Computers Math. Applic. Vol. 25, No. 5, pp. 91–100, 1993
Printed in Great Britain.

0097-4943/93 $6.00 + 0.00

GENETIC SEARCH ALGORITHMS AND THEIR RANDOMIZED OPERATORS

S. ARUNKUMAR AND T. CHOCKALINGAM
Department of Computer Science and Engineering, Indian Institute of Technology
Bombay 400 076 India

(Received June 1992)

Abstract—Genetic algorithms are stochastic search procedures based on randomized operators such as crossover and mutation. Many of the earlier works that have attempted to analyse the behaviour of genetic algorithms have marginalized the role of these operators through gross abstractions. In this paper, we place in perspective these operators within the genetic search strategy. Certain important properties of these operators have been brought out through simple formalisms. The search behaviour of the genetic algorithm has been modeled in a Markovian framework and strong convergence proved. The limiting analysis of the algorithm reveals the vital interplay of the randomized operators.

1. INTRODUCTION

The class of NP-hard problems includes several important, practically encountered optimization problems, such as the problem of mapping tasks onto processors in the domain of parallel processing. The classification of such problems within the complexity hierarchy does not preclude the need for approximate algorithms and heuristics that can offer acceptable sub-optimal solutions. In the absence of good approximation algorithms and in situations wherein their existence is not evident, the heuristic approach is considered to be the only alternative.

2. HEURISTIC APPROACH

We broadly classify and define search heuristics as under [1,2]:

- *Deterministic heuristics*: This class of heuristics is characterised by their deterministic choice of search path. They normally adopt a fixed search strategy based on the available domain knowledge. Many local search heuristics are examples of this type of heuristics [3].
- *Randomized heuristics*: This category of heuristics employs choice operators in their search strategy that are randomized and does not depend on *a priori* complete knowledge on the features of the domain. Successive executions of these heuristics need not necessarily yield the same solution. Simulated annealing and genetic algorithms are examples of this class of heuristics.
- *Random start heuristics*: These heuristics are characterized by a randomly chosen initial solution which is then iteratively improved. Most of the iterative improvement heuristics falls under this [4,5] category.

In this paper, we are interested in the theoretical analysis of genetic algorithms from the randomized class. Genetic algorithms are stochastic search procedures based on randomized operators. These operators have been conceived through abstractions of natural genetic mechanisms such as crossover and mutation and have been cast into algorithmic forms. They evoked considerable interest among researchers ever since Holland gave the schema analysis in [6]. Genetic algorithms have since been successfully applied to solve a wide variety of hard combinatorial problems in many areas [1,2,7]. Although abundant literature is available on the application aspects of these algorithms, not much is found on their theoretical basis. This is in contrast with simulated

annealing algorithms, another member of the same class whose characteristics has been well analysed.

Earlier works that have attempted to analyse the behaviour of genetic algorithms have marginalized the role of the randomized operators (crossover and mutation) through gross abstractions. We place in perspective the characteristics and importance of these operators within the genetic search strategy. Certain important properties of these operators have been brought out through simple formalisms. The search behaviour of the genetic algorithm has been modeled in a Markovian framework and strong convergence proved. The limiting analysis of the algorithm based on its Markov model reveals the vital interplay of the randomized operators.

3. THE GENETIC ALGORITHM

The use of abstractions of genetic mechanisms like natural selection, crossover, and mutation in algorithmic forms for solving hard combinatorial optimization problems is well-known [5]. The search strategy adopted in genetic algorithms is simple and in the context of combinatorial optimization problems can be described as below.

The algorithm begins with an initial generation of a uniformly random population of solution patterns. By solution pattern we mean the syntactic encoding of the solution. It could be a string of binary digits or integers depending on the problem context. New generations are evolved from the current generation by applying idealized genetic operators like the crossover operator.

The crossover operator is effectively a syntactic pattern generator. Each application of the crossover operator involves a selection of pattern strings. The criterion for selection could be either fitness based or a random choice. In fitness based selection, the probability of a string being selected as a parent is proportional to its fitness. Once the parent strings are selected, two (uniformly) random positions (crossover points) are chosen in them. That portion of the string demarcated by the crossover points is swapped between the parent strings yielding two offspring strings. As an example, let us consider an integer encoding of the solutions. Let 12753 and 14395 be the chosen parent strings. If the crossover is between positions 2 and 4, the resulting offsprings are 14393 and 12755.

