Handling Constraints in Genetic Algorithms

Zbigniew Michalewicz
Department of Computer Science
University of North Carolina
Charlotte, NC 28223, USA

Cezary Z. Janikow*
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599, USA

Abstract

The major difficulty in applicability of genetic algorithms to various optimization problems is the lack of general methodology for handling constraints. This paper discusses a new such methodology and presents results from the experimental system GENOCOP (for GEnetic algorithm for Numerical Optimization for COnstrainted Problems). The system not only handles any objective function with any set of linear constraints, but also effectively reduces the search space. The results indicate that this approach is superior to traditional methods when applied to the nonlinear transportation problem.

1 INTRODUCTION

In this paper we present a new approach to solving numerical optimization problems with linear constraints, based on genetic algorithms. Our new methodology seems to fit between the OR and AI approaches. First, it uses the OR technique for problem representation, *i.e.* the formulation of constraints is quantitative. On the other hand, our method uses a genetic algorithm, which is considered as an AI based search method.

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, all traditional GA approaches do not use this fact and rather emply 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 the proposed approach, linear constraints are divided into equalities and inequalities. The equalities are eliminated at the start, together with an equal number of problem variables; this action removes also part of the space to be searched. The remaining constraints, in the form of linear inequalities, form a convex set which must be searched for a solution. Its convexity ensures that linear combinations of solutions always yield feasible solutions — a property used throughout this approach. The inequalities are used to generate bounds for any given variable: such bounds are dynamic as they depend on the values of the other variables and can be efficiently computed.

The full discussion on the proposed approach will appear elsewhere (see [15]); in this paper we explain the main idea of the proposed approach and present the first results.

2 THE CONSTRAINTS PROBLEM IN GENETIC ALGORITHMS

Three different approaches to the constraints problem in genetic algorithms have previously been proposed. The first two involve transforming potential solutions of the problem into a form suitable for a genetic algorithm and using penalty functions or applications of "decoders" or "repair" algorithms. The third approach involves modifying the genetic algorithm to suit the problem by using new data structures and new genetic operators.

In this section, after defining the class of linearly constrained optimization problems, we briefly discuss these three approaches in turn.

*Present address: Department of Mathematics and Computer Science, University of Missouri, St. Louis, Missouri 63121–4499

2.1 THE PROBLEM

We consider a class of optimization problems that can be formulated as follows:

Optimize a function $f(x_1, x_2, \ldots, x_q)$, subject to the following sets of linear constraints:

1. Domain constraints: $l_i \leq x_i \leq u_i$ for $i = 1, 2, \ldots, q$. We write $\vec{l} \leq \vec{x} \leq \vec{u}$, where $\vec{l} = \langle l_1, \ldots, l_q \rangle$, $\vec{u} = \langle u_1, \ldots, u_q \rangle$, $\vec{x} = \langle x_1, \ldots, x_q \rangle$.

2. Equalities: $A\vec{x} = \vec{b}$, where $\vec{x} = \langle x_1, \ldots, x_q \rangle$, $A = (a_{ij})$, $\vec{b} = \langle b_1, \ldots, b_p \rangle$, $1 \leq i \leq p$, and $1 \leq j \leq q$ (p is the number of equations).

3. Inequalities: $C\vec{x} \leq \vec{d}$, where $\vec{x} = \langle x_1, \ldots, x_q \rangle$, $C = (c_{ij})$, $\vec{d} = \langle d_1, \ldots, d_m \rangle$, $1 \leq i \leq m$, and $1 \leq j \leq q$ (m is the number of inequalities).

This formulation is general enough to handle a large class of standard Operations Research optimization problems with linear constraints and any objective function. The example considered later, the nonlinear transportation problem, is one of many problems in this class.

