $\eta_{n-1} = 0$, and if $m = k_0 - 1$ then $\eta_{m+1} = 1$. Note that the shift amount can be positive or negative resulting in moving the range in any direction.

- **Stretch range** is similar to shift range, but it multiplies the selected genes by a feasible factor: $\eta^t = \langle \eta_0, \ldots, \eta_n \cdot \gamma, \ldots, \eta_m \cdot \gamma, \ldots, \eta_{k_0-1} \rangle$. The stretch factor γ is taken as a random value from the feasible range: $\gamma \in [\frac{\eta_{n-1}}{\eta_n}, \frac{\eta_{m+1}}{\eta_m}]$, and can be either greater or less than one. Here, if $n = 0$ then $\eta_{n-1} = 0$, and if $m = k_0 - 1$, then $\eta_{m+1} = \eta_m$.

Crossovers are differentiated from mutations by having two parents needed for recombination. We first present the very general crossover operator, which we subsequently specialize to achieve different types of crossovers. The general crossover is defined as follows: if $\eta^t = \langle \eta_0, \ldots, \eta_{k_0-1} \rangle$ and $\nu^t = \langle \nu_0, \ldots, \nu_{k_0-1} \rangle$ are to be crossed, the resulting offsprings are:

$$\eta^{t+1} = \langle \eta_0, \ldots, \eta_{n-1}, a \cdot \eta_n + (1-a) \cdot \nu_n, \ldots, a \cdot \eta_m + (1-a) \cdot \nu_m, \eta_{m+1}, \ldots, \eta_{k_0-1} \rangle$$
$$\nu^{t+1} = \langle \nu_0, \ldots, \nu_{n-1}, a \cdot \nu_n + (1-a) \cdot \eta_n, \ldots, a \cdot \nu_m + (1-a) \cdot \eta_m, \nu_{m+1}, \ldots, \nu_{k_0-1} \rangle,$$

where $0 \leq n < m \leq k_0 - 1$.

The problem is to find feasible values for a. Calculation of this range is straightforward when the two constraints that possibly get violated are considered. Let us deal with the offspring η^{t+1}. The constraints that we must deal with are:

$a \cdot \eta_n + (1-a) \cdot \nu_n \geq \eta_{n-1}$ and $a \cdot \eta_m + (1-a) \cdot \nu_m \leq \eta_{m+1}$.

Solving the above, we get the following feasible ranges for different cases of η and ν at locations n, m, shown in table 1. In addition, when $n = 0$, we must take $\eta_{n-1} = 0$ and when $m = k_0 - 1$, we must take $\eta_{m+1} = 1$.

Table 1
Feasible ranges for the closed crossover operations.

Case	$\eta_m < \nu_m$	$\eta_m = \nu_m$	$\eta_m > \nu_m$
$\eta_n < \nu_n$	$[\frac{\nu_m - \eta_{m+1}}{\nu_m - \eta_m}, \frac{\nu_n - \eta_{n-1}}{\nu_n - \eta_n}]$	$[-\infty, \frac{\nu_n - \eta_{n-1}}{\nu_n - \eta_n}]$	$[-\infty, min(\frac{\nu_n - \eta_{n-1}}{\nu_n - \eta_n}, \frac{\eta_{m+1} - \nu_m}{\eta_m - \nu_m})]$
$\eta_n = \nu_n$	$[\frac{\nu_m - \eta_{m+1}}{\nu_m - \eta_m}, \infty]$	$[-\infty, \infty]$	$[-\infty, \frac{\eta_{m+1} - \nu_m}{\eta_m - \nu_m}]$
$\eta_n > \nu_n$	$[max(\frac{\eta_{n-1} - \nu_n}{\eta_n - \nu_n}, \frac{\nu_m - \eta_{m+1}}{\nu_m - \eta_m}), \infty]$	$[\frac{\eta_{n-1} - \nu_n}{\eta_n - \nu_n}, \infty]$	$[\frac{\eta_{n-1} - \nu_n}{\eta_n - \nu_n}, \frac{\eta_{m+1} - \nu_m}{\eta_m - \nu_m}]$

