ELSEVIER

European Journal of Operational Research 144 (2003) 333–347

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Discrete Optimization

Genetic search over probability spaces

Siddhartha Bhattacharyya [a,*], Marvin D. Troutt [b]

[a] *Information and Decision Sciences, College of Business Administration, University of Illinois at Chicago, 601 S. Morgan Street (MC 294), Chicago, IL 60607-7124, USA*

[b] *College of Business Administration, Graduate School of Management, Kent State University, P.O. Box 5190, Kent, OH 44242-0001, USA*

Received 23 April 1999; accepted 20 June 2001

Abstract

This paper proposes two new crossover operators for searching over discrete probability spaces. The design of the operators is considered in the light of recent theoretical insights into genetic search provided by forma analysis. A non-trivial test problem in enforcing coherency of probability estimates in cross-impact analysis highlights the utility of the designed operators. The presented operators will be useful for a variety of probability-fitting and optimization applications.

Keywords: Genetic algorithms; Probability space optimization; Forma analyses

1. Introduction

Probability estimates are essential input data for many decision support techniques, including expert systems, long range forecasting, decision analysis and cross-impact analysis, among others. This paper suggests a genetic algorithm (GA)-based approach for searching over discrete probability spaces that will be useful for a variety of probability-fitting and optimization applications.

This research undertakes the design of genetic operators for effective search over a space of probability distributions. That is, we consider decision sets of the form,

$$\left\{x \in \mathbb{R}^n,\ x = (x_1, \dots, x_n),\ x_i \geqslant 0,\ \sum x_i = 1\right\}$$

which may be called the probability simplex of $\mathbb{R}^n$. This set occurs frequently in a variety of optimization problems. For example, if a convex set $C \subset \mathbb{R}^m$ has extreme points $\{e_1, \dots, e_n\}$, then the x_i give the so-called barycentric coordinates of points in C. Most practical optimization problems have convex polyhedral decision sets, which may be represented in this way. The possible mixed strategies of the players in games with finitely many strategies may be described by these sets. Such sets also represent the possible probability distributions over finitely many events or States of Nature.

Our focus in this paper is on optimization problems over the probability simplex using genetic search, and more particularly on the design of crossover and mutation operators on this space.

* Corresponding author.
E-mail address: sidb@uic.edu (S. Bhattacharyya).

0377-2217/03/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0377-2217(01)00401-5

Given the ubiquity of these sets in optimization and the increasing use of genetic search approaches, understanding of design considerations should prove useful. We illustrate the concepts and issues with the help of a particular such problem of finding coherent probability estimates for a set of related events as in cross-impact analysis (see [1,3,7,8,14]). This problem is described in detail in a later section. We hope this application setting will be of independent interest in its own right. The problem cannot be solved reliably, in general, by mathematical programming techniques due to potential convergence to local optima. However, we show one such attempt as a base of comparison for the genetic algorithm approach.

The regular genetic search operators are inappropriate for searching on the probability simplex, given the restriction that each solution, representing a probability distribution over a set of events, should specify probabilities of events that sum to one. Using the regular *n*-point or uniform crossover operators, for instance, can result in solutions that violate the unit sum requirement. While the resulting solution can be subsequently re-normalization to one, such an operator is undesirable since the resulting solution is not likely to retain characteristics of the original distributions. Similar problems with the regular GA operators arise in the case of sequence-based solutions, as in the traveling salesman problem, where specialized crossover operators need to be devised [6,13]. For the search to be effective, crossover operators should be designed to combine good building blocks of solutions.

The design of genetic search operators is considered here in the light of recent theoretical insights provided by forma analyses [17–21,24]. Forma theory provides an extension of the original schema analysis – for bit-string representations – of genetic algorithms to arbitrary representations, and suggests certain principles for the design of effective genetic search operators.

We consider a direct real number encoding of discrete probability distributions. Two crossover operators are developed, one in explicit consideration of forma processing principles, and the other a more "intuitive" operator along the lines of traditional single-point crossover. Both operators are analyzed in terms of the forma that they process, and their performance is evaluated on a non-trivial test problem.

Several books [2,6,10,13,15] provide thorough accounts of the mechanics of genetic search. Section 2 presents a brief overview of forma related principles for the design of genetic search operators. The design of two crossover operators and a mutation operator for searching over probability spaces are considered in Section 3. Section 4 then explains and develops the test problem, and is followed by a performance evaluation of the genetic search operators implemented.

2. Forma and design principles

The reader is directed to [17] for the details of Forma theory. This section refers to the concepts of equivalence, respect and assortment that are key to the material in this paper. Definitions are given next for the convenience of the reader, and the following section illustrates these in the context of two crossover operators to be considered.

In Forma theory, domain knowledge is expressed through equivalence relations defined on the search space. Any relation between pairs of solutions that satisfy reflexivity, symmetry and transitivity can be used. These equivalence relations then induce a partitioning of the search space into equivalence classes, or formae. Solutions within a forma are thus equivalent under the equivalence relation. Equivalence relations define the genes, with corresponding forma defining the alleles in the string representation.

Consider S be the search space, let ξ denote some forma on S and let Ξ be the set of formae considered. Then Ξ is a subset of the power set of S. A set of formae Ξ is said to *cover* a search space if any potential solution (population instance) is uniquely identified by the forma. Formally, Ξ covers S if and only if $\forall x \in S$, $\exists \xi_x \subseteq \Xi$ with $\bigcap_x \xi_x = x$. Coverage ensures that forma analysis can distinguish between solutions. Two forma ξ_1 and ξ_2 are *compatible* if a solution can belong to both. Two forma will thus be compatible if their intersection is non-empty, i.e. if $\xi_1 \cap \xi_2 \neq \emptyset$. It is desirable that compatible forma be *closed under* *intersection*. This allows formae to specify solutions with different degrees of accuracy and thereby gradually refine the search.

Any genetic operator R that recombines two parents in S to produce an offspring in S can be defined through a function $R: S \times S \times P_R \rightarrow S$, where P_R is the control set that determines exactly which of the various possible offspring results from application of the operator. The non-deterministic nature of typically used genetic operators can be specified through some appropriate choice of control parameters in P_R. The control set for an operator need not be explicitly defined; any recombination operator that manipulates two parents to produce offspring in a reasonable manner can be used.

In addition to the coverage and closure properties defined above, forma analysis provides three design principles for the genetic recombination operators: *respect*, *assortment*, *and ergodicity*. These are defined below:

Respect: A recombination operator R is said to respect a set of forma Ξ if recombining two instances of a particular forma produces an instance of the same forma. Formally R respects Ξ if and only if

$$\forall \xi \in \Xi,\ \forall f, g \in \xi,\ \forall p \in P_R : \quad R(f, g, p) \in \xi.$$

Respect seeks to preserve genes common to both parents in their offspring as well. It is effective at reducing forma disruption, especially towards the later stages of the search when population diversity is low.

