Pergamon

Mathl. Comput. Modelling Vol. 23, No. 5, pp. 87–111, 1996

0895-7177/96 $15.00 + 0.00

S0895-7177(96)00014-3

Genetic Algorithms in Constrained Optimization

D. J. REID
Distributed Systems Technology Centre
Department of Computer Science
The University of Queensland
St. Lucia, Queensland 4072, Australia

(Received August 1994; revised and accepted June 1995)

Abstract—The behavior of the two-point crossover operator, on candidate solutions to an optimization problem that is restricted to integer values and by some set of constraints, is investigated theoretically. This leads to the development of new genetic operators for the case in which the constraint system is linear.

The computational difficulty asserted by many optimization problems has lead to exploration of a class of randomized algorithms based on biological adaption. The considerable interest that surrounds these evolutionary algorithms is largely centered on problems that have defied satisfactory illation by traditional means because of badly behaved or noisy objective functions, high dimensionality, or intractable algorithmic complexity. Under such conditions, these alternative methods have often proved invaluable.

Despite their attraction, the applicability of evolutionary algorithms has been limited by a deficiency of general techniques to manage constraints, and the difficulty is compounded when the decision variables are discrete. Several new genetic operators are presented here that are guaranteed to preserve the feasibility of discrete aspirant solutions with respect to a system of linear constraints.

To avoid performance degradation as the probability of finding a feasible and meaningful information exchange between two candidate solutions decreases, relaxations of the modified genetic crossover operator are also proposed. The effective utilization of these also suggests a manipulation of the genetic algorithm itself, in which the population is evanescently permitted to grow beyond its normal size.

Keywords—Discrete optimization, Genetic algorithm, Linear constraints.

1. INTRODUCTION

The genetic algorithms are a class of stochastic relaxation techniques that are applicable to the solution of a wide variety of optimization problems, by emanating the evolutionary behavior of biological systems [1–7]. In contrast to sequential search methods that, at each iteration, generate a single potential solution from the last, a genetic algorithm maintains a large population of candidate solutions. Each population is generated from its predecessor by applying a set of stochastic transition operators.

Evolutionary algorithms differ also in the nature of the vicissitude that manufactures one set of possible solutions from another. While most traditional methods operate directly on the value of an iterate, genetic algorithms operate on a coding of the solution, in the form of a string of symbols of some particular alphabet. This string forms the 'chromosome' that defines a particular member of a population as distinct from its actual value, or 'phenotype.'

The author wishes to acknowledge the support of the Distributed Systems Technology Center, and the Department of Computer Science, the University of Queensland. Particular thanks are extended to M. Orlowska, and also to B. Majewski.

The most significant of these operators can be grouped according to their biologically-inspired functions: selection operators define the ebullience with which each individual reproduces, crossover operators describe the process by which individuals mate to produce offspring, and mutation operators allow occasional alteration in the encoding of a population member. Each population, in turn, undergoes transition by the suitable employment of these operators to produce the next.

The allure of genetic algorithm implementations is enhanced by their conceptual simplicity, and the capacious list of disparate problems in which they have already been utilized. Unlike most sequential search techniques, no gradient information need be presumed, although if available, may still be exploited to enhance performance; problems in which the objective function is not differentiable pose no extra challenge. Furthermore, the maintenance of a population of candidate solutions reduces the potential of convergence to local, rather than global, optima, rendering the genetic algorithm also a vigorous contender for the resolution of a multimodal problem.

The major obstacle in the application of genetic algorithms to many optimization problems is the embracement of the constraint system [3–6]. In most existing approaches, the genetic operators are permitted to generate infeasible candidate solutions, and these are either penalized by a modification of the objective function (penalty function methods), or later corrected (repair algorithms). An alternative is the adoption of a specialized representation of the problem (decoder) to ensure, or magnify the probability of, the manufacture of feasible solutions.

Penalty function methods [2,4,5] discriminate against a possible solution according to its violation of each constraint; the original constrained optimization problem is transformed into a new, unconstrained, problem. If the chosen penalties of solutions are inconsequential compared to their objective function values, the resulting algorithm invests inordinate effort exploring infeasible solutions. Large penalties may cause premature convergence to an unacceptable solution; a feasible solution, once discovered, dominates the mating pool. The remaining, infeasible, candidate solutions are rapidly extinguished, before any other feasible points can be generated. Even the imposition of moderate penalties can be calamitous, by developing solutions that violate constraints and yet are appraised more favorably than other, feasible, solutions [3].

Both decoder mappings and repair algorithms are frequently found to be computationally intensive, difficult to design, and highly problem-specific [3,6]. Furthermore, the employment of a decoder that still allows the possible generation of infeasible solutions necessitates also the use of a repair algorithm or penalty functions, further magnifying the undesirability of taking this approach.

Attention has been more recently focused on the adaption of the genetic operators themselves to preserve feasibility [3]. In doing so, the promise of the constraint system to reduce the size of the search space is fully enjoyed, potentially awarding improved behavior. Moreover, the additional computational cost imposed by these modified operators is more than offset by the would-be burden of evaluating penalty functions, decoders, or repair algorithms.

The second section outlines the problem type that is under consideration, and demonstrates that this is general enough to permit application to a vast and frequently used class of problems. However, the intention here is not to compete with the widely available techniques that already exist for problems possessing a linear or convex/concave objective function [8–16]. Instead, the emphasis is placed on problems for which the objective is ill-defined, noisy, discontinuous, nondifferentiable, or multimodal.

Section 3 is an overview of existing selection policies, and appears here for completeness, as these concepts are referred to later.

The actual behavior of the crossover operator, with regard to the feasible and infeasible solutions that it might generate, is discussed in some detail in Section 4. This theoretical exploration of crossover provides the basis for the design of methods that assure the maintenance of feasibility, and allow an extensive justification of the considerations identified and included in this pursuit.

It should be noted that this discussion is not concerned with the overall behavior of genetic algorithms, but rather with the behavior of a crossover operator on solutions feasible to a given

constraint system. For an understanding of genetic algorithms and the processing of schemata, the reader is referred to [2,17–20].

A basic crossover operator that is guaranteed to generate only feasible solutions, from feasible solutions, is presented in Section 5. The search for such a crossover operation is performed using the relationships that exist between one possibility and the next, thereby avoiding the need to check feasibility by the evaluation of the entire constraint system. These new algorithms are developed using the knowledge gained about the behavior of the simple two-point crossover.

Not all possible crossover operations produce only feasible solutions, and not all others produce none. A class of crossovers exists that produces one feasible and one infeasible point from a given pair of feasible candidate solutions, and under certain circumstances these are found to dominate those that generate only feasible points. With this in mind, highly constrained problems that might otherwise perform poorly can be enticed into faster resolution by exploiting these 'half-feasible' crossover operations, as discussed in Section 6. Given that one of the products of such a crossover must be rejected because of its infeasibility, some attention is also warranted to the choice of a replacement.

Finally, the importance of the genetic mutation operator leads also to the definition of a version that preserves the feasibility of candidate solutions. Again, the design of this algorithm is justified, and its correctness proven.

2. PROBLEM DEFINITION

The class of optimization problems herein regarded are linearly constrained, and in the context of this discussion, the decision variables are assumed to be entirely discrete-valued. By comparison, other methods [3] that consider only real-valued variables use crossovers based on the properties of a convex set in real space, and cannot be readily limited to discrete values. A composite method is easy to envisage for the solution of mixed-integer type problems.

The aspiration is the optimization (either minimization or maximization) of some function

$$f : \mathbf{Z}^n \to \mathbf{R},$$

subject to the constraint system

$$Ax \leq b, \quad l \leq x \leq u, \qquad x \in \mathbf{Z}^n, \tag{P1}$$

and where $A = [a_{i,j}] \in \mathbf{R}^{m \times n}$, $b = [b_i] \in \mathbf{R}^m$, and $l = [l_i]$, $u = [u_i] \in \mathbf{Z}^n$. The objective function f, in the context of this discussion, is typically badly behaved in terms of continuity, differentiability, modulation, or noisiness, and therefore difficult for more traditional techniques to handle. Note also that no equality constraints are included; these may be converted to a system of inequalities of this form with a reduction in the number of variables, and therefore in the size of the search space.

EXAMPLE 1. Consider minimizing a function $f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$, subject to the system of linear constraints

$$\begin{aligned}
x_1 + x_2 \quad + x_4 &= 3, \\
x_3 - x_4 + x_5 &= 1, \\
3x_2 + x_3 \quad + x_6 &= 6, \\
x_2 \quad + x_4 \quad - x_7 &= 1, \\
x_4 - x_5 \quad + x_8 &= 2, \\
x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8 &\geq 0.
\end{aligned}$$

Five of the decision variables may be expressed as functions of the remaining three:

$$\begin{aligned}
x_4 &= 3 - x_1 - x_2, \\
x_5 &= 1 - x_3 + x_4 = 4 - x_1 - x_2 - x_3,
\end{aligned}$$

$$x_6 = 6 - 3x_2 - x_3,$$
$$x_7 = x_2 + x_4 - 1 = 2 - x_1,$$
$$x_8 = 2 - x_4 + x_5 = 3 - x_3.$$

These variables can be eliminated, producing a system of inequalities

$$\begin{aligned} x_1 + x_2 &\leq 3, \\ x_1 + x_2 + x_3 &\leq 4, \\ 3x_2 + x_3 &\leq 6, \\ x_2 &\geq 0, \\ 0 \leq x_1 &\leq 2, \\ 0 \leq x_3 &\leq 3, \end{aligned}$$

and the original objective function is replaced by

$$\bar{f}(x_1, x_2, x_3) = f(x_1, x_2, x_3, 3 - x_1 - x_2, 4 - x_1 - x_2 - x_3, 6 - 3x_2 - x_3, 2 - x_1, 3 - x_3),$$

producing a new problem lower in dimensionality by the number of equality constraints present in the original. ∎

A candidate solution to this problem is expressed to the genetic algorithm as an individual member of the current population; a population represents the family of solutions presently under investigation. The individual itself is a coding of the solution it typifies as a string in some suitable alphabet, and this has often been chosen to be the concatenation of some binary representation of each of the solution's components. Any symbol in this string is referred to as an 'allele,' and any segment that maps to a complete component of the candidate solution is known as a 'gene.'