- **Multi–point crossover**. The general crossover reduces to a multi–point crossover when we select a as close to zero as possible from the feasible range, thus allowing gene exchange. In general, we need an even number of crossover points. However, for an odd number, we assume the last point to be at the end of the chromosome $m = k_0 - 1$. For example, we get a one–point crossover when we take $n > 0$ and $m = k_0 - 1$. To generate crossovers with k cross–points, we generate k random points on the range $[0, k_0 - 1]$, we order them, we assume an additional point at $k_0 - 1$ if k is odd, and we sequentially perform $\lceil k/2 \rceil$ two–point crossovers, in a random order on disjoint neighboring points.

- **Uniform crossover**. The general crossover reduces to the uniform crossover of DeJong and Spears when we select a as close to zero as possible from the feasible range, and then perform the multi–point crossover with $k = k_0$.

- **Arithmetical crossover**. The idea of the arithmetical crossover operators is to combine genes instead of exchanging them. Therefore, we may generate multi–point arithmetical crossovers by following the ideas of the multi–point crossover and selecting $a \neq 0$. In particular, when we select a as close to $1/2$ as possible from the feasible range, we will be creating new genes that are averages of the parent genes. Moreover, when $n = 0$ and $m = k_0 - 1$, we are performing the arithmetical crossover on the whole chromosomes. In this case, we may select any $a \in [0, 1]$ without the feasible range computation due to properties of convex spaces.

Finally, we need to discuss handling the possible additional constraints: $\eta_0 = 0$ and $\eta_{k_0-1} = 1$, arising in data reduction when specific sequences of signs are detected (see section 2). A possible approach would be to additionally amend our operators to deal with these constraints. However, a much simpler and uniform approach is to learn instead the sequence that excludes such boundary points, while only using the original sequence to evaluate the chromosomes. For example, suppose that after data reduction the following constraints hold: $0 = \eta_0 \leq \eta_1 \ldots \eta_{k_0-1} \leq 1$. In this case, the algorithm is run with a virtual vector $\eta_1, \ldots, \eta_{k_0-1}$ that is to satisfy the general constraint: $0 \leq \eta_1 \ldots \leq \eta_{k_0-1} \leq 1$, while the chromosomes are evaluated as if the solution vector were $\langle 0, \eta_1, \ldots, \eta_{k_0-1} \rangle$. This way we not only reduce the search space but also easily preserve integrity of the operators.

3.2.1. Initialization

Because of general lack of information on the function being sought, we decided to use random initialization according to the following algorithm: generate a vector of length k_0 initialized with random values on the range $[0, 1]$ and then sort the entries. Again, this length may be decreased by one or two if the additional equality constraints are imposed.

3.2.2. Algorithm

The algorithm follows that of the classical genetic algorithm. First, a population of a fixed size (normally 50) is initialized and evaluated. Then, the iterative simulation begins. At each iteration, first stochastically better samples are selected with chances proportional to their qualities as the sought solutions. Then, our operators are applied to some chromosomes of the newly selected population. These applications are stochastic and based on some fixed probabilities assigned to different operators. In general, these probabilities may be adaptable as well. We did not experiment with these, nor we tried to evaluate the applicability of different operators here. After these changes occur, the new chromosomes are reevaluated, and the iterative cycle continues. It stops when a chromosome of some desired quality is found, assuming we have such global criteria, or when some resources are exhausted (*e.g.*, allowed time has elapsed). Notice that an important feature of this algorithm is that it can return the currently found optimum at any time.

4. EXPERIMENTS AND RESULTS

For testing the algorithm, we decided to use three artificial data sets generated according to some commonly known distributions: stepwise, logistic, and exponential. In all cases, we used a single value for β, $(=-1)$, $\phi_0(t) = 0.02t$, and x_i as *i.i.d.* Bernoulli random variables with $P(x_i = 1) = 1/2$. The assumed $\lambda(t)$ distributions were as follows:

1. Stepwise $\lambda(t) = 0$ if $t \leq \theta$ and 1 otherwise. We used $\theta = 2$.
2. Logistic $\lambda(t) = \frac{te^{\omega t}}{(e^{\alpha} + te^{\omega t})}$. We used $\alpha = 5$ and $\omega = 1$.
3. Exponential $\lambda(t) = 1 - e^{\omega t}$. We used $\omega = -0.25$.