Assortment: A recombination operator R is said to assort a set of forma Ξ if, given two compatible forma, it is possible to recombine any two instances of these to produce an offspring that is an instance of both formae. Formally, R assorts Ξ if and only if

$$\forall \xi_1, \xi_2 \in \Xi \quad \text{such that } \xi_1 \cap \xi_2 \neq \emptyset,\ \forall f \in \xi_1,$$
$$\forall g \in \xi_2,\ \exists p \in P_R : \quad R(f, g, p) \in \xi_1 \cap \xi_2.$$

The assortment property relates to the building block hypothesis, and seeks to ensure that good building blocks from two parents *can be* effectively combined.

Ergodicity: Given any population, it should be possible to access any point in the search space through a finite sequence of applications of the genetic search operators. The mutation operator is generally used to provide for ergodicity.

A further property of formae arising from respect and assortment is separability. A set of forma, Ξ, is said to be *separable* if a recombination (crossover) operator capable of respecting and assorting it exists, i.e. if the properties of respect and assortment are not contradictory. While desirable, the twin properties of respect and assortment, as indicated earlier, are incompatible in certain practical problems. Noting that most standard crossover operators respect schemata, Radcliffe [17] writes:

> Thus if a respectful operator is used, then non-separability immediately means that assortment must fail to be achieved. The significance of this is that even when parents are chosen that contain all the genetic material (apparently) necessary to build some given child, it may be impossible for a respectful operator to construct that child. As well as being in conflict with reason, this would seem to make navigation around the search space unnecessarily difficult.

Thus, where forma are non-separable, assortment would seem to be the more desirable property sought; respect increases in relevance towards the later stages of search, as the population begins to converge.

The forma properties and GA design principles given above, however, should not be considered as providing necessary or sufficient conditions for effective genetic search [17]. Rather, they provide a set of conditions which, if satisfied, is expected to yield improved search capabilities.

3. Crossover and forma in probability distributions

Two crossover operators are designed for genetic search over probability distribution spaces. The first is an "intuitive" operator that combines the "front" and "back" ends of two parent distributions. The second operator is designed in explicit consideration of the design principles given above. This section analyses both operators with respect to the forma that they process.

As noted in Section 1, we consider genetic search over the probability simplex. Both operators are defined directly on the search space. A population member thus represents a solution directly as a string of real numbers, each number corresponding to the probability of a specific event. Probability distributions impose the constraint that each value be in $[0,1]$ and that all values sum to one. Thus, considering a problem with N events, a solution is represented as the string:

$$\langle s_1, s_2, \ldots, s_N \rangle \quad \text{with } s_i \in [0,1], \; i = 1, \ldots, N, \quad \sum_{i=1}^{N} s_i = 1.$$

3.1. Probability interval forma

Forma based on intervals in the real line have been suggested for optimizing functions of real values [20]. Considering a single-valued function, any interval $[a,b]$, where $a, b \in \mathbb{R}$ forms an equivalence class. For n real-valued parameters, the set of formae may be defined as the cartesian product:

$$\Xi = \prod_{i=1}^{n} I_i,$$

where $I_i = [a_i, b_i] \cup \{*\}$; here, the $*$ is the do not care symbol. Such forma have been called locality forma in [20]. Where the parameters are probabilities and forma correspond to distributions, we have the added restriction that all $a_i, b_i \in [0,1]$ and that valid instances of forma must have probability-values summing to one. For this, note that the lower ranges of all specified positions must sum to no more than one, and the upper ranges must sum to at least one,

$$\text{i.e., } \forall i, \quad \text{where } I_i \neq *, \; \sum_i a_i \leqslant 1$$

and

$$\forall i, \quad \text{where } I_i \neq *, \sum_i b_i \geqslant 1. \tag{a}$$

A solution s is then an instance of a forma ξ if and only if $\forall i$ where $I_i \neq *, s_i \in \xi_i$.

Example 1. For $n = 4$, $\xi = \langle [0.1, 0.4], *, [0.5, 0.6], * \rangle$ is a forma, and any solution $s = \langle s_1, s_2, s_3, s_4 \rangle$ will be an instance of ξ if $s_1 \in [0.1, 0.4]$ and $s_3 \in [0.5, 0.6]$. Thus $\langle 0.2, 0.1, 0.5, 0.2 \rangle \in \xi$.

Two probability interval subforma are compatible if they share a common interval. For instance, considering two sub-forma $\xi_1 = [0.2, 0.6]$ and $\xi'_1 = [0.4, 0.8]$, $\xi_1 \cap \xi'_1 = [0.4, 0.6]$; the two subforma are compatible since they have a non-null intersection. Considering these as subforma of two parents ξ and ξ', and considering only the probabilities of events $i = 1$ (i.e. ξ_1 and ξ'_1), assortment then demands that a crossover operator be able to produce offspring with the probability of event $i = 1$ lying within this intersection. Respect requires that given two instances with the probability of event i being in a certain interval, all offspring will also have the probability of event i in the same interval.

Note that sub-forma compatibilty is not sufficient for compatibility of the associated formae. For two probability interval formae to be compatible, it is also necessary that the restrictions (a) above hold for the intersection of the considered formae.

Example 2. With forma $\xi_1 = \langle [0.0, 0.4], [0.1, 0.3], [0.6, 1.0] \rangle$ and $\xi'_1 = \langle [0.4, 1.0], [0.0, 0.5], [0.0, 0.6] \rangle$, their intersection $\xi_1 \cap \xi'_1 = \langle [0.4, 0.4], [0.1, 0.3], [0.6, 0.6] \rangle$. Since no instance of this can satisfy (a), the forma are deemed incompatible.

3.2. A crossover operator for probability interval forma

Consider a crossover operator $X^{\mathrm{PI}} : S \times S \times P^{\mathrm{PI}} \to S$. The control parameters P^{PI} determine which string positions participate in the crossover. We consider $P^{\mathrm{PI}} = [0,1]^n$ and uniformly chosen as in uniform crossover [25]. Thus, a 1 at a string position indicates a crossover of values between the two parent strings at that position. Let $C = \{i : [0,1]_i = 1\}$ be the set of crossover sites chosen and $X = \{i : [0,1]_i = 0\}$ be the set of string positions that are carried over intact into the offspring. Then, considering parents $f = \langle f_1, \ldots, f_n \rangle$

and $g = \langle g_1, \ldots, g_n \rangle$ the offspring $p = \langle p_1, \ldots, p_n \rangle$ and $q = \langle q_1, \ldots, q_n \rangle$ are obtained as follows:

For $i \in X$, $\quad p_i = f_i \quad$ and $\quad q_i = g_i$.

For $i \in C$, $\quad p_i \in [f_i, g_i]$ such that

$$\sum_{i \in C} p_i = 1 - \sum_{i \in X} f_i \tag{b}$$

and

$$q_i \in [f_i, g_i] \quad \text{such that} \sum_{i \in C} q_i = 1 - \sum_{i \in X} g_i. \tag{c}$$

Note that given two parents, all choices of control parameter P^{PI} may not yield legal offspring. This happens when $\forall i \in C$, $f_i > g_i$, or $f_i < g_i$; in such cases, the conditions (b) and (c) cannot be satisfied.