If crossover is the only operator used, then as new patterns evolve out of old ones, there is invariably a loss in pattern diversity. Pattern diversity in the population corresponds to the breadth of the searched domain. Loss of this diversity results in a premature convergence of the algorithm which is clearly undesirable. This is avoided by the introduction of a mutation operator which is applied to the offspring strings. Mutation operator introduces random variations in the patterns. There are a variety of mutation operators that can be defined. The simplest of all is the pointwise mutation in which a few positions in the string are chosen randomly and their values are replaced by randomly chosen ones.

The solutions thus obtained are evaluated for their fitness. The termination of the algorithm is by a predefined stopping rule. The best among the population is taken as the solution at termination.

3.1. The Algorithmic Description

```
begin{main}
    readinputs;
    initialize;
    while termination not true do
        create_new_generation
    end{while}
    output best among current generation;
end{main}
```

Function create_new_generation

```
begin{function}
    while new_generation size < Population_Size do
        begin{while}
            Select parent strings (random/fitness based);
            Apply crossover operator;
            Apply mutation operator;
            Evaluate offsprings for fitness;
            Add offsprings to new_generation as determined by
            the acceptance policy;
    end{while}
end{function}
```

3.2. Generalized Crossover and Mutation Operators

Crossover is essentially a combinational operator. It generates offspring string patterns from a chosen parent pair of string patterns as follows. The crossover operator described in Section 1, is the single segment crossover in which crossing over takes place between two chosen positions along the string. Single segment crossover is the restricted form of a more generalized crossover operator in which multisegment crossing over is permitted. In the analysis to follow, we consider the generalized crossover operator instead of its constrained single segment form.

Mutation operator is used to introduce random variations in the string patterns of a population. There are several possible mutation operator definitions. Pointwise mutation is one of them. It consists of replacing string alphabets at randomly chosen positions over the string length with alphabets chosen randomly from the complete alphabet set.

4. SURVEY OF RELATED LITERATURE

Literature on the theoretical aspects of genetic algorithms is not as abundant as it is on their practical aspects. Holland in [6] gave the famous schema theorem and the notion of intrinsic parallelism which spurred of some interest in schema based analysis of the genetic algorithms. However Holland's analysis was not focussed on the randomized operators and his analysis views these operators in terms of their disruption probabilities. Subsequently, Bridges and Goldberg [8] gave a more detailed analysis on the reproduction and crossover operators of the genetic algorithm. The expected proportion of a given string in generation $(t + 1)$ under both reproduction and crossover has been derived in their work. The expression is however cumbersome and does not bring out clearly the properties of the crossover operator. Essentially the formalism adopted there reflects quantitative quest and hence, does not help in making qualitative conclusions about the crossover operator. Moreover mutation has not been considered.

Goldberg and Segrest in [9] have done a Markov analysis of the genetic algorithm. In this analysis, the state space consists of states defined by the number of ones in a population of N ones and zeros. The convergence and divergence characteristics of the algorithm in terms of the proportion of ones and zeros in a population have been obtained. Though this study throws some light on certain aspects of the genetic algorithm, the exact roles of the randomized operators of crossover and mutation have not been made clear.

5. PRELIMINARIES

We consider the domain of binary strings. In order to simplify the analysis, we partition the binary string space under consideration into non-overlapping subsets by adopting the schema abstraction given by Holland [6] as the basis. Crossover and mutation are in fact string operators. However, in our analysis we consider the effect of these operators on schema abstractions instead of at the atomic level of strings. This does not affect the generality of the results as the schema

abstractions have a nice hiearchical structure that can be made use of. The following definitions are necessary for the analysis.

Let $\Sigma = \{*, 0, 1\}$ and $\Sigma' = \{0, 1\}$ be some string alphabets. Let Σ_l and $\Sigma_l{}'$ be the set of all possible strings of length l over Σ and Σ_l, respectively. Let $X \subseteq \Sigma_l'$ and $S \in \Sigma_l$. For any string m, m^i corresponds to the i^{th} position from the right, with 0 as the least significant position.