For the first two test cases, we generated artificial 300 data points, and in both cases the data reduction technique (see section 2) yielded discretization vectors (with 66 and

69 values) that had the additional constraints: $0 = \eta_0$ and $\eta_{k_0-1} = 1$. For the last case, we generated artificial 600 data points. Here, the data reduction technique produced discretization vector of length 141, with only one additional constraint on the right: $\eta_{k_0-1} = 1$.

To establish some reference solutions, we solved each of these problems assuming that we knew all of the parameters of the data generation process, the functions and their parameters, and selected the t values that corresponded to those data points selected into the reduced data set. With those values, we computed the corresponding $\eta(t)$ and then used these to find the value $F(\eta(t))$. The optima generated in this way are presented in table 2, and in figure 1 we show the functions as discretely represented by the data. Because the limited number of data points may not perfectly fit the distribution from which the data was generated, this approach does not guarantee finding an optimal solution. Nevertheless, we found it still much better than a simple parametric method that would only assume the knowledge of the function and was to maximize the objective function by adjusting the parameters only.

Table 2
Tests summary.

Test case	Number of data points	Length of of discretized vector	Generated maximum	From data maximum
Stepwise	300	66	-1391.3938	-1391.4994
Logistic	300	69	-1408.1364	-1410.8669
Exponential	600	141	-3225.1811	-3229.1907

Finally, we ran our algorithm on these three cases and observed both the generated vectors and their quality. In this case, we did not make any assumptions about the shape of the solution except for the constraints. The results are combined in table 2 and the derived discretized function are presented in figure 2. To our surprise, our algorithm found the best optimal values. These optima correspond to quite disturbed functions, as seen in figure 2. Given this, we may argue that it would be impossible for any parametric method, especially those not assuming the knowledge of the type of the function, to generate these solutions.

For all cases, our algorithm ran for 5,000 iterations, which took about two hours on a SPARCstation2. However, as we show in table 3, our algorithm could find high quality solutions after quite fewer iterations — the remaining ones were used to fine–tune the solutions. For example, only ten iterations on the logistic data were sufficient to generate a solution better than any of the two reference solutions.

5. CONCLUSIONS

We presented here an algorithm for survival analysis in biostatistics. The problem itself can be described as a numerical optimization problem with highly complex objective function and fairly complex linear constraints. The algorithm is based on genetic algorithms, but differs by the operators utilized. Our operators are designed to be closed

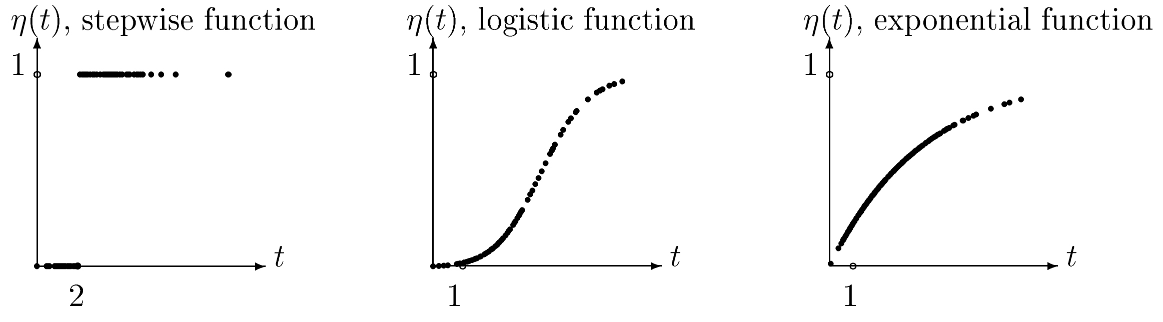

Figure 1. Discretized functions directly recovered from data under the assumption of precisely knowing the generation functions.