The representation of the candidate solution in this discussion diverges from the tradition of a binary alphabet. The value of a decision variable is here represented as that number itself, rather than as a binary string, so that each gene contains a single allele, and the population member is a string of integer values. In this way, the solution vector is less removed from its symbolization, so that greater control is afforded over the behavior of the genetic operators with respect to the feasibility conditions. Also because of this, and for notational brevity, the distinction between a solution point and its representation will henceforth become somewhat blurred.

3. SELECTION OF A MATING POOL

The genetic selection operator is an emblematisation of the process of natural selection in biological systems [1,2,7]. Different population members achieve different levels of fitness, and therefore survive to reproduce at varying rates. The genetic algorithm's selection mechanism emulates this discrimination by constructing a 'mating pool,' with each individual represented according to its objective function value. Pairs of candidate solutions will be later chosen from this mating pool to undergo crossover.

The selection operator drives the search towards better individuals, but this may be at the expense of genotypic diversity. An intolerant selection mechanism tends to extinguish all but the most superior candidate solutions, and while this may produce a remarkable enhancement in performance for some problems, the convergence may be premature for others. Conversely, an unexacting selection policy maintains a high heterogeneity of individuals, allowing a more complete search of the feasible set, yet this may be unwarranted in many cases [1,2].

The design of the selection operator is therefore highly dependent on the demeanor of the objective function. For a unimodal problem, a less charitable selection criterion coerces the search into the gradient direction, with lower population diversity, and more a trajectory-oriented temperament. Multimodal problems, requiring a more widely exploring search, are best resolved using more lenient selection policies.

Perhaps the simplest, and one of the most widely employed, of the selection methods that currently exist is 'Proportional Selection,' also known as 'Roulette Wheel Selection,' or 'Stochastic Sampling With Replacement,' and is described here in terms of its application to the maximization problem. Each individual x_i in a population containing λ members is assigned a probability of selection that is determined by its contribution to the total of the objective values of all population members:

$$p(x_i) = \frac{f(i)}{\sum_{j=1}^{\lambda} f(x_j)}.$$

Efforts have been made to improve upon this basic strategy by reducing the stochastic errors that it may involve; the method of 'Stochastic Sampling Without Replacement,' for example, demands that any population members previously selected be excluded from the denominator.

An alternative class of selection policies is based on eliminating the direct involvement of the actual objective value $f(x_i)$ itself from the computation of the selection probability $p(x_i)$. Instead, the population members are ranked according to their objective function, and it is the rank of a member x_i, say $\zeta_i \equiv \zeta(x_i)$, that directly determines its selection probability. The 'Linear Ranking' selection scheme is given below:

$$p(x_i) = \frac{1}{\lambda}\left(\eta_{\max} - (\eta_{\max} - \eta_{\min})\frac{\zeta_i - 1}{\lambda - 1}\right),$$

where $1 \leq \eta_{\max} \leq 2$ and $\eta_{\min} = 2 - \eta_{\max}$.

The proportional selection mechanism, basing its selection probabilities on actual objective function values, discriminates more harshly against unfit individuals, and favors more strongly those that are extremely good. The linear ranking scheme is more forgiving, since the relative differences in actual fitness value do not influence the selection probability, and therefore tends to preserve population diversity.

As presented, both of these selection schemes assign to each population member a probability of selection, all according to the same rules. However, extreme individuals may warrant some special treatment.

Usually, parent individuals are permitted to undergo reproduction only once, then being replaced in the new population by their offspring. This allows the possibility that super-fit individuals, having undergone crossover with other, less outstanding individuals, are replaced by considerably inferior candidate solutions. To avoid this eventuality, an 'elitist' (or k-elitist) scheme copies unchanged the best population member (or the k best members) to the new population unaltered. Selection continues as usual, except that the mating pool is appropriately reduced in size.

The least fit individuals of the population may also be singled out for separate treatment, by assigning to them a zero probability of selection, and creating an 'extinctive' selection mechanism. The roulette wheel selection policy so modified, producing the (μ, λ)-proportional selection scheme, can be expressed as

$$p(x_i) = \begin{cases} \dfrac{f(i)}{\sum_{j=1}^{\lambda} f(x_j)} & \text{if } 1 \leq \zeta_i \leq \mu, \\ 0 & \text{if } \mu < \zeta_i \leq \lambda \end{cases}$$

and similarly the generalization (μ, λ)-linear ranking, as

$$p(x_i) = \begin{cases} \dfrac{1}{\mu}\left(\eta_{\max} - (\eta_{\max} - \eta_{\min})\dfrac{\zeta_i - 1}{\mu - 1}\right) & \text{if } 1 \leq \zeta_i \leq \mu, \\ 0 & \text{if } \mu < \zeta_i \leq \lambda \end{cases}$$

where $\mu : 1 \leq \mu \leq \lambda$ is the number of individuals that are permitted a chance of reproduction. The $\lambda - \mu$ least fit individuals become extinct. Clearly, when $\mu = \lambda$, the extinctive scheme

degenerates into the corresponding preservative scheme. As μ decreases, the selective pressure on population members increases, thereby allowing greater control of the selection mechanism's demeanor.

4. TWO-POINT CROSSOVER

The virility of each population member having been so adjudged, pairs are randomly selected from the mating pool to undergo crossover. Modeled on the reproduction of biological organisms by the exchange of genetic information, the crossover operator facilitates the parturition of new candidate solutions.

Every member of a population represents a complete potential solution to the problem. Fragments of each illustrate partial solutions that are notions of the structure of the optimum; a whole population therefore contains a multitude of information about the problem in the form of these partial solutions, or 'schemata,' and through the selection mechanism, evaluations of each.

The crossover operator speculates on new solutions constructed by combining these partial solutions, according to their performance. The new population is an innovation comprised of solutions that are the juxtaposition of partial solutions that have previously proven successful. Superior partial solutions are well represented in the mating pool, and are therefore likely to survive in the new population, as a part of a new individual, whereas inferior schemata rapidly become extinct.

Numerous crossover operators have been proposed, for a whole range of problem domains [2,3,6,18,21–24]. The two-point crossover has been chosen here because it facilitates the preservation of integral values, unlike those proposed for the continuous case [3]. Furthermore, it promises better performance than the simple one-point crossover [2,24].

The two-point crossover operator effectuates the fabrication of new solutions from old by exchanging information between pairs of vectors. For each pair, two randomly-generated indices mark the boundaries of the corresponding segments to be interchanged. In this section, this operator is formally defined for candidate solutions to problem (P1), leading to a theoretical description of its behavior.

The feasible region P is a convex polyhedron in n-dimensional space,

$$P = \{x \in \mathbf{Z}^n : Ax \leq b \wedge l \leq x \leq u\}.$$

This polyhedron is covered by the smallest closed bounded set $\Lambda \subset \mathbf{Z}^n$ that forms an interval of n-dimensional space around P,

$$\Lambda(P) = \{x \in \mathbf{Z}^n : \bar{l} \leq x \leq \bar{u}\},$$

where

$$\bar{l} = [\bar{l}_i]_{i=1}^n \bullet \bar{l}_i = \min_x \{x_i : x = [x_j]_{j=1}^n \in \mathbf{Z}^n \wedge Ax \leq b \wedge l \leq x \leq u\},$$

and

$$\bar{u} = [\bar{u}_i]_{i=1}^n \bullet \bar{u}_i = \max_x \{x_i : x = [x_i]_{i=1}^n \in \mathbf{Z}^n \wedge Ax \leq b \wedge l \leq x \leq u\}.$$

That is, $\bar{l}$ and $\bar{u}$ are the vectors whose components are the smallest and largest, respectively, corresponding components of all feasible solutions x. Where the feasible region P is understood, this interval will be abbreviated as

$$\Lambda \equiv \Lambda(P).$$

Note also that the size of this interval is

$$|\Lambda| = \prod_{i=1}^{n} (\bar{u}_i - \bar{l}_i + 1).$$

The crossover operator takes as its arguments two vectors in n-dimensional space and the indices defining the crossover segment, and produces a new pair of vectors. No ordering is defined between the vectors comprising such a pair, and therefore all such pairs is a set

$$\Omega = \{\{x, y\} : x, y \in \mathbf{Z}^n\}.$$

The indices that describe the actual information exchange each indicate particular corresponding components of the vectors $\{x, y\}$, and for convenience the set of all indices of an n-dimensional vector is here denoted as

$$\chi = \{i : 1 \leq i \leq n \wedge i \in \mathbf{N}\}.$$

Two-point crossover is therefore a function

$$\Xi : \Omega \times \chi \times \chi \to \Omega$$

defined as

$$\Xi(\{x, y\}, h, k) = \begin{cases} \{x', y'\} & \text{if } h < k, \\ \{x, y\} & \text{otherwise,} \end{cases}$$

and where x, y, and the resultant vectors x' and y' have the form

$$\begin{aligned} x &= \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^\top, \quad y = \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix}^\top, \\ x' &= \begin{bmatrix} x_1 & \cdots & x_h & y_{h+1} & \cdots & y_k & x_{k+1} & \cdots & x_n \end{bmatrix}^\top, \\ y' &= \begin{bmatrix} y_1 & \cdots & y_h & x_{h+1} & \cdots & x_k & y_{k+1} & \cdots & y_n \end{bmatrix}^\top. \end{aligned}$$

The crossover will be called 'meaningful' if $h < k$; no information exchange occurs when $h \geq k$.