The crossover operator X^{PI} both respects and assorts probability interval forma. Respect is observed by noting that if f_i and g_i are both instances of sub-forma ξ_i, the interval $[f_i, g_i]$ is also in ξ_i; since X^{PI} produces offspring with ith members either retaining their parental values or taking some value in $[f_i, g_i]$, this value must be in ξ_i. For assortment, observe that if $f_i \in [a_i, b_i], g_i \in [k_i, l_i]$, then if the sub-forma are compatible, their intersection must be non-null; let $[a_i, b_i] \cap [k_i, l_i] = [c_i, d_i]$. Since $[f_i, g_i] \cap [c_i, d_i]$ cannot be null, it is always possible to obtain offspring that lie in both the subformae. For compatible forma, the same rationale holds for all string positions i.

The X^{PI} crossover operator can be implemented as follows:

1. Select crossover positions uniformly. As before, let $C = \{i : [0,1]_i = 1\}$ be the selected crossover sites and $X = \{i : [0,1]_i = 0\}$ be those positions not selected for crossover.
2. Determine $\Delta_i^+ = \delta(f_i - g_i)$, $i \in C$ and $\Delta_i^- = \delta(g_i - f_i)$, $i \in C$.
   Let $C^+ = \{i \in C : \delta(f_i - g_i) = 1\}$ and $C^- = \{i \in C : \delta(g_i - f_i) = 1\}$.
   Here,
   $$\delta(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$
   Calculate the sums $\Delta^+ = \sum_{i \in C^+} \Delta_i^+$ and $\Delta^- = \sum_{i \in C^-} \Delta_i^-$.
3. If $\Delta_{\min} = \min\{\Delta^+, \Delta^-\} > 0$, then the two parents f and g are crossover compatible with respect to the chosen crossover sites in C. If found incompatible, go to step 1. Give up after some fixed number of trials.
4. Choose $\Delta = \mathrm{U}(0, \Delta_{\min})$, where the $\mathrm{U}(\cdot)$ denotes a uniform random distribution.
5. Let C_k^+ and C_k^-, $k = \{1, \ldots, n\}$ denote the kth element of C^+ and C^-, respectively.
   For each $k \in C^+$, select increments $d_{C_k^+} = \mathrm{U}(d_{C_{k-1}^+}, \partial - d_{C_{k-1}^+})$, where $\partial = \partial + d_{C_k^+}$ keeps the running totals of increments selected so far. Here, $d_{C_0^+} = 0$.
   Similarly, for each $k \in C^-$, select increments $d_{C_k^-} = \mathrm{U}(d_{C_{k-1}^-}, \partial - d_{C_{k-1}^-})$, where $\partial = \partial + d_{C_k^-}$ and $d_{C_0^-} = 0$.
6. Form the offspring:
   $$\forall k \in C^+, \quad p_k = f_k + d_{C_k^+},$$
   $$\forall k \in C^-, \quad p_k = f_k - d_{C_k^-},$$
   $$\forall i \notin C, \quad p_k = f_k.$$
7. Repeat steps 1–6 to obtain the second offspring $q = \langle q_1, \ldots, q_n \rangle$ in a similar manner.

Example 3. Consider $f = \langle 0.1, 0.4, 0.3, 0.2 \rangle$ and $g = \langle 0.4, 0.1, 0.1, 0.4 \rangle$ and let the crossover sites chosen be $C = \{1, 3\}$. Here, $C^+ = \{1\}$ and $C^- = \{3\}$ and $\Delta^+ = 0.3$ and $\Delta^- = 0.2$, and $\Delta_{\min}$ is thus 0.2. If $\Delta = \mathrm{U}(0, 0.2) = 0.15$, then the offspring produced is $p = \langle 0.25, 0.4, 0.15, 0.2 \rangle$. Similarly, with $\Delta = \mathrm{U}(0, 0.2) = 0.08$, the second offspring produced is $q = \langle 0.32, 0.1, 0.18, 0.4 \rangle$.

3.3. Distribution-swap crossover

A second crossover operator is modeled, not through forma processing considerations, but along the lines of traditional GA crossover operators that swap alleles in two parents to obtain two offspring. Here, the offspring are formed by exchanging the "front" and "tail" ends of the parent distributions.

A cumulative distribution function (CDF) representation facilitates the design of this operator. Each population member represents a CDF corresponding to the probability distribution

$x = \langle x_1, x_2, \ldots, x_n \rangle$. Then from any population member $S(x)$, the probabilities can be determined as

$$x_i = S\left(\frac{i}{n}\right) - S\left(\frac{i-1}{n}\right), \quad i = 1, \ldots, n$$

with $S(0) = 0$ and $S(1) = 1$ being fixed. Thus each population member is represented as a n-component vector $(x_1, \ldots, x_n)$ whose coordinates are monotone increasing to the right.

Given two parents $F(x)$ and $G(x)$, the distribution-swap crossover operator obtains two offspring $P(x)$ and $Q(x)$ as follows:

1. Select a crossover site $x_c = \mathrm{U}[0,1]$ (uniformly in $[0,1]$).
2. $P(x) = \begin{cases} F(x) & \text{if } x \leqslant x_c, \\ F(x_c) + (1 - F(x_c))\left[\frac{G(x) - G(x_c)}{1 - G(x_c)}\right] & \text{if } x > x_c. \end{cases}$
3. $Q(x) = \begin{cases} G(x) & \text{if } x \leqslant x_c, \\ G(x_c) + (1 - G(x_c))\left[\frac{F(x) - F(x_c)}{1 - F(x_c)}\right] & \text{if } x > x_c. \end{cases}$

(This CDF decomposition was employed earlier, in non-GA related work [27].)

Note that this operator considers distributions as continuous, and crossover sites can fall in between two defined positions on a string. This is easily avoided, where desirable, by choosing $x_c = (r/n)$, where $r = \mathrm{U}[1, n]$; this ensures that the crossover site corresponds to some string position. The operator also exhibits a high positional bias [4] as associated with traditional single-point crossover. This can be overcome in a uniform version of the operator (UDSX) defined as follows:

1. Considering probabilities $(x_1, \ldots, x_n)$, select crossover positions uniformly ($[0,1]^n$). Let C be the set of selected crossover positions.
2. Rearrange the ordering of the probabilities x_i such that $\{x_i \in C\}$ appears to the left of $\{x_i : i \in C\}$. Transform this rearranged probability distribution to CDF form and perform crossover with crossover site $x_c = (\max\{i : i \in C\}/n)$. If x_c is not required to correspond to any defined string position, it may be selected uniformly in $[x_c, x_{c+1}]$.

It is readily observed that this distribution-swap crossover neither respects nor assorts probability interval schema. This operator, instead, preserves the probability ratios amongst events in the front and back ends of a distribution, i.e., on either side of the crossover site x_c.

Example 4. Given two parent probability distributions $f = \langle f_1, f_2, f_3, f_4 \rangle$ and $g = \langle g_1, g_2, g_3, g_4 \rangle$, and with the crossover site chosen to be, say, at the second string position, offspring $p = \langle p_1, p_2, p_3, p_4 \rangle$ will maintain $p_1/p_2 = f_1/f_2$ and $p_3/p_4 = g_3/g_4$, and similarly for the second offspring.

The distribution-swap crossover operator may thus be considered as processing *Probability-ratio forma*.