DEFINITION 1.

$$S = \{x \mid x \in \Sigma_l, \forall i, S^i \neq * \Rightarrow x^i = S^i\}$$

is a schema. Thus schema S is an element of Σ_l which represents a membership condition over Σ_l'.

EXAMPLE 1. For example the schema 000** represents a set of all strings of length 5 whose first three positions are 0's, i.e., $\{00000, 00001, 00010, 00011\}$.

DEFINITION 2. *Two schemas S_a and S_b are non-overlapping if and only if the string subsets defined by them are disjoint, i.e., $S_a \cap S_b = \emptyset$. They are overlapping otherwise.*

EXAMPLE 2. Schemas 00** and 01** are non-overlapping.

DEFINITION 3. *Two schemas S_a and S_b are positionally equivalent if and only if*

$$S_a{}^i = * \iff S_b{}^i = *, \qquad 0 \leq i \leq l.$$

EXAMPLE 3. Schemas 0*1* and 1*0* are positionally equivalent. Schemas 00** and 1*0* are not positionally equivalent.

DEFINITION 4. *Let Ψ be a set of non-overlapping and positionally equivalent schemas. Let the alphabet set $\{0,^*\}$ be mapped to integer 0, and alphabet 1 be mapped to integer 1. The schema family function $f : \Psi \rightarrow I$ is defined as*

$$f(S) = \sum_{i=0}^{l} S^i 2^i.$$

EXAMPLE 4. $f(011^{**}) = 3$. We identify a schema belonging to a set of positionally equivalent and non-overlapping schemas by its family identity. Thus 011** corresponds to the schema S_3.

DEFINITION 5. *Let Ψ be a set of non-overlapping and positionally equivalent schemas. Let the alphabet set $\{0,^*\}$ be mapped to integer 0, and alphabet 1 be mapped to integer 1. The schema constituent function $g : \Psi \rightarrow I$ is defined as*

$$g(S) = \sum_{i=0}^{l} S^i.$$

EXAMPLE 5. $g(011^{**}) = 2$.

DEFINITION 6. *Let $S_a, S_b \in \Psi$ Let the alphabet set $\{0,^*\}$ be mapped to integer 0, and alphabet 1 be mapped to integer 1. Let $* \oplus * = *$ where $\oplus$ is the usual XOR operator. $d : \Psi \times \Psi \rightarrow I$, is a schema distance function and is defined as*

$$d(S_a, S_b) = \sum_{i=0}^{l} (S_a^i \oplus S_b^i).2^i$$

EXAMPLE 6. $d(001^{**}, 110^{**}) = 28$.

DEFINITION 7. *For any $S \in \Sigma_l$, let $\Pi = \{0, 1, 2, \ldots, l-1\}$ be the set of position indices over S. $\Delta = \{(i,j) \mid (i,j) \in \Pi \times \Pi, i \leq j\}$ is the set of all possible crossover segments.*

DEFINITION 8. *Let $Z \subset \Delta i$ and $S_a, S_b, S_{a'}, S_{b'} \in \Psi$. The generalized crossover operator, $C : \Psi \times \Psi \times Z \rightarrow \Psi \times \Psi$, over Ψ, is defined as*

$$C(S_a, S_b, Z) = \{S_{a'}, S_{b'}\},$$

where $\forall i$,

$$\left.\begin{aligned} S^i_{a'} &= S^i_b \\ S^i_{b'} &= S^i_a \end{aligned}\right\} \quad \text{iff } l \le i \le m, \quad (l,m) \in Z$$

$$\left.\begin{aligned} S^i_{a'} &= S^i_a \\ S^i_{b'} &= S^i_b \end{aligned}\right\} \quad \text{otherwise.}$$

EXAMPLE 7. Let $S_a = 0010011^{****}$ and $S_b = 1000110^{****}$. Let $Z = \{(4,5),(7,9)\}$. $C(S_a, S_b, Z) = \{0000010^{****}, 1010111^{****}\}$.