For convenience, the set of points that are produced by the crossover operator will be abbreviated as

$$\Xi_{h,k}(x, y) \equiv \Xi(\{x, y\}, h, k).$$

Any meaningful crossover for which $\Xi_{h,k}(x, y) = \{x, y\}$ will be termed 'trivial,' including the case when $x = y$. All other meaningful crossover operations are 'nontrivial.'

The set of points that can be produced by the crossover of two distinct vectors $x, y \in \Lambda$, including x and y themselves, will be denoted as

$$\begin{aligned} \Xi(x, y) &= \{z \in \mathbf{Z}^n : \exists h, k \in \chi, z \in \Xi_{h,k}(x, y)\} \\ &= \bigcup_{h,k \in \chi} \Xi_{h,k}(x, y). \end{aligned}$$

The number of points that might be generated from x and y, in this way, can be described. In particular, the extreme and average sizes of $\Xi(x, y)$ for all possible distinct pairs of vectors x and y in the interval Λ is given below.

LEMMA 4.1. *The number of points that can be generated by the crossover of two distinct points $x, y \in \Lambda$ satisfies*

$$2 \leq |\Xi(x, y)| \leq n^2 - n + 2.$$

Moreover, the expected number of such points is

$$\mathbf{E}(|\Xi(x, y)|) = 2 + \rho(\rho - 1), \qquad \text{where } \rho = \sum_{i=1}^{n} \frac{\bar{u}_i - \bar{l}_i}{\bar{u}_i - \bar{l}_i - 1}.$$

PROOF. Clearly, a crossover operation produces two new vectors when the values of the crossover segments $[x_{h+1} \cdots x_k]$ and $[y_{h+1} \cdots y_k]$ differ, for $h, k : 1 \leq h < k \leq n$. Conversely, when $h \geq k$ or the crossover segments are identical, $\Xi_{h,k}(x, y) = \{x, y\}$. Therefore, summing over all differing crossover segments, and adding x and y themselves,

$$\begin{aligned} |\Xi(x, y)| &= 2\left(1 + \sum_{h=1}^{d-1} \sum_{k=h+1}^{d} 1\right) \\ &= 2 + d^2 - d, \end{aligned}$$

where d is the number of corresponding components of x and y that differ. In particular, when $d = n$, all corresponding components of x and y differ, and $|\Xi(x, y)| = 2 + n^2 - n$. Clearly, since $x, y \in \Xi(x, y)$, $|\Xi(x, y)| \geq 2$. Note also that x and y are distinct, so that $d \geq 1$.

Consider now the number of components that can be expected to be equal. Let

$$Q_i = \begin{cases} 1 & \text{if } x_i = y_i, \\ 0 & \text{otherwise.} \end{cases}$$

Then, summing over all possible values $v \in \{\bar{l}_i, \ldots, \bar{u}_i\}$ that component i may take, the probability that $x_i = y_i$ is

$$\begin{aligned} \Pr(Q_i = 1) &= \sum_{v=\bar{l}_i}^{\bar{u}_i} \Pr(x_i = v) \cdot \Pr(y_i = v) \\ &= \sum_{v=\bar{l}_i}^{\bar{u}_i} \frac{1}{(\bar{u}_i - \bar{l}_i + 1)^2} \\ &= \frac{1}{\bar{u}_i - \bar{l}_i + 1}. \end{aligned}$$

Then the expected value of Q_i is

$$\mathbf{E}(Q_i) = \sum_i i \cdot \Pr(Q_i = i) = \frac{1}{\bar{u}_i - \bar{l}_i + 1}.$$

Since events $Q_i : i = 1, \ldots, n$ are disjoint, by the linearity of expectation,

$$\mathbf{E}\left(\sum_{i=1}^{n} Q_i\right) = \sum_{i=1}^{n} \mathbf{E}(Q_i) = \sum_{i=1}^{n} \frac{1}{\bar{u}_i - \bar{l}_i + 1}.$$

The expected number of differing components in x and y is therefore

$$\begin{aligned} \rho &= n - \mathbf{E}\left(\sum_{i=1}^{n} Q_i\right) \\ &= \sum_{i=1}^{n} \frac{\bar{u}_i - \bar{l}_i}{\bar{u}_i - \bar{l}_i + 1}. \end{aligned}$$

Then $\mathbf{E}(|\Xi(x, y)|) = \rho^2 - \rho + 2$, as claimed. ∎

The set of all vectors generated from a particular set $Z \subseteq \Lambda$ by crossover, with the crossover segment designated by (h, k), where $h, k \in \chi$, is the set

$$\begin{aligned} \Xi_{h,k}(Z) &= \{z \in \Xi_{h,k}(x, y) : x, y \in Z\} \\ &= \bigcup_{x,y \in Z} \Xi_{h,k}(x, y). \end{aligned}$$

The set of points produced from Z by crossover, by any exchanges of information, is then defined as

$$\begin{aligned}\Xi(Z) &= \{z \in \mathbf{Z}^n : \exists h, k \in \chi, z \in \Xi_{h,k}(Z)\} \\ &= \bigcup_{h,k \in \chi} \Xi_{h,k}(Z) \\ &= \bigcup_{x,y \in Z} \Xi(x,y).\end{aligned}$$

In particular, the set of all vectors that can be manufactured by the crossover of feasible vectors is $\Xi(P)$.

The crossover of feasible solutions to problem (P1) cannot result in the generation of points that lie outside the interval Λ that surrounds the feasible set. Proof of this is trivial. Note also that the converse is not true.

LEMMA 4.2. *Crossover of feasible points produces points in* Λ. *That is,*

$$\Xi(P) \subseteq \Lambda.$$

Furthermore, the crossover of any points in the enclosing interval must manufacture points that are themselves contained within this interval. Again, proof is trivial.

LEMMA 4.3. Λ *is closed under* Ξ. *That is,*

$$\Xi(\Lambda) = \Lambda.$$

Therefore, in considering the behavior of the crossover operator, as applied to problem (P1), attention may be restricted entirely to the interval that encloses the feasible set. No points outside Λ ever need be regarded. In addition, only crossovers that have the potential to offer new candidate solutions need be considered, and therefore assiduity is limited only to meaningful crossovers. That is, all crossovers of relevance are of the form

$$\Xi_{h,k}(x,y) \bullet (\{x,y\}, h, k) \in \Phi(\Lambda),$$

where the set $\Phi(S)$ of meaningful crossover operations between points in $S \subseteq \Lambda$ is defined as

$$\Phi(S) = \{(\{x,y\}, h, k) : x, y \in S \wedge x \neq y \wedge h, k \in \chi \wedge h < k\}.$$

The number of crossovers between distinct points in the set S is easily derived.

LEMMA 4.4. *The number of meaningful crossovers between points in* $S \subseteq \Lambda$ *is*

$$|\Phi(S)| = \frac{1}{4}(n^2 - n)|S|(|S| - 1).$$

PROOF. Each distinct crossover requires two points from S. Considering all meaningful crossover segments, $h, k : h, k \in \chi \wedge h < k$,

$$\begin{aligned}|\Phi(S)| &= \sum_{h=1}^{n-1} \sum_{k=h+1}^{n} \binom{|S|}{2} \\ &= \frac{n^2 - n}{2} \cdot \frac{|S|(|S| - 1)}{2},\end{aligned}$$

as claimed. ∎

The number of meaningful crossover operations $\mathcal{N}(S)$ that actually produce points of a given set $S \subseteq \Lambda$, from points in Λ, can now be defined. Note that this includes any crossover that produces at least one element of S; the second vector produced may or may not be in S. That is, let

$$\mathcal{N}(S) = |\{(\{x, y\}, h, k) \in \Phi(\Lambda) : \Xi_{h,k}(x, y) \cap S \neq \emptyset\}|.$$

The value of $\mathcal{N}(S)$ can be determined, in terms of the cardinality of S, the vectors $\bar{l}$ and $\bar{u}$ that define the interval Λ, and the dimensionality n of the problem.

LEMMA 4.5. *The number of meaningful crossovers producing points in a set*

$$S \subseteq \Lambda$$

from distinct points in the interval $\Lambda = \{x : \bar{l} \leq x \leq \bar{u}\} \subset \mathbf{Z}^n$ *is*

$$\mathcal{N}(S) = \left(\frac{n^2 - n}{2}\right) \cdot |S| \cdot \left(\prod_{i=1}^{n} (\bar{u}_i - \bar{l}_i + 1) - \frac{|S| + 1}{2}\right).$$

PROOF. Consider first the number of meaningful crossovers that produce a given point $z \in \Lambda$, but do not produce points in some set $X \subseteq \Lambda$, from distinct points in Λ. That is, consider

$$\mathcal{N}'(z, X) = |\{(\{x, y\}, h, k) \in \Phi(\Lambda) : \exists w \notin X \bullet \{z, w\} = \Xi_{h,k}(x, y)\}|.$$

Then let $\Xi_{h,k}(x, y) = \{z, w\}$, where, say,

$$\begin{aligned} z &= [z_1 \cdots z_n]^\top, \\ w &= [w_1 \cdots w_n]^\top \in (\Lambda - X) - \{z\}. \end{aligned}$$

Then $x, y \in \Lambda$ are of the form

$$\begin{aligned} x &= [w_1 \cdots w_h \; z_{h+1} \cdots z_k \; w_{k+1} \cdots w_n]^\top, \\ y &= [z_1 \cdots z_h \; w_{h+1} \cdots w_k \; z_{k+1} \cdots z_n]^\top, \end{aligned}$$

for some $h, k : 1 \leq h < k \leq n$. That is, there is a meaningful crossover producing z for each $w \in (\Lambda - X) - \{z\}$, and each $h, k : 1 \leq h < k \leq n$. Therefore,

$$\begin{aligned} \mathcal{N}'(z, X) &= \sum_{h=1}^{n-1} \sum_{k=h+1}^{n} (|\Lambda| - |X| - 1) \\ &= \sum_{h=1}^{n-1} \sum_{k=h+1}^{n} \left(\prod_{i=1}^{n} (\bar{u}_i - \bar{l}_i + 1) - |X| - 1\right) \\ &= \left(\frac{n^2 - n}{2}\right) \left(\prod_{i=1}^{n} (\bar{u}_i - \bar{l}_i + 1) - |X| - 1\right). \end{aligned}$$