2.2 PENALTY FUNCTIONS

One way of dealing with candidate solutions that violate the constraints is to generate potential solutions without considering the constraints and then penalizing them by decreasing the "goodness" of the evaluation function. In other words, a constrained problem is transformed to 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. Davis discusses this problem in [2] listing disadvantages of using high, moderate, or light penalties:

> "If one incorporates a high penalty into the evaluation routine and the domain is one in which production of an individual violating the constraint is likely, one runs the risk of creating a genetic algorithm that spends most of its time evaluating illegal individuals. Further, it can happen that when a legal individual is found, it drives the others out and the population converges on it without finding better individuals, since the likely paths to other legal individuals require the production of illegal individuals as intermediate structures, and the penalties for violating the constraint make it unlikely that such intermediate structure will reproduce. If one imposes moderate penalties, the system may evolve individuals that violate the constraint but are rated better than those that do not because the rest of the evaluation function can be satisfied better by accepting the moderate constraint penalty than by avoiding it".

In [18] and [16] the authors present the most recent approaches for using penalty functions in GAs for constrained optimization problems. However, the paper by Siedlecki and Sklansky [18] discusses a particular constrained optimization problem and the proposed method is problem specific. The paper by Richardson, Palmer, Liepins, and Hillard [16] examines the penalty approach discussing the strengths and weaknesses of various penalty function formulations and illustrate a technique for solving three dimensional constrained problem. However, in the last section of their article the authors say:

> "The technique used to solve the three dimensional problem described above can't be generalized since quantities like the trend and maximum derivative are seldom available".

We do not believe this to be a promising direction. For a heavily constrained problem, such as the transportation problem, the probability of generating an infeasible candidate is too great to be ignored. The technique based on penalty functions, at the best, seems to work reasonably well for narrow classes of problems and for few constraints, or for cases of non-essential constraints.

2.3 DECODERS AND REPAIR ALGORITHMS

Another approach concentrates on the use of special representation mappings (decoders) which guarantee (or at least increase the probability of) the generation of a feasible solution and on the application of special repair algorithms to "correct" any infeasible solutions so generated. However, decoders are frequently computationally intensive to run [2], not all constraints can be easily implemented this way, and the resulting algorithm must be tailored to the particular application. The same is true for repair algorithms. Again, we do not believe this is a promising direction for incorporating constraints into genetic algorithms. Repair algorithms and decoders may work reasonably well but are highly problem specific, and the chances of building a general genetic algorithm to handle different types of constraints based on this principle seem to be slim.

2.4 SPECIALIZED DATA STRUCTURES AND GENETIC OPERATORS

The last and relatively new approach for incorporating constraints in genetic algorithms is to introduce richer data structures together with an appropriate family of applicable "genetic" operators which can "hide" the constraints presented in the problem (see [10], [12]).

Several experiments ([7], [13], [19], [20]) indicate the usefulness of this approach, but it is not always possible, for an arbitrary set of constraints, to develop an efficient data structure hiding such constraints. In addition, such structures require specialized genetic operators to maintain feasibility. Such extensions lack also the theoretical basis enjoyed by the classical genetic operators. Despite these objections, experimental results suggest that this approach may be promising. However, the particular choice of representation and operators must still be tailored to the specific problem to be solved.

3 A NEW METHODOLOGY: THE GENOCOP SYSTEM

The proposed methodology provides a way of handling constraints that is both general and problem independent. It combines some of the ideas seen in the previous approaches, but in a totally new context. The main idea behind this approach lies in (1) an elimination of the equalities present in the set of constraints, and (2) careful design of special "genetic" operators, which guarantee to keep all "chromosomes" within the constrained solution space. This can be done very efficiently for linear constraints and while we do not claim these results extend easily to nonlinear constraints, the former class contains many interesting optimization problems.

A full description of the GENOCOP system is presented in [15]; below we give a small example which should provide some insight into the proposed methodology.