DEFINITION 9. *For any* $S_a, S_b \in \Psi$, $\bigcirc_{ab}$ *is the set of all possible schemas through all possible generalized crossover of schemas* S_a *and* S_b.

EXAMPLE 8.

$$\begin{aligned} S_a &= 00** \\ S_b &= 11** \\ \Pi &= \{0,1,2,3\} \\ \Delta &= \{(0,0),(0,1),(0,2),0,3),(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\} \\ \bigcirc_{ab} &= \{00**, 01**, 10**, 11**\}. \end{aligned}$$

DEFINITION 10. *Let* $\Upsilon = \mathcal{P}(\Pi)$, *the power set of* Π. *The mutation operator* $M : \Psi \times \Upsilon \to \Psi$ *over* Ψ *is defined as*

$$M(S_a, X) = S_{a'},$$

where

$$\begin{aligned} S^i_{a'} &= \overline{S^i_a}, && \text{iff } i \in X, \quad X \in \Upsilon, \quad S^i_a \neq * \\ S^i_{a'} &= S^i_a, && \text{otherwise.} \end{aligned}$$

EXAMPLE 9.

$$\begin{aligned} S_a &= 0010011**** \\ X &= \{5,7,9\} \\ M(S_a, X) &= \{0111001****\}. \end{aligned}$$

6. THE CONSERVATIVE CROSSOVER

LEMMA 1. *If* $C(S_a, S_b, Z) = \{S_{a'}, S_{b'}\}$ *for some* $Z \subset \Delta$, *then*

$$\begin{aligned} &(i) && f(S_a) + f(S_b) = f(S_{a'}) + f(S_{b'}), \\ &(ii) && g(S_a) + g(S_b) = g(S_{a'}) + g(S_{b'}), \\ &(iii) && h(S_a, S_b) = h(S_{a'}, S_{b'}), \end{aligned}$$

PROOF.

(i) From the definition of the mapping function f, the equation in (i) above becomes

$$\sum_{i=0}^{l} S^i_a 2^i + \sum_{i=0}^{l} S^i_b 2^i = \sum_{i=0}^{l} S^i_{a'} 2^i + \sum_{i=0}^{l} S^i_{b'} 2^i,$$

which is same as

$$\sum_{i=0}^{l} (S^i_a + S^i_b) 2^i = \sum_{i=0}^{l} (S^i_{a'} + S^i_{b'}) 2^i.$$

From the definition of the crossover operator, which involves only an exchange of substring, it is clear that for any position i, $S^i_a + S^i_b = S^i_{a'} + S^i_{b'}$.

(ii) The second statement is an invariance property which says that the total number of ones (zeros) in the parent and offspring strings remains unchanged by crossover. From the definition of g, the second statement of the lemma becomes,

$$\sum_{i=0}^{l} S_a^i + \sum_{i=0}^{l} S_b^i = \sum_{i=0}^{l} S_{a'}^i + \sum_{i=0}^{l} S_{b'}^i, \quad \sum_{i=0}^{l} (S_a^i + S_b^i) = \sum_{i=0}^{l} (S_{a'}^i + S_{b'}^i).$$

Following the same argument as in (i), $S_a^i + S_b^i = S_{a'}^i + S_{b'}^i$.

(iii) The third statement of the lemma says that the hamming distance (the number of differing bit positions) between the parent schema pair and the offspring schema pair should be the same. From the definition of crossover operator, $S_a^i \oplus S_b^i = S_{a'}^i \oplus S_{b'}^i$ for all the positions and hence

$$\sum_{i=0}^{l} (S_a^i \oplus S_b^i).1 = \sum_{i=0}^{l} (S_{a'}^i \oplus S_{b'}^i).1,$$

where $\oplus$ is the XOR operator. ∎

LEMMA 2.

$$S_c \in \bigcirc(S_a, S_b) \Longleftrightarrow d(S_a, S_b) = d(S_c, S_{a+b-c}).$$

PROOF. Lemma 2 gives a necessary and sufficient condition for any schema to be a possible offspring of a crossover of given pair of schemes.