Adding these for all points $z \in S$, with X being the set of points already counted,

$$\begin{aligned} \mathcal{N}(S) &= \sum_{j=1}^{|S|} \left(\left(\frac{n^2 - n}{2}\right) \left(\prod_{i=1}^{n} (\bar{u}_i - \bar{l}_i + 1) - (j - 1) - 1\right)\right) \\ &= \left(\frac{n^2 - n}{2}\right) \cdot \left(\sum_{j=1}^{|S|} \prod_{i=1}^{n} (\bar{u}_i - \bar{l}_i + 1) - \sum_{j=1}^{|S|} j\right) \\ &= \left(\frac{n^2 - n}{2}\right) \cdot |S| \cdot \left(\prod_{i=1}^{n} (\bar{u}_i - \bar{l}_i + 1) - \frac{|S| + 1}{2}\right), \end{aligned}$$

as required. ∎

In contrast to the number $\mathcal{N}(S)$ of crossovers that produce at least one element of the set $S \subseteq \Lambda$, the number of crossover operators $\mathcal{M}(S)$ that strictly produce only vectors in S may also be considered. That is, let

$$\mathcal{M}(S) = |\{(\{x,y\},h,k) \in \Phi(\Lambda) : \Xi_{h,k}(x,y) \subseteq S\}|\,.$$

(Note also that $\mathcal{M}(S) \leq \mathcal{N}(S)$.) The actual value of $\mathcal{M}(S)$ can be surmised similarly to that of $\mathcal{N}(S)$.

LEMMA 4.6. *The number of meaningful crossovers that produce only elements of a set*

$$S \subseteq \Lambda$$

from distinct points in the interval $\Lambda = \{x : \bar{l} \leq x \leq \bar{u}\} \subset \mathbf{Z}^n$ *is*

$$\mathcal{M}(S) = \left(\frac{n^2-n}{4}\right) \cdot |S| \cdot (|S|-1)\,.$$

PROOF. Consider the number $\mathcal{M}'(z,X) \equiv \mathcal{M}(z,S,X)$ of meaningful crossovers that, from distinct points in Λ, produce a given point $z \in \Lambda$, and some other point w also in S but not in a set $X \subseteq \Lambda$.

$$\mathcal{M}'(z,X) = |\{(\{x,y\},h,k) \in \Phi(\Lambda) : \exists w \in S - X \bullet \{z,w\} = \Xi_{h,k}(x,y)\}|\,.$$

Then $\Xi_{h,k}(x,y) = \{z,w\}$, where $z \in S$, and $w \in (S-X)-\{z\}$, so that there exists a meaningful crossover producing z, and some other point in S but not in X, for each $w \in (S-X)-\{z\}$, and for each $h,k : 1 \leq h < k \leq n$. Therefore,

$$\begin{aligned}\mathcal{M}'(z,X) &= \sum_{h=1}^{n-1} \sum_{k=h+1}^{n} (|S| - |X| - 1) \\ &= \left(\frac{n^2-n}{2}\right)(|S| - |X| - 1)\,.\end{aligned}$$

Adding these for all points $z \in S$, and with X the set of points already accounted for,

$$\begin{aligned}\mathcal{M}(S) &= \sum_{j=1}^{|S|} \left(\frac{n^2-n}{2}\right)(|S| - (j-1) - 1) \\ &= \left(\frac{n^2-n}{2}\right)\left(\sum_{j=1}^{|S|} |S| - \sum_{j=1}^{|S|} j\right) \\ &= \left(\frac{n^2-n}{2}\right)|S|\left(\frac{|S|-1}{2}\right),\end{aligned}$$

as required. ∎

In particular, these lemmas may be exercised in understanding the behavior of the two-point crossover operator on feasible solutions. Clearly, the potential exists for the generation of infeasible candidate solutions, whenever $|P| < |\Lambda|$, where $|P|$ is the total number of feasible points. The design of an appropriate mechanism for managing the crossover depends upon exactly how great this propensity is.

The immediate strategy is to confine attention entirely to those crossovers that produce only feasible points, from feasible points. Crossover operations of this type,

$$\{(\{x,y\},h,k) \in \Phi(\Lambda) : x,y \in P \wedge \Xi_{h,k}(x,y) \subseteq P\}\,,$$

will be termed 'feasible,' or 'entirely feasible' to emphasize that only feasible solutions are produced. Note the distinction between this set, that allows only feasible solutions as arguments

to a crossover, and the set of points described in Lemma 4.6, that counts any crossover that manufactures two feasible solutions, from any points in Λ.

The number of crossovers that produce only feasible points, from any points in Λ, is $\mathcal{M}(P)$. However, of the total $|\Phi(\Lambda)|$ of crossover operations possible, $|\Phi(P)|$ of them are between feasible solutions. The number of feasible crossover operations is therefore

$$\begin{aligned}\bar{\mathcal{M}}(P) &= \mathcal{M}(P) \cdot \frac{|\Phi(P)|}{|\Phi(\Lambda)|} \\ &= \left(\frac{n^2-n}{4}\right) \cdot \frac{|P|^2(|P|-1)^2}{|\Lambda|(|\Lambda|-1)}\end{aligned} \tag{1}$$

by Lemmas 4.4 and 4.6. Put another way, of the $|\Phi(P)|$ crossovers between feasible points, $\bar{\mathcal{M}}(P)$ produce only feasible solutions. The probability that a crossover randomly chosen from $\Phi(P)$ is feasible is therefore

$$\begin{aligned}\Pr\left(\Xi_{h,k}(x,y) \subseteq P\right) &= \frac{\bar{\mathcal{M}}(P)}{|\Phi(P)|} \\ &= \frac{\mathcal{M}(P)}{|\Phi(\Lambda)|} \\ &= \frac{|P|(|P|-1)}{|\Lambda|(|\Lambda|-1)}.\end{aligned} \tag{2}$$

Given any two points, there are a total of $(1/2)(n^2-n)$ meaningful crossover operations on these points, one for each conceivable crossover segment $h,k : 1 \leq h < k \leq n$. Given two randomly-chosen feasible points, the expected number of feasible crossovers between these points is therefore

$$\begin{aligned}\mathbf{E}\left(|\{(h,k) : h < k \wedge \Xi_{h,k}(x,y) \subseteq P\}|\right) &= \Pr\left(\Xi_{h,k}(x,y) \subseteq P\right) \cdot \frac{1}{2}\left(n^2-n\right) \\ &= \frac{1}{2}\left(n^2-n\right) \cdot \frac{|P|(|P|-1)}{|\Lambda|(|\Lambda|-1)}.\end{aligned} \tag{3}$$

Given a pair of solutions chosen from the mating pool, the policy initially proposed is the restriction of the choice of crossover to those that are entirely feasible.

As the size of the feasible region P decreases with respect to that of the enclosing interval Λ, so too does the probability (2) of a random crossover between points in P being feasible. The result is a decrease in the expected number of feasible crossovers (3) between two points $x, y \in P$, and therefore in the number of points that can be generated from them. Reconsidering Lemma 4.1, the expected number of candidate solutions that might be produced by crossing x and y is

$$2 + \rho(\rho-1),$$

but of these, on average, only

$$2 + \rho(\rho-1) \cdot \Pr(\Xi_{h,k}(x,y) \subseteq P) = 2 + \rho(\rho-1) \cdot \frac{|P|(|P|-1)}{|\Lambda|(|\Lambda|-1)}$$

are produced by entirely feasible crossovers. As the size of P so increases, the possibility that, for a specific pair of points $x, y \in P$, no nontrivial feasible crossover exists, becomes increasingly likely. When this occurs, no productive information exchange can be realized, resulting in a degradation in performance.

This may be ameliorated in the realization that not all crossovers that produce feasible solutions are themselves entirely feasible. The class of feasible crossovers on P is wholly contained within the broader set of crossovers that produce at least one feasible point; by exploiting this, at least some potentially new solution might be developed.

The set of 'half-feasible' crossovers that produce only one feasible point is

$$\{(\{x,y\},h,k) \in \Phi(\Lambda) : x,y \in P \wedge |\Xi_{h,k}(x,y) \cap P| = 1\},$$

and its size can be derived similarly to that of the set of feasible crossovers (1), by using Lemmas 4.4–4.6:

$$\begin{aligned} \bar{\mathcal{H}}(P) &= (\mathcal{N}(P) - \mathcal{M}(P)) \cdot \frac{|\Phi(P)|}{|\Phi(\Lambda)|} \\ &= \frac{n^2-n}{2} \cdot \frac{(|P|-1)(|\Lambda|-|P|)|P|^2}{|\Lambda|(|\Lambda|-1)}. \end{aligned} \tag{4}$$

Moreover, the total number of crossovers between feasible points that produce either one or two feasible points is

$$\begin{aligned} \bar{\mathcal{N}}(P) &= \bar{\mathcal{M}}(P) + \bar{\mathcal{H}}(P) \\ &= \frac{n^2-n}{4} \cdot \frac{|P|^2(|P|-1)(2|\Lambda|-|P|-1)}{|\Lambda|(|\Lambda|-1)}. \end{aligned} \tag{5}$$

By comparing equations (1) and (4), the number of half-feasible crossovers is found to exceed the number of feasible crossovers when the feasible region is sufficiently small, when compared to the size of the bounding interval Λ. That is,

$$\bar{\mathcal{H}}(P) > \bar{\mathcal{M}}(P), \qquad \text{whenever } |P| < \frac{2(|\Lambda|+1)}{3}, \tag{6}$$

for any feasible region $P : |P| > 1$. Under these conditions, it is readily seen that the probability

$$\begin{aligned} \Pr\left(|\Xi_{h,k}(x,y) \cap P| = 1\right) &= \frac{\bar{\mathcal{H}}(P)}{|\Phi(P)|} \\ &= \frac{2(|\Lambda|-|P|)|P|}{|\Lambda|(|\Lambda|-1)} \end{aligned} \tag{7}$$

of a random crossover between feasible points being half-feasible exceeds the probability (2) of it being entirely feasible. The expected number of half-feasible crossovers between two points $x, y \in P$

$$\mathbf{E}\left(|\{(h,k) : h < k \wedge |\Xi_{h,k}(x,y) \cap P| = 1\}|\right) = (n^2-n) \cdot \frac{(|\Lambda|-|P|+1)|P|}{|\Lambda|(|\Lambda|-1)} \tag{8}$$

is larger than the expected number of entirely feasible crossovers, and therefore utilizing the half-feasible crossover may provide considerable advantages, especially in the resolution of highly constrained examples.