Let us assume we wish to minimize a function of six variables:

$$f(x_1, x_2, x_3, x_4, x_5, x_6)$$

subject to the following constraints:

$$\begin{aligned}&x_1 + x_2 + x_3 = 5\\&x_4 + x_5 + x_6 = 10\\&x_1 + x_4 = 3\\&x_2 + x_5 = 4\\&x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0,\ x_5 \geq 0,\\&x_6 \geq 0.\end{aligned}$$

We can take an advantage from the presence of four independent equations and express four variables as functions of the remaining two:

$$\begin{aligned}&x_3 = 5 - x_1 - x_2\\&x_4 = 3 - x_1\\&x_5 = 4 - x_2\\&x_6 = 3 + x_1 + x_2\end{aligned}$$

We have reduced the original problem to the optimization problem of a function of two variables x_1 and x_2:

$$\begin{aligned}g(x_1, x_2) = f(&x_1, x_2, (5 - x_1 - x_2),\\&(3 - x_1), (4 - x_2), (3 + x_1 + x_2)).\end{aligned}$$

subject to the following constraints (inequalities only):

$$\begin{aligned}&x_1 \geq 0,\ x_2 \geq 0\\&5 - x_1 - x_2 \geq 0\\&3 - x_1 \geq 0\\&4 - x_2 \geq 0\\&3 + x_1 + x_2 \geq 0\end{aligned}$$

These inequalities can be further reduced to:

$$\begin{aligned}&0 \leq x_1 \leq 3\\&0 \leq x_2 \leq 4\\&x_1 + x_2 \leq 5\end{aligned}$$

This would complete the first step of our algorithm: elimination of equalities.

Now let us consider a single point from the search space, *e.g.* $\vec{x} = \langle x_1, x_2 \rangle = \langle 1.8, 2.3 \rangle$. If we try to change the value of variable x_1 without changing the value of x_2 (*uniform mutation*), the variable x_1 can take any value from the range: $[0,\ 5 - x_2] = [0,\ 2.7]$. Additionally, if we have two points within search space, $\vec{x} = \langle x_1, x_2 \rangle = \langle 1.8, 2.3 \rangle$ and $\vec{x'} = \langle x_1', x_2' \rangle = \langle 0.9, 3.5 \rangle$, then any linear combination $a\vec{x} + (1 - a)\vec{x'}$, $0 \leq a \leq 1$, would yield a point within search space, *i.e.* all constraints must be satisfied (*whole arithmetical crossover*). Therefore, both examples of operators would not move a vector outside the constrained solution space.

The above example explains the main idea behind the GENOCOP system. Linear constraints were of two types: equalities and inequalities. We first eliminated all the equalities, reducing the number of variables and appropriately modifying the inequalities. Reducing the set of variables both decreased the length of the representation vector and reduced the search space. Since we were left with only linear inequalities, the search space was convex — which, in the presence of closed operators, could be searched efficiently. The problem then became one of designing such closed operators. We achieved this by defining them as being context-dependent, that is dynamically adjusting to the current context. Such operators, used in GENOCOP system, are quite different from the classical ones. This is because:

1. We deal with a real valued space R^q, where a solution is coded as a vector with floating point type components,
2. The genetic operators are dynamic, *i.e.* a value of a vector component depends on the remaining values of the vector,
3. Some genetic operators are non-uniform, *i.e.* their action depends on the age of the population.

In the convex space, the value of the i-th component of a feasible solution $\vec{s} = \langle v_1, \ldots, v_m \rangle$ is always in some (dynamic) range $[l, u]$; the bounds l and u depend on the other vector's values $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_m$, and the set of inequalities.

Before we describe the operators, we present two important characteristics of convex spaces (due to the linearity of the constraints, the solution space is always a convex space $\mathcal{S}$), which play an essential role in the definition of these operators:

1. For any two points s_1 and s_2 in the solution space $\mathcal{S}$, the linear combination $a \cdot s_1 + (1-a) \cdot s_2$, where $a \in [0,1]$, is a point in $\mathcal{S}$.
2. For every point $s_0 \in \mathcal{S}$ and any line p such that $s_0 \in p$, p intersects the boundaries of $\mathcal{S}$ at precisely two points, say $l_p^{s_0}$ and $u_p^{s_0}$.

Since we are only interested in lines parallel to each axis, to simplify the notation we denote by $l_{(i)}^s$ and $u_{(i)}^s$ the i-th components of the vectors l_p^s and u_p^s, respectively, where the line p is parallel to the axis i. We assume further that $l_{(i)}^s \leq u_{(i)}^s$.