3.4. Probability-ratio forma

Probability-ratio formae relate distribution strings having the same probability-ratios for the defining events. A probability distribution $f = \langle f_1, \ldots, f_n \rangle$ can be represented with its probability-ratio equivalent

$$f' = \left\langle \ldots, \frac{f_i}{f_{i+1}}, \frac{f_i}{f_{i+2}}, \ldots \frac{f_i}{f_n}, \ldots \right\rangle.$$

The set of probability-ratio forma can be defined as $\Xi = \prod_j F_j$, where

$$F_j = \left\{ \frac{f_k}{f_r}, k, r \in P \right\} \cup \{*\};$$

here P denotes the set of events with specified probabilities in the distribution, and $*$ is a do not care symbol corresponding to events with unspecified probabilities. Then, two strings f and g (distributions, or their probability-ratio equivalents) are equivalent or instances of a forma ξ^R according to:

$$f, g \in \xi^R \iff \exists P \subseteq \{1, \ldots, n\},$$

$$\frac{f_p}{f_q} = \frac{g_p}{g_q} \quad \forall p, q \in P.$$

Example 5. Considering probability distributions $f = \langle .2, .3, .4, .1 \rangle$ and $g = \langle .1, .6, .2, .1 \rangle$, their probability-ratio equivalents are $f' = \langle \frac{2}{3}, \frac{2}{4}, \frac{2}{1}, \frac{3}{4}, \frac{3}{1}, \frac{4}{1} \rangle$ and $g' = \langle \frac{1}{6}, \frac{1}{2}, \frac{1}{1}, \frac{6}{2}, \frac{6}{1}, \frac{2}{1} \rangle$. Thus, both f and g are instances of the forma $\langle *, \frac{1}{2}*, *, *, * \rangle$.

The terms f_k and f_r above can, in general, be considered as real intervals in $[0, 1]$. The correspondence between probability-interval forma and probability-ratio forma is then easily established: given probability-interval forma $\xi = \langle [a_1, b_1], *, [a_3, b_3], * \rangle$, the corresponding probability-ratio forma is

$$\xi^R = \left\langle *, \frac{f_1}{f_3}, *, * \right\rangle$$

with $f_1 \in [a_1, b_1], f_3 \in [a_3, b_3]$.

Distribution-swap crossover does not respect probability-ratio forma, as seen in the following example.

Example 6. Consider parents $f' : \langle 0.1, 0.3, 0.4, 0.2 \rangle$ and $g : \langle 0.2, 0.1, 0.3, 0.4 \rangle$. Note that $f_1/f_4 = g_1/g_4 = 0.5$, and a probability-ratio respecting operator will preserve this ratio in offspring as well. However, distribution-swap crossover, with crossover site at, say, the second position yields offspring $\langle 0.1, 0.3, 0.26, 0.34 \rangle$ and $\langle 0.2, 0.1, 0.47, 0.23 \rangle$, none of which maintain the ratio between the first and fourth event probabilities.

Respect here, however, rises with increasing similarity in gene values as the search progresses towards convergence. Since respect is an important property in the later stages of search, distribution-swap crossover can thus still be expected to perform well.

The uniform version of the distribution-swap crossover operator (UDSX) also assorts probability-ratio forma.

Example 7. Consider parents $f' : \langle 0.1, 0.3, 0.4, 0.2 \rangle$ and $g : \langle 0.2, 0.1, 0.3, 0.4 \rangle$, or their probability ratio equivalents $f' : \langle \frac{1}{3}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{3}{2}, 2 \rangle$ and $g' : \langle 2, \frac{2}{3}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{3}{4} \rangle$. Here f is an instance of forma $\xi_1 : \langle \frac{1}{3}, *, \frac{1}{2}, *, *, 2 \rangle$ and g is an instance of the forma $\xi_2 : \langle *, *, \frac{1}{2}, *, \frac{1}{4}, * \rangle$. Now instances f and g of the above formae can be recombined by distribution-swap crossover to give offspring that are instances of both formae. Note that $\xi_1 \cap \xi_2 = \langle *, *, \frac{1}{2}, *, *, * \rangle$, i.e., having $x_1/x_4 = 0.5$. Instances of this intersection forma are always obtainable from parents f and g by considering events x_1 and x_4 (first and fourth positions of the string) in the crossover set C (see description of the uniform version of the operator above).

3.5. Mutation

As noted earlier, mutation plays a key role in implementing the ergodicity property. With both the crossover operators above, mutation is designed to randomly change the probability of any event. The probabilities of the other events will also then need to be adjusted so as to satisfy the probability distribution restriction (summation to unity). These adjustments are made proportional to their existing values.

In addition, noting that both crossover operators carry an inherent bias away from the endpoints (0 and 1 probability values), the mutation operator is implemented to insert a 0-value to randomly selected events. This 0-value mutation shares equally with the regular mutation. Since a mutation to a value of 1 would reduce the probabilities of all the other events to 0, and is thus, in general, not expected to yield good solutions, we do not implement a 1-value mutation.

4. Test problem: Enforcing coherency of probability estimates

We examine the performance of the two genetic search operators on a problem of obtaining coherent probabilities estimates for a set of related events. Obtaining high quality probability estimates from decision-makers is a problem that occurs with many decision support models and methods. Such estimates, particularly those of related probabilities, can fail to be consistent with the laws of probability theory, a situation known as incoherency [9,16,23,28,29]. Here, we first develop a search space for representing related probabilities in such a way that coherent scenario and event probability estimates can be obtained. Genetic search then provides a general approach for finding estimates which are exactly coherent while being close to the decision maker's estimates as well. The constructed problem is also amenable to mathematical programming solution approaches, except that global optima may not be obtained, in general. However, we employ one mathematical programming attempt as a means for comparing with the designed genetic search operators.

The example problem we consider may be described as follows. Given data of the type shown in Table 1, find coherent probability estimates for the various possible scenarios and events which are as nearly consistent as possible with the assignments of Table 1. Our starting point is an interpretation of the constraints in Table 1 as relationships between conditional probabilities and marginal probabilities as follows: the first constraint is interpreted as requiring

$$\mathrm{P}(B|A) = \mathrm{P}(B) + 0.25(1 - \mathrm{P}(B)). \tag{1}$$

Table 1
Data for four hypothetical interrelated events

1	If event A occurs then the probability of event B is increased by 25% of its allowable increase
2	If event A occurs then the probability of event C is decreased 10% (or as near to 10% as possible)
3	If event D does not occur, then the probability of event C is increased by 30% of its allowable increase
4	If event D occurs, then the probability of event A is decreased 50%
5	If event C occurs, then the probability of event D is increased 20% of its allowable increase
6	If event B occurs, then the probability of event D is decreased 30% of its allowable decrease
	Initial probability estimates:
	$\mathrm{P}(A) = .50$
	$\mathrm{P}(B) = .40$
	$\mathrm{P}(C) =$ Not specified
	$\mathrm{P}(D) = .10$

Similarly the next two give rise to

$$\mathrm{P}(C|A) = \mathrm{P}(C) - 0.10, \tag{2}$$

$$\mathrm{P}(C|\overline{D}) = \mathrm{P}(C) + 0.3(1 - \mathrm{P}(C)), \tag{3}$$

respectively.