EXAMPLE 2. The behavior of the two-point crossover with respect to changing feasible region size is here delineated by the cases for which the bounding interval Λ contains 100 points. The first graph (see Figure 1) shows the probabilities that a random crossover between feasible solutions

(a) is entirely feasible, $\Pr\left(\Xi_{h,k}(x,y) \subseteq P\right)$,
(b) is half-feasible, $\Pr\left(|\Xi_{h,k}(x,y) \cap P| = 1\right)$, and
(c) generates at least one feasible point, $\Pr\left(\Xi_{h,k}(x,y) \cap P \neq \emptyset\right)$.

The second graph (Figure 2) shows the proportion of crossovers producing at least one feasible point (5) that are

(a) entirely feasible, $\bar{\mathcal{M}}(P)/\bar{\mathcal{N}}(P)$, and
(b) half-feasible, $\bar{\mathcal{H}}(P)/\bar{\mathcal{N}}(P)$.

Notice how the half-feasible crossovers dominate the entirely feasible variety for smaller feasible regions, when $|P| < 2(|\Lambda|+1)/3$, that is, whenever $|P| \leq 67$. ∎

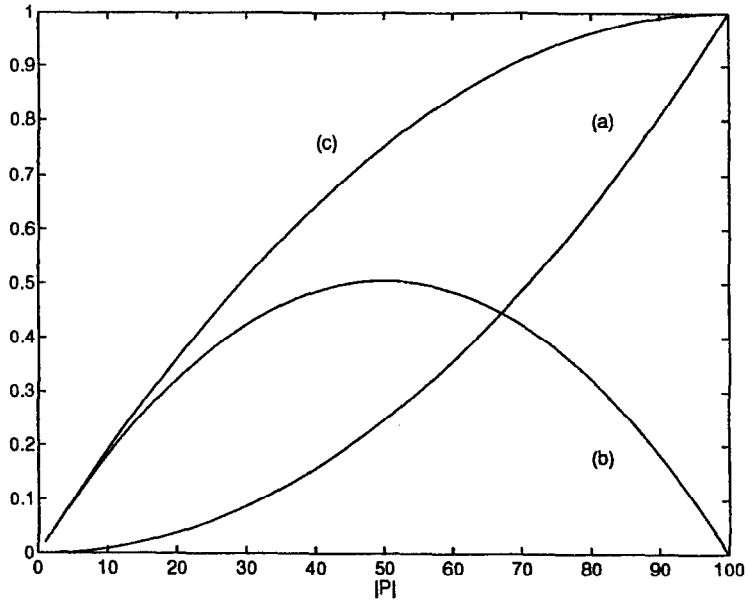

Figure 1. Behaviors of a random crossover.

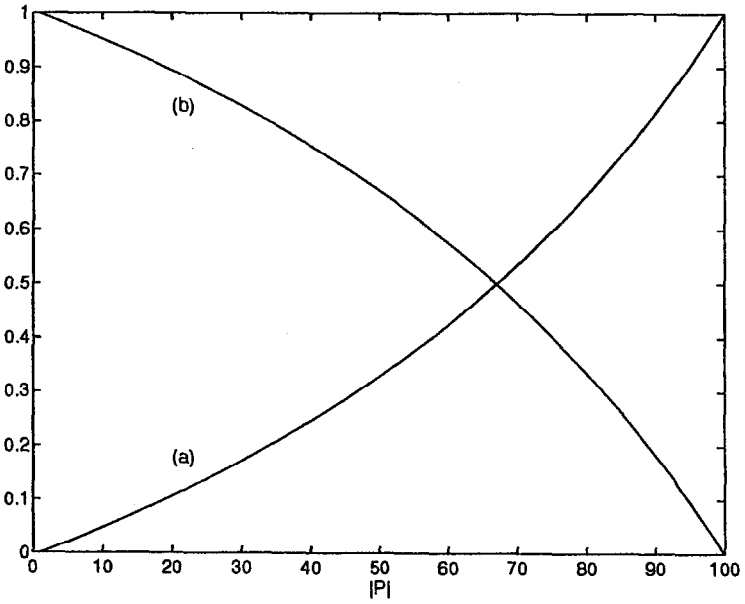

Figure 2. Proportions of entirely and half-feasible crossovers.

5. MAINTAINING FEASIBILITY

With this understanding of the behavior of two-point crossover on solutions lying in some feasible region, effort can now be directed at modifying this operator to account fully for the constraint system. The results of the last section therefore underly the designs of the algorithms presented below.

The traditional genetic algorithms, that regard only some kind of bound constraints, choose the crossover segment according to a uniform distribution from all those possible. In the limitation that only feasible solutions may be manufactured, the election of a crossover segment should be correspondingly made at random, from the set of all that are so viable.

In order to afford the restriction of the choice of the crossover segment to those that produce only feasible solutions, a sensible and efficient search for such a crossover is required. The

generation of all feasible crossovers, for a given pair of parent solutions, is superfluously and prohibitively expensive; this would demand attention for each of the $(1/2)\,n(n-1)$ possible pairs distinct pairs of crossover points. By first deciding upon one of the segment boundaries, by a random selection, the number of possibilities remaining for the other is $n-1$; this boundary may be interpreted either as the lower delimitatation h, or as the upper k.

The actual search for feasible crossovers between a pair of parent solutions that have this chosen point as one of the edges of the crossover segment might be effectuated by producing, in turn, each of the would-be resultant pairs, and appraising their satisfaction of the constraint system. This involves the multiplication of the constraint matrix A consecutively by each of the aspiring child solutions, and comparison of these with the constraint bounds b. The feasibility check for each conceivable crossover section, when performed in this manner, demands $O(mn+m)$ operations, giving a total of $O(mn^2+mn)$ for the entire search.

By disregarding the relationship that exists between crossovers that have a particular boundary in common, unnecessary and excessive work is invested in the foray for a feasible crossover. More careful excogitation reveals that the feasibility of each potentiality can be determined in $O(m+n)$ operations, giving a total of $O(mn+n^2)$ for the whole search, if done in an agreeable order.

For clarity in the discussion that follows, the residual of a vector in Λ is a function

$$r : \Lambda \to \mathbf{R}^m,$$

defined as

$$r(x) = b - Ax,$$

and represents the amount by which the vector x violates, or satisfies, the constraints. The proposal is made that the residual of each population member be stored with it, to enable a rapid search for a feasible crossover. Having been discovered, the residuals of the new vectors produced by such a crossover can be quickly found from those of the parents. The search technique is justified by Theorem 5.1, below.

THEOREM 5.1. *The crossover* $\{x', y'\} = \Xi_{h,k}(x, y)$ *is feasible iff*

$$-r(y) \le \delta^{(h,k)} \le r(x)$$

(in terms of the definition of crossover, x' *is feasible iff* $\delta^{(h,k)} \le r(x)$*, and likewise* y' *iff* $-r(y) \le \delta^{(h,k)}$*), where* $\delta^{(h,k)} \equiv \delta^{(h,k)}(x, y)$ *satisfies*

$$\begin{aligned}
\delta^{(h-1,k)} &= \delta^{(h,k)} + (y_h - x_h) \cdot [a_{i,h}]_{i=1}^{m}, \\
\delta^{(h,k+1)} &= \delta^{(h,k)} + (y_{k+1} - x_{k+1}) \cdot [a_{i,k+1}]_{i=1}^{m}, \\
\delta^{(h,h)} &= 0,
\end{aligned}$$

for any h, k *satisfying* $1 \le h \le k \le n$*. Moreover,*

$$r(x') = r(x) - \delta^{(h,k)} \quad \text{and} \quad r(y') = r(y) + \delta^{(h,k)}.$$

PROOF. Let $h, k : 1 \le h \le k$, $x, y \in P$, and suppose that $\{x', y'\} = \Xi_{h,k}(x, y)$. Then let the vectors x_L, x_M, x_H, y_L, y_M, y_H be defined as

$$\begin{aligned}
x_L &= [x_1 \dots x_h]^\top, \quad x_M = [x_{h+1} \dots x_k]^\top, \quad x_H = [x_{k+1} \dots x_n]^\top, \\
y_L &= [y_1 \dots y_h]^\top, \quad y_M = [y_{h+1} \dots y_k]^\top, \quad y_H = [y_{k+1} \dots y_n]^\top,
\end{aligned}$$

so that

$$x = \begin{bmatrix} x_L \\ x_M \\ x_H \end{bmatrix}, \quad y = \begin{bmatrix} y_L \\ y_M \\ y_H \end{bmatrix}, \quad x' = \begin{bmatrix} x_L \\ y_M \\ x_H \end{bmatrix}, \quad y' = \begin{bmatrix} y_L \\ x_M \\ y_H \end{bmatrix}.$$