Because of intuitional similarities, we cluster the operators into the standard two classes: mutation and crossover. The proposed crossover and mutation operators use the two properties to ensure that the offspring of a point in the convex solution space $\mathcal{S}$ belongs to $\mathcal{S}$. However, some operators (*e.g.* non-uniform mutation) have little to do with GENOCOP methodology: they have other "responsibilities" like fine tuning and prevention of premature convergence. For a detailed discussion on these topics, the reader is referred to [9] and [11].

Mutation group: Mutations are quite different from the traditional one with respect to both the actual mutation (a gene, being a floating point number, is mutated in a dynamic range) and to the selection of an applicable gene. A traditional mutation is performed on static domains for all genes. In such a case the order of possible mutations on a chromosome does not influence the outcome. This is not true anymore with the dynamic domains. To solve the problem we proceed as follows: we randomly select $p_{um} \cdot \textit{pop\_size}$ chromosomes for uniform mutation, $p_{bm} \cdot \textit{pop\_size}$ chromosomes for boundary mutation , and $p_{nm} \cdot \textit{pop\_size}$ chromosomes for non-uniform mutation (all with possible repetitions), where p_{um}, p_{bm}, and p_{nm} are probabilities of the three mutations defined below. Then, we perform these mutations in a random fashion on the selected chromosome.

- **uniform mutation** selects a random gene k of the chromosome $s_v^t = \langle v_1, \ldots, v_m \rangle$: the result of this mutation is a vector $s_v^{t+1} = \langle v_1, \ldots, v'_k, \ldots, v_m \rangle$, where v'_k is a random value (uniform probability distribution) from the range $[l_{(k)}^{s_v^t}, u_{(k)}^{s_v^t}]$. The dynamic values $l_{(k)}^{s_v^t}$ and $u_{(k)}^{s_v^t}$ are easily calculated from the set of constraints (inequalities).
- **boundary mutation** is a variation of the uniform mutation with v'_k being either $l_{(k)}^{s_v^t}$ or $u_{(k)}^{s_v^t}$, each with equal probability.
- **non-uniform mutation** is one of the operators responsible for the fine tuning capabilities of the system. It is defined as follows: if $s_v^t = \langle v_1, \ldots, v_m \rangle$ is a chromosome and the element v_k was selected for this mutation from the set of movable genes, the result is a vector $s_v^{t+1} = \langle v_1, \ldots, v'_k, \ldots, v_m \rangle$, with $k \in \{1, \ldots, n\}$, and

$$v'_k = \begin{cases} v_k + \triangle(t, u_{(k)}^{s_v^t} - v_k) & \text{if a random digit is 0} \\ v_k - \triangle(t, v_k - l_{(k)}^{s_v^t}) & \text{if a random digit is 1} \end{cases}$$

  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:

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

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.

Crossover group: Chromosomes are randomly selected in pairs for application of the crossover operators according to appropriate probabilities.

- **simple crossover** is defined as follows: if $s_v^t = \langle v_1, \ldots, v_m \rangle$ and $s_w^t = \langle w_1, \ldots, w_m \rangle$ are crossed after the k-th position, the resulting offspring are: $s_v^{t+1} = \langle v_1, \ldots, v_k, w_{k+1}, \ldots, w_m \rangle$ and $s_w^{t+1} = \langle w_1, \ldots, w_k, v_{k+1}, \ldots, v_m \rangle$. Note that the only permissible split points are between individual floating points (using float representation it is impossible to split anywhere else).

  However, such operator may produce offspring outside of the convex solution space $\mathcal{S}$. To avoid this problem, we use the property of the convex spaces saying, that there exist $a \in [0,1]$ such that

  $$s_v^{t+1} = \langle v_1, \ldots, v_k, w_{k+1} \cdot a + v_{k+1} \cdot (1-a), \\ \ldots, w_m \cdot a + v_m \cdot (1-a) \rangle \in \mathcal{S}$$

  and

  $$s_w^{t+1} = \langle w_1, \ldots, w_k, v_{k+1} \cdot a + w_{k+1} \cdot (1-a), \\ \ldots, v_m \cdot a + w_m \cdot (1-a) \rangle \in \mathcal{S}$$

  The only question to be answered yet is how to find the largest a to obtain the greatest possible information exchange: due to the real interval, we cannot perform an extensive search. In GENOCOP we implemented a binary search (to some depth only for efficiency). Then, a takes the largest appropriate value found, or 0 if no value satisfied the constraints. The necessity for such actions is small in general and decreases rapidly over the life of the population. Note, that the value of a is determined separately for each single arithmetical crossover and each gene.