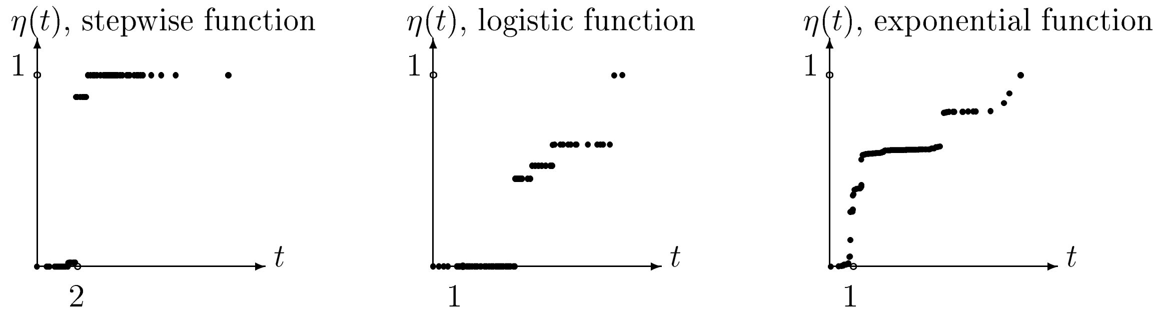

Figure 2. Discretized functions found by the algorithm.

in the inequality–constraints–restricted convex space. This guarantees only feasible solutions to be generated and dramatically reduces the search space. Additional dramatic search space reduction is achieved by detecting and removing the equality constraints *a priori*.

The experiments presented here indicate that appropriately modified genetic algorithms can be successfully used for such complex problems, which otherwise cannot be solved due to high nonlinearity, large dimensionality, and highly constrained structure. Since the algorithm uses the same mechanisms as genetic algorithms (except for operators), it is also a potential candidate for parallel implementations.

In these experiments we assumed the β parameter was fixed and scalar. The next step would be to relax this assumption and let this vector value be found along with the other values. Combining unknown β and equality removal seems contradictory and troublesome since the data reduction process depends on the β value. However, for many practical problems, like those considered here, $x_i \in \{0, 1\}$. Then, only a change in the sign of β invalidates the reduction. Initial experiments suggest that this observation leads to a high

Table 3
Solutions found after fewer iterations.

Number of iterations	Found optimum	CPU time
10	-1410.2786	15.5 sec
50	-1408.9285	64.2 sec
500	-1408.3609	606.6 sec
5000	-1408.1364	5934.2 sec

quality general algorithm capable of optimizing both the function λ and $\vec{\beta}$ simultaneously.

6. BIBLIOGRAPHY

1 Caruana, R.A. & Schaffer, J.D., *"Representation and Hidden Bias: Grey vs. Binary Coding for Genetic Algorithms"*, *Proceedings of the Fifth International Conference on Machine Learning*, Morgan Kaufmann, 1988.

2 Davis, L., (Editor), *Genetic Algorithms and Simulated Annealing*, Pitman, London, 1987.

3 Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

4 Holland, J., *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

5 Janikow, C. & Michalewicz, Z., *"An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms"*, *Proceedings of the Fourth International Conference on genetic Algorithms*, pp.31–36. Morgan Kaufmann.

6 Michalewicz, Z. & Janikow, C., *"GENOCOP: A Genetic Algorithm for Numerical Optimization Problems with Linear Constraints"*, to appear in *Communications of ACM.*

7 Richardson, J.T., Palmer, M.R., Liepins, G. & and Hilliard, M., *"Some Guidelines for Genetic Algorithms with Penalty Functions"*, in *Proceedings of the Third International Conference on genetic Algorithms*, pp.191–197. Morgan Kaufmann.

8 Siedlecki, W. & Sklanski, J., *"Constrained Genetic Optimization via Dynamic Reward–Penalty Balancing and Its Use in Pattern Recognition"*, in *Proceedings of the Third International Conference on genetic Algorithms*, pp.141–150. Morgan Kaufmann.

9 Turnbull, B.W., Luo, X. & Cai, H., *"Regression for censored data in case of time lag"*, preprint, 1992.