Similarly partition the constraint matrix $A = [A_L A_M A_H]$ according to

$$\begin{aligned} A_L &= \left[[a_{i,1}]_{i=1}^m \cdots [a_{i,h}]_{i=1}^m \right], \\ A_M &= \left[[a_{i,h+1}]_{i=1}^m \cdots [a_{i,k}]_{i=1}^m \right], \\ A_H &= \left[[a_{i,k+1}]_{i=1}^m \cdots [a_{i,n}]_{i=1}^m \right]. \end{aligned}$$

Then

$$Ax' = A_L x_L + A_M y_M + A_H x_H, \quad \text{and} \quad Ay' = A_L y_L + A_M x_M + A_H y_H.$$

But $Ax = A_L x_L + A_M x_M + A_H x_H$, so that

$$A_H x_H = Ax - A_L x_L - A_M x_M,$$

and also $Ay = A_L y_L + A_M y_M + A_H y_H$, giving

$$A_H y_H = Ay - A_L y_L - A_M y_M.$$

Consequentially,

$$Ax' = Ax + A_M y_M - A_M x_M, \quad \text{and} \quad Ay' = Ay + A_M x_M - A_M y_M.$$

Letting $\delta^{(h,k)} \equiv \delta^{(h,k)}(x,y) = A_M \cdot (y_M - x_M)$,

$$Ax' = Ax + \delta^{(h,k)} \quad \text{and} \quad Ay' = Ay - \delta^{(h,k)}.$$

Then, note that

$$\begin{aligned} r(x') &= b - Ax - \delta^{(h,k)} \quad \text{and} \quad r(y') = b - Ay + \delta^{(h,k)} \\ &= r(x) - \delta^{(h,k)} \qquad\qquad = r(y) + \delta^{(h,k)}, \end{aligned}$$

as required.

Now imposing feasibility on the candidate solution x',

$$\begin{aligned} b - Ax' = b - \left(Ax + \delta^{(h,k)} \right) &\geq 0, \\ \delta^{(h,k)} &\leq b - Ax, \\ \delta^{(h,k)} &\leq r(x), \end{aligned}$$

and likewise for y',

$$\begin{aligned} b - Ay' = b - \left(Ay - \delta^{(h,k)} \right) &\geq 0, \\ \delta^{(h,k)} &\geq -(b - Ay), \\ \delta^{(h,k)} &\geq -r(y). \end{aligned}$$

That is, x' and y' are feasible iff $-r(y) \leq \delta^{(h,k)} \leq r(x)$, as required.

Furthermore, observe that $\delta^{(h,h)} = 0$, for any $h : 1 \leq h \leq n$, by definition. Now expressing $\delta^{(h-1,k)}$ in terms of $\delta^{(h,k)}$, for h, k such that $1 \leq h \leq k \leq n$,

$$\begin{aligned} \delta^{(h-1,k)} &= \left[[a_{i,h}]_{i=1}^m \cdots [a_{i,k}]_{i=1}^m \right] \cdot \left([y_j]_{j=h}^k - [x_j]_{j=h}^k \right) \\ &= [a_{i,h}]_{i=1}^m \cdot (y_h - x_h) \\ &\qquad + \left[[a_{i,h+1}]_{i=1}^m \cdots [a_{i,k}]_{i=1}^m \right] \cdot \left([y_j]_{j=h+1}^k - [x_j]_{j=h+1}^k \right) \\ &= \delta^{(h,k)} + [a_{i,h}]_{i=1}^m \cdot (y_h - x_h). \end{aligned}$$

Expressing $\delta^{(h,k+1)}$ also in terms of $\delta^{(h,k)}$,

$$\begin{aligned}\delta^{(h,k+1)} &= \left[[a_{i,h+1}]_{i=1}^{m} \cdots [a_{i,k+1}]_{i=1}^{m}\right] \cdot \left([y_j]_{j=h}^{k+1} - [x_j]_{j=h}^{k+1}\right) \\ &= \left[[a_{i,h+1}]_{i=1}^{m} \cdots [a_{i,k}]_{i=1}^{m}\right] \cdot \left([y_j]_{j=h+1}^{k} - [x_j]_{j=h+1}^{k}\right) \\ &\quad + [a_{i,k+1}]_{i=1}^{m} \cdot (y_{k+1} - x_{k+1}) \\ &= \delta^{(h,k)} + [a_{i,k+1}]_{i=1}^{m} \cdot (y_{k+1} - x_{k+1}),\end{aligned}$$

as required. ∎

Given a pair of feasible vectors from the mating pool, the quest for a feasible crossover begins with the random choice of a boundary for the crossover segment. This might be interpreted either as the lower or upper boundary, so the search proceeds in two stages. All feasible crossovers that have this chosen point as the lower crossover point h might be generated, according to Theorem 5.1, by initializing

$$\delta^{(h,h)} = 0,$$

and successively applying the recurrence relation

$$\delta^{(h,k+1)} = \delta^{(h,k)} + (y_{k+1} - x_{k+1}) \cdot [a_{i,k+1}]_{i=1}^{m},$$

with each such crossover segment (h, k) producing only feasible candidate solutions iff

$$-r(y) \leq \delta^{(h,k)} \leq r(x).$$

The generation of feasible crossovers with the chosen point treated as the upper boundary of the crossover segment may be similarly achieved, using instead the recurrence relation

$$\delta^{(h-1,k)} = \delta^{(h,k)} + (y_h - x_h) \cdot [a_{i,h}]_{i=1}^{m}.$$

After performing both searches, one of the feasible crossovers discovered (if there are any) might be chosen at random, and applied.

Eliminating the need to store the segment boundaries h and k, and the values of $\delta^{(h,k)}$ for each feasible crossover found, and allowing early termination, clearly improves both the space and time efficiency. This can be achieved by accepting or rejecting each feasible crossover as they are illuminated.

Presuming a uniform distribution in the choice of the crossover from all those found, the probability that any one is accepted is $1/c$, where c is the total number of feasible crossovers revealed by the search. However, given the probability that any crossover between feasible points is feasible (2), and since there are $n-1$ potential crossovers between these two points that have i as one of the segment boundaries, the expected number of feasible crossovers that are discovered is

$$(n-1) \cdot \frac{|P|(|P|-1)}{|\Lambda|(|\Lambda|-1)}.$$

Therefore, on average, the probability that a feasible crossover is chosen is

$$p_F = \frac{|\Lambda|(|\Lambda|-1)}{(n-1)|P|(|P|-1)}, \tag{9}$$

provided that

$$|P|(|P|-1) \geq \frac{|\Lambda|(|\Lambda|-1)}{n-1}. \tag{10}$$

When this condition (10) is violated, the number expected number of feasible crossovers found is less than one.

When a feasible crossover is revealed in either of the two stages of the search, it is accepted or rejected by a toss of a biased coin; the probability that it is accepted is p_F. To safeguard against termination without accepting any crossover, when indeed at least one entirely feasible crossover was discovered, temporary variables are used to store information concerning the last feasible crossover found. An outline of the search for a crossover segment with the lower boundary initially determined is given in Figure 3; the other case is analogous.

```
[δ, k, found] = SearchUp1 (x, y, r(x), r(y), h, p_F) {
   k = h;    δ = 0;
   found = 0;    lastk = 0;
   while (k < n and not found){
      δ = δ + [a_{i,k+1}]_{i=1}^{m} · (y_{k+1} − x_{k+1});    k = k + 1;
      if (−r(y) ≤ δ ≤ r(x)) {
         if (rand() < p_F)    found = 1;
         else { lastδ = δ;    lastk = k; };
         t_f = t_f + 1; };
      t_c = t_c + 1; };
   if (not found and lastk > 0) {
      δ = lastδ;
      k = lastk;
      found = 1; }; }
```

Figure 3. Search for an entirely feasible crossover.

In most practical situations, the parameter p_F cannot be known prior to the application of the genetic algorithm, as it is defined in terms of the cardinality of the feasible set. The global variables t_f and t_c are counters, maintained to track the total number of feasible crossovers found, and the total number of possible crossovers explored, respectively. Then

$$\frac{t_f}{t_c}$$

is the proportion of crossovers investigated that have been found to be feasible. This offers an estimation of the probability (2) that a random crossover between feasible points is entirely feasible, and so can be also applied to approximate the probability p_F that a feasible crossover will be chosen (9), as

$$p_F \approx \frac{t_c}{(n-1)t_f}.$$

The parameter p_F in the algorithm is actually this estimation, and is periodically updated, typically when the new population is completed. Note that as the genetic algorithm progresses, the greater the number of crossovers it samples, so that this approximation converges to the the exact probability.

The overall search algorithm calls either the function 'SearchUp1' given above, or its sister 'SearchDown1,' based on a flip of another biased coin, to determine which is allowed the first opportunity of discovering a feasible crossover. The prejudice towards one or the other is founded on the difference in the proportion of the total $1/p_F$ expected number of feasible crossovers that each is likely to encounter. That is, if i is the selected boundary point, then, on average, the proportions of the feasible crossovers recognized by 'SearchDown1' and 'SearchUp1' are

$$\frac{i-1}{n-1} \quad \text{and} \quad \frac{n-i}{n-1},$$

respectively. The foray for a feasible crossover thereby takes the form described in Figure 4.

```
[h, k, δ, found] = FeasibleCross (x, y, r(x), r(y), p_F) {
    i = randint(1, n);
    if (rand() < (i − 1)/(n − 1)) {
        k = i;
        [δ, h, found] = SearchDown1 (x, y, r(x), r(y), k, p_F);
        if (not found) {
            h = i;
            [δ, k, found] = SearchUp1 (x, y, r(x), r(y), h, p_F); }; }
    else {
        h = i
        [δ, k, found] = SearchUp1 (x, y, r(x), r(y), h, p_F); };
        if (not found) {
            k = i;
            [δ, h, found] = SearchDown1 (x, y, r(x), r(y), k, p_F); }; }; };
```

Figure 4. Entirely feasible crossover.