- **single arithmetical crossover** is defined as follows: if $s_v^t = \langle v_1, \ldots, v_m \rangle$ and $s_w^t = \langle w_1, \ldots, w_m \rangle$ are to be crossed, the resulting offspring are $s_v^{t+1} = \langle v_1, \ldots, v'_k, \ldots, v_m \rangle$ and $s_w^{t+1} = \langle w_1, \ldots, w'_k, \ldots, w_m \rangle$, where $k \in [1,m]$, $v'_k = a \cdot w_k + (1-a) \cdot v_k$, and $w'_k = a \cdot v_k + (1-a) \cdot w_k$. Here, a is a dynamic parameter calculated in the given context (vectors s_v, s_w) so that the operator is closed (points x_v^{t+1} and x_w^{t+1} are in the convex constrained space $\mathcal{S}$). Actually, a is a random choice from the following range:

  $$\begin{array}{ll} [max(\frac{l^{s_w}_{(k)} - w_k}{v_k - w_k}, \frac{u^{s_v}_{(k)} - v_k}{w_k - v_k}), min(\frac{l^{s_v}_{(k)} - v_k}{w_k - v_k}, \frac{u^{s_w}_{(k)} - w_k}{v_k - w_k})] & \\ & \text{if } v_k > w_k \\ [0,0] & \text{if } v_k = w_k \\ [max(\frac{l^{s_v}_{(k)} - v_k}{w_k - v_k}, \frac{u^{s_w}_{(k)} - w_k}{v_k - w_k}), min(\frac{l^{s_w}_{(k)} - w_k}{v_k - w_k}, \frac{u^{s_v}_{(k)} - v_k}{w_k - v_k})] & \\ & \text{if } v_k < w_k \end{array}$$

  To increase the applicability of this operator (to ensure that a will be non–zero, which actually always nullifies the results of the operator) it is wise to select the applicable gene as a random choice from the intersection of movable genes of both chromosomes. Note again, that the value of a is determined separately for each single arithmetical crossover and each gene.

- **whole arithmetical crossover** is defined as a linear combination of two vectors: if s_v^t and s_w^t are to be crossed, the resulting offspring are $s_v^{t+1} = a \cdot s_w^t + (1-a) \cdot s_v^t$ and $s_w^{t+1} = a \cdot s_v^t + (1-a) \cdot s_w^t$ This operator uses a simpler static system parameter $a \in [0..1]$, as it always guarantees closedness (according to characteristic (1) of convex spaces given earlier in this section).

4 EXPERIMENTS AND RESULTS

For experiments we selected the following problem (transportation problem) of forty nine variables:

minimize $f(\vec{x}) = \sum_{i=1}^{49} g(x_i)$,

subject to the following equality constraints:

$$\begin{array}{l} x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 27 \\ x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} = 28 \\ x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} = 25 \\ x_{22} + x_{23} + x_{24} + x_{25} + x_{26} + x_{27} + x_{28} = 20 \\ x_{29} + x_{30} + x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 20 \\ x_{36} + x_{37} + x_{38} + x_{39} + x_{40} + x_{41} + x_{42} = 20 \\ x_{43} + x_{44} + x_{45} + x_{46} + x_{47} + x_{48} + x_{49} = 20 \end{array}$$

$$\begin{array}{l} x_1 + x_8 + x_{15} + x_{22} + x_{29} + x_{36} + x_{43} = 20 \\ x_2 + x_9 + x_{16} + x_{23} + x_{30} + x_{37} + x_{44} = 20 \\ x_3 + x_{10} + x_{17} + x_{24} + x_{31} + x_{38} + x_{45} = 20 \\ x_4 + x_{11} + x_{18} + x_{25} + x_{32} + x_{39} + x_{46} = 23 \end{array}$$

$$x_5 + x_{12} + x_{19} + x_{26} + x_{33} + x_{40} + x_{47} = 26$$
$$x_6 + x_{13} + x_{20} + x_{27} + x_{34} + x_{41} + x_{48} = 25$$
$$x_7 + x_{14} + x_{21} + x_{28} + x_{35} + x_{42} + x_{49} = 26$$

Experiments were made for six nonlinear cost functions g (for a full discussion on the selection and classification of these functions, see [13]); their graphs are presented in Figure 1.