The remaining three equations are, respectively,

$$\mathrm{P}(A|D) = 0.5\mathrm{P}(A), \tag{4}$$

$$\mathrm{P}(D|C) = \mathrm{P}(D) + 0.2(1 - \mathrm{P}(D)), \tag{5}$$

$$\mathrm{P}(D|B) = 0.3\mathrm{P}(D). \tag{6}$$

Here we use the notation $\mathrm{P}(\cdot)$ to denote the probability of event $(\cdot)$ and use the overbar to denote complementary events.

To develop the representation, consider a special type of sample space, S, whose sample points are potentially elements of one or more of the event sets A, B, C, and D. A particular point in S can belong to one and only one of the $2^4 = 16$ mutually exclusive subsets or scenarios characterized according to whether the point is or is not an element of each of the four sets A, B, C, and D. In fact, we may list and number these scenarios as in Table 2. Also in Table 2 we associate a variable x_i to represent the relative frequency of sample points in scenario i. We call S a flexible sample space (FSS). That is, instead of each sample space point having equal probability, these points are allowed to have flexible probabilities denoted by the x_i. [1]

Since $\sum_{i=1}^{16} x_i$ must be unity, the reader may check that the relevant marginal and joint events involved in system (1)–(6) may be expressed as follows:

$$\mathrm{P}(A) = \sum_{i=1}^{8} x_i, \tag{7}$$

$$\mathrm{P}(C) = \sum_{i=1}^{2} x_i + \sum_{i=5}^{6} x_i + \sum_{i=9}^{10} x_i + \sum_{i=13}^{14} x_i, \tag{8}$$

[1] The FSS approach appears to be a relatively straightforward modeling tool which may have been used elsewhere. It appears to be essentially the same as what some have called the minimal relevant sample space.

Table 2
Scenarios and their relative frequency variables

Scenario number	Relative frequency	Scenario[a]
1	x_1	$ABCD$
2	x_2	$ABC\overline{D}$
3	x_3	$AB\overline{C}D$
4	x_4	$AB\overline{C}\overline{D}$
5	x_5	$A\overline{B}CD$
6	x_6	$A\overline{B}C\overline{D}$
7	x_7	$A\overline{B}\overline{C}D$
8	x_8	$A\overline{B}\overline{C}\overline{D}$
9	x_9	$\overline{A}BCD$
10	x_{10}	$\overline{A}BC\overline{D}$
11	x_{11}	$\overline{A}B\overline{C}D$
12	x_{12}	$\overline{A}B\overline{C}\overline{D}$
13	x_{13}	$\overline{A}\overline{B}CD$
14	x_{14}	$\overline{A}\overline{B}C\overline{D}$
15	x_{15}	$\overline{A}\overline{B}\overline{C}D$
16	x_{16}	$\overline{A}\overline{B}\overline{C}\overline{D}$

[a] For example, $A\overline{B}C\overline{D}$ may be described as the scenario in which events A and C both occur, while B and D fail to occur.

$$\mathrm{P}(B) = \sum_{i=1}^{4} x_i + \sum_{i=9}^{12} x_i, \tag{9}$$

$$\mathrm{P}(D) = x_1 + x_3 + x_5 + \cdots + x_{15}, \tag{10}$$

$$\mathrm{P}(B \cap A) = \sum_{i=1}^{4} x_i, \tag{11}$$

$$\mathrm{P}(C \cap A) = \sum_{i=1}^{2} x_i + \sum_{i=5}^{6} x_i, \tag{12}$$

$$\mathrm{P}(C \cap \overline{D}) = x_2 + x_6 + x_{10} + x_{14}, \tag{13}$$

$$\mathrm{P}(A \cap D) = x_1 + x_3 + x_5 + x_7, \tag{14}$$

$$\mathrm{P}(D \cap C) = x_1 + x_5 + x_9 + x_{13}, \tag{15}$$

$$\mathrm{P}(D \cap B) = x_1 + x_3 + x_9 + x_{11}. \tag{16}$$

The requirements of Table 1 may now be expressed as follows after some simple algebra:

$$\sum_{i=1}^{4} x_i = 0.25 \sum_{i=1}^{8} x_i + 0.75 \left(\sum_{i=1}^{4} x_i + \sum_{i=9}^{12} x_i \right) \left(\sum_{i=1}^{8} x_i \right), \tag{17}$$

$$\sum_{i=1}^{2} x_i + \sum_{i=5}^{6} x_i + 0.1 \sum_{i=1}^{8} x_i = \left(\sum_{i=1}^{2} x_i + \sum_{i=5}^{6} x_i + \sum_{i=13}^{14} x_i \right) \left(\sum_{i=1}^{8} x_i \right), \tag{18}$$

$$x_2 + x_6 + x_{10} + x_{14} = 0.3(x_2 + x_4 + \cdots + x_{16}) + 0.7(x_2 + x_4 + \cdots + x_{16}) \times \left(\sum_{i=1}^{2} x_i + \sum_{i=5}^{6} x_i + \sum_{i=9}^{10} x_i + \sum_{i=13}^{14} x_i \right), \tag{19}$$

$$x_1 + x_3 + x_5 + x_7 = 0.5 \left(\sum_{i=1}^{8} x_i \right) (x_1 + x_3 + \cdots + x_{15}), \tag{20}$$

$$x_1 + x_5 + x_9 + x_{13} = (0.2 + 0.8(x_1 + x_3 + \cdots + x_{15})) \times \left(\sum_{i=1}^{2} x_i + \sum_{i=5}^{6} x_i + \sum_{i=9}^{10} x_i + \sum_{i=13}^{14} x_i \right), \tag{21}$$

$$x_1 + x_3 + x_9 + x_{11} = 0.3(x_1 + x_3 + \cdots + x_{15}) \left(\sum_{i=1}^{4} x_i + \sum_{i=9}^{12} x_i \right), \tag{22}$$

$$\sum_{i=1}^{8} x_i = 0.5, \tag{23}$$

$$\sum_{i=1}^{4} x_i + \sum_{i=9}^{12} x_i = 0.4, \tag{24}$$

$$x_1 + x_3 + \cdots + x_{15} = 0.1. \tag{25}$$

The last three requirements, (23)–(25), are due to the initial marginal probability estimates in the lower half of Table 1.

Thus the FSS decomposition describes a hypothetical sample space in terms of the possible elementary scenarios related to the particular set of events of interest. All possible joint, marginal and conditional probabilities of these events can therefore be modeled in a coherent way. It is important to emphasize here that coherency is strictly enforced by this scheme. That is, all information about the events A, B, C, D, their intersections, complements, and other set operations, is contained in the x_i, $i = 1, \ldots, 16$. The reader may check that coherency requirements, such as $\mathrm{P}(A \cap B) = \mathrm{P}(A|B)\ \mathrm{P}(B)$, reduce to identities in the x_i. Moreover the above points remain true even if the system (17)–(25) is solved only approximately. Thus the FSS model used here precisely enforces coherency but permits some possible lack of fit with the decision maker's data such as given by Table 1.