The search for an appropriate crossover segment so completed, and given that one has been found, the exchange of information can transpire with the confidence that only feasible candidate solutions will be generated. Finally, the residuals of the new points x' and y' are computed from those of their parents, as

$$r(x') = r(x) - \delta^{(h,k)} \quad \text{and} \quad r(y') = r(y) + \delta^{(h,k)}.$$

6. UTILIZING THE HALF-FEASIBLE CROSSOVERS

Given the increasing dominance (8) of crossovers that manufacture a single feasible point over those that produce two (3), as the size of the feasible set diminishes within its bounding interval (6), exploitation of these half-feasible crossovers must be considered. Most noticeably, when the expected number of entirely feasible crossovers approaches the point at which the condition (10) is violated, the rate of information exchange between population members rapidly depreciates. This undertaking demands the development of the search algorithm to incorporate the half-feasible crossovers in some sensible manner, and two possible schemes are proposed here.

Given the number $n - 1$ of crossover segments that are potentially explored by the search, and the probability (7) that a random crossover between two feasible points is half-feasible, the probability that any one is chosen is

$$p_H = \frac{|\Lambda|(|\Lambda| - 1)}{2(n-1)(|\Lambda| - |P|)|P|}, \tag{11}$$

for n, P, Λ satisfying $(|\Lambda| - |P|)|P| \geq |\Lambda|(|\Lambda| - 1)/(2n - 2)$. (This is violated when the feasible region is impractically small, and when the probability that a crossover is entirely feasible becomes very high.)

Like the probability p_F (9) that a feasible crossover is utilized, p_H cannot be assumed to be available. As a result, it is likewise approximated by the maintenance of an additional counter, t_h, of the number of half-feasible crossovers encountered in total. Then

$$p_H \approx \frac{t_c}{(n-1)t_h},$$

and it is this estimation that is generated and employed by the actual search algorithms.

The simplest modification accepts an entirely feasible or half-feasible crossover on the basis of the probabilities p_F and p_H, respectively. However, a half-feasible crossover is less desirable than one that produces two feasible candidate solutions, and should therefore be regarded accordingly. Any search that has already found at least one entirely feasible crossover subsequently ignores any half-feasible crossovers it may encounter. This is illustrated by the function 'SearchUp2' in Figure 5 (the function 'SearchDown2' that presumes the given component index to be the upper boundary of the crossover segment is similar).

```
[δ, k, type, found] = SearchUp2(x, y, r(x), r(y), h, p_F, p_H) {
  k = h;   δ = 0;
  found = 0;   lastk = 0;
  while (k < n and not found)
    δ = δ + [a_{i,k+1}]_{i=1}^{m} · (y_{k+1} − x_{k+1});   k = k + 1;
    if (−r(y) ≤ δ {
      if (δ ≤ r(x)) { //An entirely feasible crossover is found.
                type = 'f';
                if (rand() < p_F) found = 1
                else {lastδ = δ;   lastk = k;   lasttype = 'f'; };
                t_f = t_f + 1; }
      elseif (type ≠ 'f') { // Half-feasible crossover found.
                if (rand() < p_H) {found = 1;   type = 'y'; }
                else { lastδ = δ;   lastk = k;   lasttype = 'y'; };
                t_h = t_h + 1; }; }
      elseif (δ ≤ r(x) and type ≠ 'f') { // Half-feasible crossover found.
        if (rand() < p_H) { found = 1;   type = 'x'; }
        else { lastδ = δ;   lastk = k;   lasttype = 'x'; };
        t_h = t_h + 1; };
      t_c = t_c + 1; };
    if (not found and lastk > 0) {
      δ = lastδ;
      k = lastk;
      type = lasttype; }; }
```

Figure 5. Search for entirely feasible or half-feasible crossovers.

The possibility still exists, however, that a half-feasible crossover will be accepted, even when there is an entirely feasible crossover segment with the chosen boundary point; clearly, it is desirable that a half-feasible crossover is accepted only when no entirely feasible option can be found. This premature acceptance can only be avoided by first completing the search for a crossover producing two feasible solutions, and only if none is found, then accepting in its place a half-feasible crossover, if there is one. This is achieved by introducing additional local variables, to record a tentatively-accepted feasible crossover, to be utilized only if the search is completed in the absence of an entirely feasible crossover being discovered (see Figure 6).

The price paid for the assurance that a half-feasible crossover is accepted only in the absence of any that are entirely feasible is the relinquishment of possible early termination, when a half-feasible crossover is finally approved. In turn, the overheads associated with either of the search techniques modified to consider also half-feasible crossovers are greater than those associated with the original; the aptness of each depends upon the potential advantage gained, versus these increased costs.

```
[δ, k, type, found] = SearchUp3(x, y, r(x), r(y), h, p_F, p_H) {
  k = h;   δ = 0;
  found = 0;   lastk = 0;   hk = 0;
  while (k < n and not found)
    δ = δ + [a_{i,k+1}]_{i=1}^{m} · (y_{k+1} − x_{k+1});   k = k + 1;
    if (−r(y) ≤ δ {
      if (δ ≤ r(x)) { //An entirely feasible crossover is found.
            type = 'f';
            if (rand() < p_F) found = 1
            else {lastδ = δ;   lastk = k;   lasttype = 'f'; };
            t_f = t_f + 1; }
      elseif (type ≠ 'f' and hk = 0) { // Half-feasible crossover found.
            if (rand() < p_H) {hδ = δ;   hk = k;   htype = 'y'; }
            else { lastδ = δ;   lastk = k;   lasttype = 'y'; };
            t_h = t_h + 1; }; }
      elseif (δ ≤ r(x) and type ≠ 'f' and hk = 0) {
        // Half-feasible crossover found.
        if (rand() < p_H) { hδ = δ;   hk = k;   htype = 'x'; }
        else { lastδ = δ;   lastk = k;   lasttype = 'x'; };
        t_h = t_h + 1; };
      t_c = t_c + 1; };
    if (not found) {
      if (hk > 0 and lasttype = 'f')
        { δ = lastδ;   k = lastk;   type = lasttype; }
      elseif (hk > 0) { { δ = hδ;   k = hk;   type = htype; };
      elseif (lastk > 0) { δ = lastδ;   k = lastk;   type = lasttype; }; }; }
```

Figure 6. Search for crossovers without early termination.

When the probability that an entirely feasible crossover will be discovered is sufficiently small (see (1), (4), and Example 2), completing this search is unlikely to be useful. The additional benefits of a minuscule increase in the number of entirely feasible crossovers is, under these conditions, unlikely to offset the cost of almost always examining every possibility. That is, when the size of the feasible region is sufficiently dominated by that of its enclosing interval, the more casual method 'SearchUp2,' and its cousin 'SearchDown2,' are better suited.

For problems in which the feasible region is close to filling its bounding interval, the number of crossovers that are entirely feasible dominates the number that are half feasible (see (1), (4), and Example 2). The chance that a half-feasible crossover is accepted is remote, virtually eliminating the need to consider them at all. The behaviors of these modified functions that entertain the possibility of utilizing a half-feasible crossover are therefore both confluent to that of the original, entirely feasible only, scheme, but with the overheads associated with a more complicated algorithm. The original 'SearchUp1,' and its partner 'SearchDown1,' are most applicable in circumstances such as these.

When the expected numbers of entirely feasible and half-feasible crossovers are comparable, both the potential for discovering no entirely feasible crossover, and that for finding one after the acceptance of a half-feasible crossover, may be significant. While the half-feasible crossovers cannot be ignored, nor can the effects of prematurely terminating with only a half-feasible crossover; the 'SearchUp3,' the associated 'SearchDown3,' functions are most useful in problems in which the feasible region contains about two-thirds of the number of points as the smallest interval that envelops it (see (6)).

With the restriction that only feasible points be permitted to enter the new population, one of the solutions offered by a half-feasible crossover must be discarded; however, the crossover operation as a whole must produce two new offspring to replace their parents. One way to overcome this is to allow one (or possibly even both) of the parent solutions to survive. Typically, this might be performed by choosing the best parent, or by the flip of a coin biased suitably towards the best. An alternative is to choose the best two individuals from the three involved (that is, the two parents and the child), allowing the possibility that both parents survive and the crossover is effectively rejected.

This concept of choosing the result of crossover based on objective value is essentially a pre-emptive form of selection, and on a smaller scale. However, this diminutive selection discriminates harshly between individuals; this is a decision best left to the selection operator itself.

Furthermore, the decision to replace only one of the parent solutions with a child, and preserve the other, potentially results in a loss of genotypic information. While the parents may themselves differ in all component values, following the utilization of the half-feasible crossover in this way produces two solutions in the new population that often have a significant number of component values in common. This may be harmless or even beneficial in some problems, but possibly disastrous in others.

This concept can be extended, to allow the actual selection mechanism to determine, in effect, the overall crossover result, by allowing the population to temporarily grow (to at most half its own size again), and then selecting only the required number of individuals at the beginning of the following generation. The genotypic diversity of each population is thereby managed solely by the selection operator, as expressed in its design.

7. FEASIBLE MUTATION

The genetic selection and crossover operators together express most of the algorithm's ability to progress towards the realization of an optimal (or almost optimal) solution. However, with the utilization of these operators alone, it may occur that some potentially beneficial genetic information is lost.

A detrimental forfeit of knowledge might occur, for example, when some substructure of the optimum is found to be combined with another, overwhelmingly poor, partial solution. Overexuberence of the genetic algorithm in its quest may immediately eliminate this individual, and the desired substructure with it, from future consideration. Indeed, it may occur that no individual exists in the initial population that has some particular component value; perhaps this is a value that is required to construct an optimum (or a satisfactory near-optimal) solution.