Figure 1: Six cost functions A – F.

There are thirteen independent and one dependent equations here; therefore, we eliminate thirteen variables: $x_1, \ldots, x_8, x_{15}, x_{22}, x_{29}, x_{36}, x_{44}$. All remaining variables are renamed $y_1, \ldots, y_{36}$. Each of these variables has to satisfy four two–sided inequalities, which result from the initial domains and our transformations. Now, each chromosome is a float vector $\langle y_1, \ldots, y_{36} \rangle$.

For comparative experiments we implemented all previously discussed GA approaches to the constraint problem (penalties, decoders, and specialized data structures), the GENOCOP system, and we used a version (the *student version*) of GAMS, a package for the construction and solution of large and complex mathematical programming models [1].

The experiments with penalty functions and decoders were not successful. For example, in experiments with penalty functions the evaluation function *Eval* was composed of the optimization function f and the penalty P:

$$Eval(\vec{x}) = f(\vec{x}) + P,$$

For our experiments we followed a suggestion (see [16]) to start with relaxed penalties and to tight them as the run progresses. We used

$$P = k \cdot (\tfrac{t}{T})^p \cdot \overline{f} \cdot \textstyle\sum_{i=1}^{14} d_i$$

where $\overline{f}$ is the average fitness of the population at the given generation t, k and p are parameters, T is the maximum number of generations, and d_i returns the "degree of constraint violation".

We experimented with various values of p (close to 1), k (close to $\frac{1}{14}$, where 14 is total number of equality constraints; the static domain constraints were naturally satisfied by a proper representation), and $T = 8000$. However, this method did not lead to feasible solutions: in over 1200 runs (with different seeds for random number generator and various values for parameters k and p) the best chromosomes (after 8000 generations) violated at least 3 constraints in a significant way. The best solution was "far away" from the feasible solution for transportation problem.

Only GENOCOP system and a specialized data structures system (GENETIC-2, based on matrix data structure as a chromosome) gave feasible results: these are reported in [15]. In general, they are quite similar (GENETIC-2 was slightly better); however, note that the matrix approach was tailored to the specific (transportation) problem, whereas GENOCOP is problem independent and works without any hard-coded domain knowledge. In other words, while one might expect the GENOCOP to perform similarly well for other constrained problems, the GENETIC-2 cannot be used at all.

The results and comparison of GENOCOP and GAMS (summarized in Figure 2) indicate both usefulness of our method in the presence of many constraints and its superiority over some standard systems on nontrivial problems.

GENOCOP was run for 8000 iterations, with the population size equal 40. A single run of 8000 iterations took 2:28 CPU sec on Cray Y-MP. GAMS was run on an Olivetti 386 with a math co-processor and a single run took 1:20 sec.

Function	GAMS	GENOCOP	% difference
A	96.00	24.15	+297.52%
B	1141.60	205.60	+455.25%
C	2535.29	2571.04	−1.41%
D	565.15	480.16	+17.70%
E	208.25	204.82	+1.67%
F	43527.54	119.61	+36291.22%

Figure 2: GENOCOP versus GAMS: the results for the 7×7 problem.

5 CONCLUSIONS

After considering alternative ways for handling constraints in genetic algorithms for optimization problems, we proposed a new method for handling linear constraints. This new methodology should enable such constrained problems with difficult objective functions to be solved without incurring the heavy computational overhead associated with frequent constraint checking and without a need for designing a specific system's architecture.