Eqs. (17)–(25) constitute a set of nine constraints, some of which are nonlinear, in 16 variables. In addition, it is necessary that $\sum_{i=1}^{16} x_i = 1.0$ and $x_i \geqslant 0$, $i = 1, \ldots, 16$. Hence, it is not known a priori whether an exact solution exists. We might therefore seek scenario probabilities x_i which come as close to these conditions as possible in the least-squared error sense. Define e_1 as the difference of left- and right-hand sides of (17), e_2 similarly for (18) and so on to e_9 for (25). This line of reasoning would lead to the constrained programming problem (26) and (27) as follows:

$$\min \quad \sum_{i=1}^{9} e_i^2 \tag{26}$$

$$\text{s.t.} \quad \sum_{i=1}^{16} x_i = 1, \quad x_i \geqslant 0, \ i = 1, \ldots, 16. \tag{27}$$

This problem was coded and solved using GAMS. Table 3 shows the results for both the scenario probabilities and the probabilities of events A, B, C, and D.

Note that the objective function in (26) cannot easily be checked for convexity. The GAMS solution thus might not guarantee a global optimum. In order to compute a global minimum of (26), one would need a starting point grid search procedure. However, if the grid mesh is based on dividing the interval $[0, 1]$ into k points, then given the 16 variable problem at hand, the number of such search points is of the order of k^{15}. Thus this strategy is therefore unattractive even for the small example at hand. A genetic algorithm approach proves much more desirable.

Although we have included a mathematical programming solution, it is not our intent to compare GA approaches to mathematical programming ones, in general, for this example problem. The GAMS solution serves as a base of comparison and reasonableness for the GA solutions. It might also be useful to compare the GA approach with Simulated Annealing [11], Tabu Search [5] or other metaheuristics [12,22] for this example problem. This, however, is beyond the scope of the paper and presents a topic for future research.

4.1. Solution using genetic algorithms

Since the GA seeks solutions with high fitness, the following fitness-function, consistent with the objective of (26) is used:

$$f_j = \left(\sum_{i=1}^{9} e_i^2 \right)^{-2}, \tag{28}$$

where f_j is the fitness of the jth population member, and e_i, $i = 1, \ldots, 9$ are the errors as determined by (17)–(25). The second power in (28) provides for greater discrimination between solutions with close values of the total sum of squared errors (SSE).

Tables 4(a) and (b) present the SSE, the scenario probabilities and the probabilities of A, B, C, and D as obtained using genetic search with the two different crossover operators discussed earlier. Table 4(a) presents results using the crossover operator designed to process probability-interval

Table 3
GAMS solution

SSE	P(A)	P(B)	P(C)	P(D)	x_i, $i = 1, \ldots, 16$ (by row order)			
0.0347	0.498	0.402	0.873	0.118	0.0	0.172	0.0	0.103
					0.111	0.105	0.007	0.0
					0.79	0.037	0.0	0.011
					0.0	0.374	0.0	0.0

Table 4
Genetic algorithm solutions

SSE	P(A)	P(B)	P(C)	P(D)	$x_i,\ i = 1, \ldots, 16$ (by row order)			
(a) *PI-crossover*								
0.0199	0.499	0.398	0.772	0.1196	0.004	0.178	0.0	0.092
					0.031	0.117	0.0	0.077
					0.014	0.107	0.0	0.003
					0.071	0.251	0.0001	0.056
0.0198	0.498	0.399	0.769	0.127	0.005	0.141	0.0	0.127
					0.027	0.156	0.0	0.042
					0.014	0.093	0.0	0.019
					0.08	0.253	0.0	0.043
0.0199	0.496	0.403	0.773	0.121	0.0001	0.198	0.0	0.079
					0.030	0.105	0.0	0.084
					0.019	0.106	0.0003	0.0002
					0.072	0.243	0.0	0.064
(b) *Distribution-swap crossover*								
0.0203	0.497	0.410	0.753	0.120	0.02	0.0	0.19	0.087
					0.039	0.086	0.0	0.095
					0.014	0.07	0.0	0.05
					0.066	0.288	0.0	0.145
0.0221	0.495	0.410	0.671	0.123	0.022	0.0001	0.054	0.223
					0.048	0.167	0.0	0.002
					0.026	0.097	0.0002	0.009
					0.019	0.26	0.0295	0.065
0.0198	0.5	0.399	0.774	0.129	0.019	0.008	0.111	0.0
					0.156	0.027	0.189	0.0
					0.009	0.003	0.121	0.0
					0.09	0.223	0.0	0.062

forma, while Table 4(b) presents results using the distribution-swap operator. These results correspond to the following settings for the GA parameters: a population size of 100 was maintained for each simulated generation, and the best fitness member was retained intact in the next generation (elitist selection). The mutation rate, was set at 0.025 per string position and the crossover rate used was 0.7. The search was terminated after 500 generations. Genetic search, through selection and recombination, from one population to the next, was effected as follows:

```
Repeat
  Select parents f & g (fitness-proportionate selection)
  Generate uniform random Uc ∈ U[0, 1]
  If Uc <= PCROSS
    Perform crossover on f & g, to yield offspring p & q
  For both p & q
    Generate uniform random Um ∈ U[0, 1]
    If Um <= PMU
      Perform mutation on a randomly chosen position on p
  Insert p & q into new population.
Until new population is filled (N).
```

For comparison with a standard crossover operator not designed specifically for use with probability spaces, the regular arithmetic crossover [13] commonly used for real-valued representations is also considered. Here, given two parents $\mathbf{w}_1$ and $\mathbf{w}_2$, an offspring $\mathbf{w}_0$ is generated through a linear combination of the two parent strings: $\mathbf{w}_0 = \lambda\mathbf{w}_1 + (1 - \lambda)\mathbf{w}_2$, where λ is uniformly

Table 5
Comparing crossover performance

DS-crossover	PI-crossover	Arithmetic crossover
0.0199195	0.0199391	0.0221955
0.0202749	0.0199313	0.0221328
0.019935	0.0199559	0.0208151
0.0221673	0.0198573	0.0219981
0.0199466	0.0198127	0.0207584
0.0198259	0.0198924	0.0199947
0.0202723	0.0199279	0.0201446
0.0198796	0.0198675	0.0199219
0.0199395	0.0199095	0.0299570
0.0199763	0.0198698	0.0211992

chosen in $[0, 1]$. Since the resulting offspring may not satisfy the unit sum requirement of a probability distribution, they are subsequently re-normalized.

Tables 4(a) and (b) present three different solutions obtained with different random number streams. All the GA solutions are found to outperform the GAMS solution as expected. Table 5 compares the performance (SSE) of the two discussed probability-space crossover operators and the standard arithmetic crossover operator for 10 different random number streams. While the performance of PI-crossover (average $\mathrm{SSE} = 0.0199$) is marginally better than that of distribution-swap crossover (average $\mathrm{SSE} = 0.0202$), a paired comparison reveals a (one-tailed) p-value of 0.097. The standard arithmetic operator, as expected, shows poorer performance (average $\mathrm{SSE} = 0.0219$); paired comparison against the PI-crossover gives a (one-tailed) p-value of 0.029, and against the distribution-swap crossover operator gives a (one-tailed) p-value of 0.055. Table 6 shows the average number of generations, over the 10 independent runs, taken by the GA using the different crossover operators and with a population size of 100 – both the average generations to achieve within 1% of the best (minimum SSE) solution, and within 5% of the best solution are given. In both cases, the distribution-swap crossover is found to take significantly higher number of generations, with one-tailed p-values on a comparison of mean generations being 0.025 and 0.028 for the within-1% and within-5% cases, respectively. The best solution using the standard arithmetic crossover operator does not achieve a within-5% performance in 500 generations in 6 out of 10 runs. The CPU time for 500 generations with either crossover operator is of the order of 4 seconds on a 400 MHz Intel processor running Windows NT.