A mutation operator, while secondary to the selection and crossover mechanisms, is fundamental because of its ability to overcome these situations. Also modeled on its biological counterpart, each component (allele) of every solution vector may be randomly altered, according to some small probability. Typically in genetic algorithms, this probability of mutation is of the order of one in several thousand.

The mutation operator on a binary alphabet is a simple flip of a bit. Here, where the algorithm operates on integer vectors, a mutation instead makes a random choice from the range of relevant values. In the context of this discussion, mutation is a function

$$\Psi : \mathbf{Z}^n \times \mathbf{Z} \times \chi \to \mathbf{Z}^v,$$

defined as

$$\Psi\left([x_1 \cdots x_n], v, k\right) = [x_1 \cdots x_{k-1}\, v\, x_{k+1} \cdots x_n].$$

This operator does not respect the limitations that must be placed on the value of v in order to maintain feasibility. Like the two-point crossover operator, it can be extended to make this possible.

In biological systems, the possibility exists that, through mutation, an individual may be developed that is unviable, resulting in its death. In the confines of the artificially controlled population size of the genetic algorithm, an infeasible solution generated by mutation must be replaced by the original, feasible, solution. While this is acceptable for lightly constrained problems, the rate of mutation in real terms is outside the direct control of the user, and must suffer significantly in highly-constrained examples, thereby reducing its effectiveness.

The mutation operator may also be viewed as an adaption and degeneration of crossover. Effectively, an individual is crossed with a random vector, with a crossover segment that consists only of the chosen allele. For this reason, the justification of the search for a feasible mutation takes a similar form to that of feasible crossover.

THEOREM 7.1. *The mutation*

$$x' = \Psi(x, \epsilon + x_k, k), \qquad \text{where } \epsilon \in \left\{\bar{l}_k - x_k, \ldots, \bar{u}_k - x_k\right\},$$

is feasible if and only if, for all $i : 1 \leq i \leq m$,

$$a_{ik} = 0 \vee \left(a_{ik} > 0 \wedge \epsilon \leq \frac{r(x)_i}{a_{ik}}\right) \vee \left(a_{ik} < 0 \wedge \epsilon \geq \frac{r(x)_i}{a_{ik}}\right).$$

PROOF. Let $x' = [x_1 \cdots x_n]$ be a feasible candidate solution. Then the result of the mutation $\Psi(x, \epsilon + x_k, k)$ is

$$\begin{aligned} x' &= [x_1 \cdots \epsilon + x_k \cdots x_n] \\ &= x + \epsilon e^k, \end{aligned}$$

where e^k is the k^{th} unit vector,

$$e^k = [e_i^k]_{i=1}^n \bullet e_i^k = \begin{cases} 0 & \text{if } i \neq k, \\ 1 & \text{if } i = k. \end{cases}$$

Requiring that $x' \in \Lambda$,

$$\begin{aligned} \bar{l}_k \leq \epsilon + x_k \leq \bar{u}_k, \\ \bar{l}_k - x_k \leq \epsilon \leq \bar{u}_k - x_k. \end{aligned}$$

Now requiring that x' is feasible,

$$\begin{aligned} Ax' &\leq b, \\ b - A\left(x + \epsilon e^k\right) &\geq 0, \\ (b - Ax) - \epsilon A e^k &\geq 0, \\ \epsilon \cdot [a_{ik}]_{i=1}^m &\leq r(x), \end{aligned}$$

since Ae^k is the k^{th} column of the constraint matrix A. Rewriting this in terms of each of the rows of $\epsilon \cdot [a_{ik}]_{i=1}^m$,

$$\forall i : 1 \leq i \leq n \bullet \epsilon a_{ik} \leq r(x)_i.$$

Then, if $a_{ik} > 0$ (and note here that x is feasible, so that $r(x) \geq 0$),

$$\epsilon \leq \frac{r(x)_i}{a_{ik}}.$$

Similarly, if $a_{i,k} < 0$, then

$$\epsilon \geq \frac{r(x)_i}{a_{ik}}.$$

When $a_{ik} = 0$, no additional restriction is placed on ϵ. ∎

Applying this result is simple (see Figure 7); given the component to be mutated, the search for a feasible mutation requires the determination of the range of values that ϵ, and therefore the replacement value, may take. Note also that the given solution x to be mutated is itself feasible, so that the upper bound on this range is nonnegative and the lower nonpositive, and the set of possible values is not empty, for trivially $\epsilon = 0$ produces a feasible point.

$[\epsilon] = \text{FeasibleMutation}(x,\ k,\ A,\ \bar{l},\ \bar{u})\ \{$
  $\epsilon_L = \bar{l}_k - x_k; \quad \epsilon_U = \bar{u}_k - x_k;$
  for $i = 1$ to m $\{$
    if $(a_{ik} > 0$ and $\epsilon_U \cdot a_{ik} > r(x)_k)$
      $\epsilon_U = \lfloor r(x)_i / a_{ik} \rfloor;$
    elseif $a_{ik} < 0$ and $\epsilon_L \cdot a_{ik} < r(x)_k)$
      $\epsilon_L = \lceil r(x)_i / a_{ik} \rceil;$ $\};$
  if $\epsilon_L < \epsilon_U$
    Choose at random $\epsilon : \epsilon_L \leq \epsilon \leq \epsilon_U \wedge \epsilon \neq 0;$
  else $\epsilon = 0;$ $\}$

Figure 7. Feasible mutation.

The component update so determined, the mutation itself can now be performed, by substituting the new value $\epsilon + x_k$ for the old x_k. The calling function is also free to decide whether to permit a trivial mutation, when $\epsilon = 0$, or to try another component, or a different candidate solution.

8. CONCLUSION

The clarification by a genetic algorithm of a discrete and linearly constrained optimization problem has been addressed, resulting in the proposal of several new genetic operators that preserve the feasibility of candidate solutions. The emanation is a methodology that is attractive and general, avoiding the pitfalls of penalty function, decoder and repair algorithms.

Through the theoretical investigation of the traditional two-point crossover, a deeper understanding of its behavior with respect to the feasible region has been gained. Crossover operations of this type applied to feasible points are found to fall into three categories: those that exclusively manufacture feasible solutions, those that produce no viable points, and those that produce one feasible and one infeasible point. The proportions of these are discovered to vary with the nature of the feasible region itself.

Upon this basis, an efficient crossover mechanism is developed that regards those crossovers that fabricate only feasible points. However, restricting attention to the entirely feasible variety may also, under certain conditions, unnecessarily limit the information processing ability of the genetic algorithm. This is particularly relevant for problems that are more highly constrained, in which the number of crossovers solely producing feasible points is greatly diminished. Two new crossover operators are proposed to alleviate this, and their effective application furthermore alludes to a transient increase in the actual population size. Finally, a mutation operator is offered that also guarantees the feasibility of its result.

REFERENCES

1. T. Bäch and F. Hoffmeister, Extended selection mechanisms in genetic algorithms, In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 92–99, (1991).
2. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, (1989).

3. Z. Michalewicz and C.Z. Janikow, Handling constraints in genetic algorithms, In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 151–157, (1991).
4. J.T. Richardson, M.R. Palmer, G. Liepins and M. Hilliard, Some guidelines for genetic algorithms with penalty functions, In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 191–197, (1989).
5. W. Siedlecki and J. Sklanski, 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, (1989).
6. G.A. Vignaux and Z. Michalewicz, A genetic algorithm for the linear transportation problem, *IEEE Transactions on Systems, Man, and Cybernetics* **21** (2) (1991).
7. D. Whitley, The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best, In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 116–121, (1989).
8. N. Ahitur and O. Berman, *Plenum Press*, John Wiley and Sons, New York, (1988).
9. V. Chvátal, *Linear Programming*, W.H. Freeman and Company, New York, (1983).
10. S.E. Elmaghraby, *Activity Networks: Project Planning and Control by Network Models*, John Wiley and Sons, New York, (1977).
11. M. Grotschel, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, (1988).
12. T.C. Hu, *Integer Programming and Network Flows*, Addison-Wesley, Reading, MA, (1969).
13. M. Iri, *Network Flow, Transportation, and Scheduling: Theory and Algorithms*, Academic Press, New York, (1969).
14. A. Kaufmann, *Integer and Mixed Programming: Theory and Applications*, Academic Press, New York, (1977).
15. R.P. Stanley, *Enumerative Combinatorics*, Volume I, Wadsworth and Brooks/Cole Advanced Books and Software, Belmont, CA, (1986).
16. J.E. Strum, *Introduction to Linear Programming*, Holden-Day, San Francisco, (1972).
17. K.A. DeJong, An analysis of the behavior of a class of genetic adaptive systems, Doctoral Dissertation, University of Michigan, *Dissertation Abstracts International* **36** (10) (1975).
18. D.E. Goldberg, Genetic and evolutionary algorithms come of age, *Communications of the ACM* **37** (3), 113–119 (March 1994).
19. J. Holland, Genetic algorithms and optimal allocation of trials, *SIAM Journal of Computing* **2** (2), 88–105 (1973).
20. J. Holland, *Adaption in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, (1975).
21. D.B. Fogel, Applying evolutionary programming to selected control problems, *Computers Math. Applic.* **27** (11), 89–104 (1994).
22. D.G. Conway and M.A. Venkataramanan, Genetic search and the dynamic facility layout problem, *Computers and Operations Research* **21** (8) (October 1994).
23. A.-L. Nordström and S. Tufekci, A genetic algorithm for the talent scheduling problem, *Computers and Operations Research* **21** (8) (October 1994).
24. G. Syswerda, Uniform crossover in genetic algorithms, In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 2–9, (1989).