The equality constraints are handled immediately by eliminating some variables, at one stroke removing constraints and reducing the search space. The inequalities are then processed to provide a set of bounds for each of the remaining variables considered in isolation. These bounds are dynamic in that they depend on the values of other variables of the current solution. Since the GA modifies each variable independently, this is not computationally complex.

Our results suggest that the method is useful as compared to the standard methods, and may lead towards the solution of some difficult Operations Research problems.

It is relatively easy to extend the GENOCOP system to handle discrete variables (nominal, linear, and boolean). Such variables would undergo different mutations and crossovers, to keep the solution vector within the constrained space. Also, it is possible to extend the GENOCOP system to handle nonlinear constraints provided that the search space is convex.

Acknowledgments: This research was supported by a grant from the North Carolina Supercomputing Center.

References

[1] Brooke, A., Kendrick, D., and Meeraus, A., *GAMS: A User's Guide*, The Scientific Press, 1988.

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

[3] De Jong, K.A., *Genetic Algorithms: A 10 Year Perspective*, in [5], pp.169–177.

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

[5] Grefenstette, J.J., (Editor), Proceedings of the First International Conference on Genetic Algorithms, Pittsburg, Lawrence Erlbaum Associates, Publishers, July 24–26, 1985.

[6] Grefenstette, J.J., (Editor), Proceedings of the Second International Conference on Genetic Algorithms, MIT, Cambridge, Lawrence Erlbaum Associates, Publishers, July 28–31, 1987.

[7] Groves, L., Michalewicz, Z., Elia, P., Janikow, C., *Genetic Algorithms for Drawing Directed Graphs*, Proceedings of the Fifth International Symposium on Methodologies of Intelligent Systems, Knoxville, pp.268–276, October 25–27, 1990.

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

[9] Janikow, C., and Michalewicz, Z., *Specialized Genetic Algorithms for Numerical Optimization Problems*, Proceedings of the International Conference on Tools for AI, Washington, pp.798–804, November 6–9, 1990.

[10] Michalewicz, Z., Vignaux, G.A., Groves, L., *Genetic Algorithms for Optimization Problems*, Proceedings of the 11-th NZ Computer Conference, Wellington, New Zealand, pp.211–223, August 16–18, 1989.

[11] Michalewicz, Z. and Janikow, C., *Genetic Algorithms for Numerical Optimization*, Statistics and Computing, Vol.1, No.1, 1991.

[12] Michalewicz, Z., Schell, J., and Seniw, D., *Data Structures + Genetic Operators = Evolution Programs*, UNCC Technical Report, 1991.

[13] Michalewicz, Z., Vignaux, G.A., Hobbs, M., *A Non-standerd Genetic Algorithm for the Nonlinear Transportation Problem*, ORSA Journal on Computing, Vol.3, 1991.

[14] Michalewicz, Z., Krawczyk, J., Kazemi, M., Janikow, C., *Genetic Algorithms and Optimal Control Problems*, Proceedings of the 29th IEEE Conference on Decision and Control, Honolulu, pp.1664–1666, December 5–7, 1990.

[15] Michalewicz, Z. and Janikow, C., *GENOCOP: A Genetic Algorithm for Numerical Optimization Problems with Linear Constraints*, to appear in Communications of ACM, 1991.

[16] Richardson, J.T., Palmer, M.R., Liepins, G., and Hilliard, M., *Some Guidelines for Genetic Algorithms with Penalty Functions*, in [17], pp.191–197.

[17] Schaffer, J., (Editor), Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, Morgan Kaufmann Publishers, June 4–7, 1989.

[18] Siedlecki, W. and Sklanski, J., *Constrained Genetic Optimization via Dynamic Reward–Penalty Balancing and Its Use in Pattern Recognition*, in [17], pp.141–150.

[19] Vignaux, G.A. and Michalewicz, Z., *Genetic Algorithms for the Transportation Problem*, Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems, Charlotte, NC, pp.252–259, October 12–14, 1989.

[20] Vignaux, G.A. and Michalewicz, Z., *A Genetic Algorithm for the Linear Transportation Problem*, IEEE Transactions on Systems, Man, and Cybernetics, Vol.21, No.2, 1991.