Table 6
Comparing average number of generations

	DS-crossover	PI-crossover	Arithmetic crossover
For within 1% of best	441	302.7	500+
For within 5% of best	230.7	125.1	500+

Fig. 1 examines the performance of the distribution-swap crossover (UDSX) for varying population sizes. With higher population sizes, the best solution is obtained in a fewer number of generations of search; the same is also observed for the PI crossover (Fig. 3). The interplay between population size and mutation rate is depicted in the graphs of Fig. 2(a)–(c), for the distribution-swap crossover. Higher mutation rates are seen to be necessary with smaller population sizes. Similar behavior is also noticed for the PI-crossover operator (Fig. 4(a)–(c)). We note here that the indicated mutation rates are for one entire string, and are not mutation probabilities for single string positions. Thus, for example, a considered mutation rate of 0.4 translates to a per position mutation rate of 0.025. Though slightly higher than conventionally considered mutation rates (of the order of 0.001), it is in agreement with theoretical findings that suggest higher mutation rates with real-valued representations [26,30].

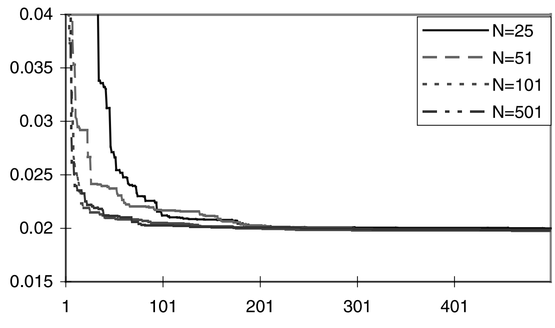

Fig. 1. Different population sizes (distribution-swap crossover).

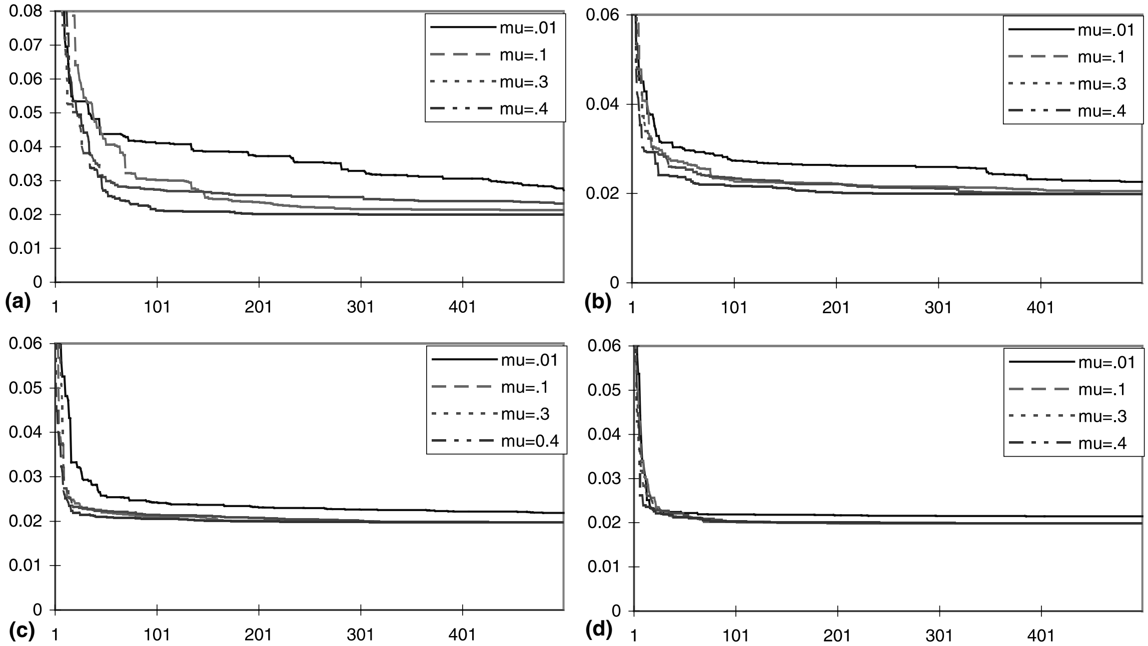

Fig. 2. (a) Comparing mutation with population size = 25 (distribution-swap crossover); (b) comparing mutation with population size = 51 (distribution-swap crossover); (c) comparing mutation with population size = 101 (distribution-swap crossover); (d) comparing mutation with population size = 501 (distribution-swap crossover).

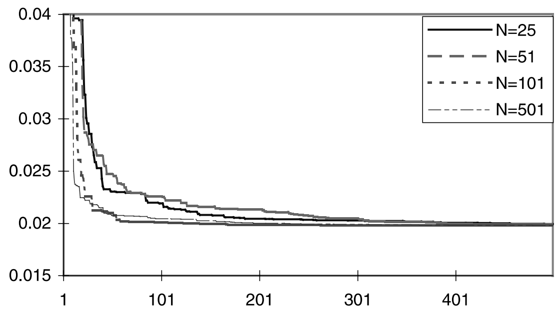

Fig. 3. Different population sizes (PI-crossover).

5. Conclusion

This paper considers the design of genetic algorithm (GA)-based procedures for search over discrete probability spaces. A test problem illustrates an application to the estimation of coherent probability assignments. More generally, these procedures apply to optimization over such spaces, or simplexes described by barycentric coordinates.

Two different crossover operators are proposed and compared. One is explicitly designed to possess the forma properties of respect and assortment, in consideration of GA design principles emanating from forma (extended schema) analysis [17]. Probability-interval formae are defined as potentially useful structures for GA manipulation in searching over probability spaces, and the operator designed for effective processing of such formae. The second crossover operator is an intuitive heuristic that forms offspring through an exchange of the front and tail ends of distributions. This second operator neither respects nor assorts probability-interval forma, but instead is shown to process what may be called probability-ratio forma. The operator assorts, but does not fully respect probability-ratio forma; however, respect holds to a degree which increases as convergence in the population advances.

These PI and distribution-swap crossover operators, designed specifically for searching probability spaces, are found to outperform the standard arithmetic crossover operator. Arithmetic crossover

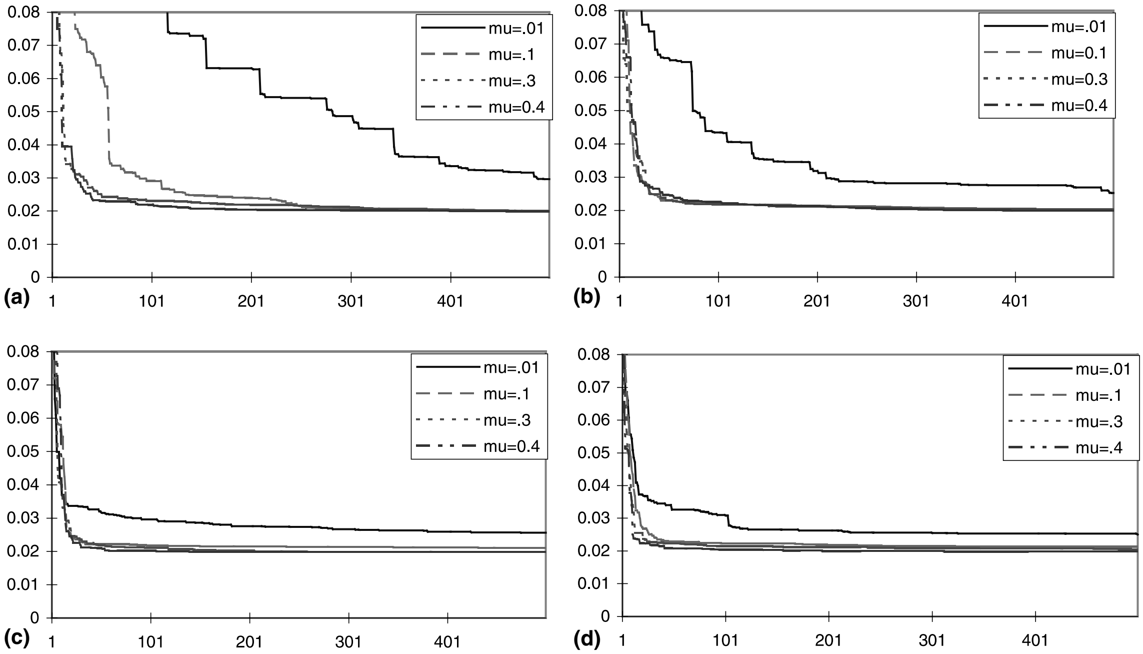

Fig. 4. (a) Comparing mutation with population size = 25 (PI-crossover); (b) comparing mutation with population size = 51 (PI-crossover); (c) comparing mutation with population size = 101 (PI-crossover); (d) comparing mutation with population size = 501 (PI-crossover).

reveals both a poorer overall performance, and takes a higher number of generations to reach comparable performance levels. This highlights the need for establishing and using operators that have been designed in explicit consideration of the search space involved. Forma theory provides valuable guidance in the design of such operators.

Both operators are found to perform significantly better than one conventional mathematical programming approach for the test problem. Experiments also reveal that the PI-crossover operator performs marginally better on solution quality, but does so in a significantly fewer number of generations than the second, intuitively designed distribution-swap crossover operator. These results can thus be viewed as confirming the utility of forma theory, and the design principles arising therefrom. The relative success of the distribution-swap operator also adds credence to the argument that in situations where the twin properties of respect and assortment are in conflict, forma assortment is more desirable, with respect gaining in relevance as the population converges. The distribution-swap crossover, analyzed as processing probability-ratio forma, possesses precisely these characteristics.

Acknowledgements

The authors would like to thank two anonymous reviewers for helpful comments in improving this paper.

References

[1] N.C. Dalkey, An elementary cross-impact model, Technological Forecasting and Social Change 3 (3) (1972) 341–352.
[2] D. Dasgupta, Z. Michalewicz, Evolutionary Algorithms in Engineering Applications, Springer, New York, 1997.
[3] J.C. Duperrin, M. Godet, SMIC 74 – A method for constructing and ranking scenarios, Futures 7 (4) (1975).
[4] L.J. Eshelman, R.A. Caruna, J.D. Schaffer, Biases in the crossover landscape, in: J.D. Schaffer (Ed.), Proceedings of

the Third International Conference on Genetic Algorithms, 1989, pp. 10–19.
[5] F. Glover, Tabu search: A tutorial, Interfaces 20 (4) (1986) 74–94.
[6] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[7] T.J. Gordon, H. Hayward, Initial experiment with the cross-impact matrix method of forecasting, Futures 1 (2) (1968).
[8] O. Helmer, Problems in futures research, Futures 9 (1) (1977) 2–31.
[9] R.M. Hogarth, Cognitive processes and the assessment of subjective probability distributions, Journal of the American Statistical Association 70 (1975) 271–294.
[10] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
[11] S. Kirpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[12] F.T. Lin, C.Y. Kao, C.C. Hsu, Applying the genetic approach to simulated annealing in solving some NP-hard problems, IEEE Transactions on Systems, Man, and Cybernetics 23 (6) (1993) 1752–1767.
[13] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolutionary Programs, second ed., Springer, Berlin, 1994.
[14] J.P. Martino, Technological forecasting for decision making, second ed., North-Holland, New York, 1983.
[15] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, 1996.
[16] H. Moskowitz, R.K. Sarin, Improving the consistency of conditional probability assessments for forecasting and decision-making, Management Science 29 (1983) 735–749.
[17] N.J. Radcliffe, The algebra of genetic algorithms, Annals of Mathematics and Artificial Intelligence 10 (1994) 339–384.
[18] N.J. Radcliffe, Genetic set recombination, in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms, 2, Morgan Kaufmann, Los Altos, CA, 1992.
[19] N.J. Radcliffe, Forma analysis and random respectful recombination, in: R. Belew, L. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991, pp. 221–229.
[20] N.J. Radcliffe, Equivalence class analysis of genetic algorithms, Complex Systems 5 (1991) 183–205.
[21] N.J. Radcliffe, P.D. Surry, Fitness variance of forma and performance prediction, in: M.D. Vose, L.D. Whitley (Eds.), Foundations of Genetic Algorithms, vol. III, Morgan Kaufmann, Los Altos, CA, 1994, pp. 51–72.
[22] V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, G.D. Smith (Eds.), Modern Heuristic Search Methods, Wiley, New York, 1996.
[23] C.S. Spetzler, C.S. Staël von Holstein, Probability encoding in decision analysis, Management Science 22 (1975) 340–358.
[24] P.D. Surry, N.J. Radcliffe, Fomal Algorithms + Formal Representations = Search Strategies, in: W. Ebeling, I. Rechenberg, H. Schwefel, H. Voigt (Eds.), Parallel Problem Solving from Nature, vol. IV, Springer, Berlin, 1996.
[25] G. Syswerda, Uniform crossover in genetic algorithms, in: J.D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1989, pp. 2–9.
[26] D.M. Tate, A.E. Smith, Expected allele coverage and the role of mutation in genetic algorithms, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1993, pp. 31–37.
[27] M.D. Troutt, T.B. Paine, A monotone variational method for optimal probability distributions, OR Spektrum 12 (1990) 201–206.
[28] A. Tversky, D. Kahneman, Causal schemas in judgments under uncertainty, in: M. Fishbein (Ed.), Progress in Social Psychology, Erlbaum, Hillsdale, NJ, 1982.
[29] A. Tversky, A.D. Kahneman, Extensional versus intuitive reasoning: The conjunction fallacy in probability judgment, Psychological Review 90 (1983) 293–315.
[30] A.H. Wright, Genetic algorithms for real parameter optimization, in: G.E. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991, pp. 205–218.