(i) To prove that $S_c \in \bigcirc(S_a, S_b) \Longrightarrow d(S_a, S_b) = d(S_c, S_{a+b-c})$ is straight forward from Lemma 1. If S_c is an offspring of the crossover between S_a and S_b, then by the first statement of Lemma 1, the other partner of S_c is S_{a+b-c}. By the definition of the function d,

$$d(S_a, S_b) = \sum_{i=0}^{l} (S_a^i \oplus S_b^i).2^i, \quad d(S_c, S_{a+b-c}) = \sum_{i=0}^{l} (S_c^i \oplus S_{a+b-c}^i).2^i.$$

For any i, $S_a^i \oplus S_b^i = S_c^i \oplus S_{a+b-c}$, since crossover at any bit position involves only an exchange of the bits in that position.

(ii) To prove that for any S_a, S_b and S_c, $d(S_a, S_b) = d(S_c, S_{a+b-c}) \Longrightarrow S_c \in \bigcirc(S_a, S_b)$ we proceed as follows. Assuming the above statement, S_c and S_{a+b-c} are posssible offsprings of S_a and S_b only if there exists possible crossover points along the length of the parent schemas such that S_a and S_b are transformable through the exchange of bit positions at those points to S_c and S_{a+b-c}. With the generalized crossover operator, it is possible to have arbitrary number of crossover segments and hence, the possibility of the crossover points for the required transformation occuring exists. ∎

7. THE EXPLORATIVE MUTATION

LEMMA 3.

$$p_{ij} = p[(K \mid M(S_i, K) = S_j)] > 0,$$

where $S_i, S_j \in \Psi, K \in \Upsilon$.

PROOF. The pointwise mutation operator is realized in two steps. In the first step a random choice of the total number of positions to be mutated is made. In the second step, the actual positions to be mutated are chosen. Let $h(S_i, S_j) = x$ and $x > 0$ indicating that $S_i \neq S_j$. Let the positions in which S_i and S_j differ be indexed 1 through x from right to left.

The probability, p_{ij} of S_i transforming to S_j can be expressed as the probability of the pointwise mutation operator choosing $K \in \Upsilon$ such that $M(S_i, K) = S_j$. It is clear that $|K| \geq x$. From the two step realization of the mutation operator,

$$p[(K \mid M(S_i, K = S_j)] = p[(K \mid M(S_i, K) = S_j) \mid (X = m \mid m \geq x)] \times p[(X = m \mid m \geq x)],$$

where $p[(K \mid M(S_i, K) = S_j) \mid (X = m \mid m \geq x)]$ is the conditional probability of choosing a subset K as the set of mutation positions given that, by the first step of mutation operation, the random variable X which denotes the number of positions to be mutated takes a value m which is greater than x, the hamming distance between the source schema S_i and the target schema S_j. $p[(X = m \mid m \geq x)]$ is the probability of choosing m positions for mutation such that $m \geq x$. With the values of X being drawn from an uniform distribution in the range 0 to l (l is the length of the schemas),

$$p[(X = m \mid m \geq x)] = 1 - \frac{x}{1+l}.$$

The value m of the random variable X can now be treated as the number of mutation trials. The probability of the events of positions 1 through x of S_i differing from S_j being selected for mutation exactly once and other positions being selected for the remaining $m - x$ mutation trials can be given by the multinomial distribution,

$$p[(K \mid M(S_i, K) = S_j) \mid (X = m \mid m \geq x)] = \sum_{\forall m, x \leq m \leq l} \frac{m!}{(m-x)!} p_1 p_2 \cdots p_x p_{(m-x)}^{(m-x)},$$

where p_1 to p_x denotes the probability of selection of the differing positions, and p_{m-x} is the probability of selection of positions other than the x positions of interest.

The selection of events are mutually independent and are equally likely. Thus,

$$p_i = \frac{1}{l} \quad \text{and} \quad p_{m-x} = 1 - \frac{x}{l}, \qquad \forall i, 1 \leq i \leq x.$$

From the above expressions,

$$p_{ij} = \sum_{\forall m, x \leq m \leq l} \frac{m!}{(m-x)!} \frac{1}{l^x} \left(\frac{l-x}{l}\right)^{k-x} \times \left(1 - \frac{x}{1+l}\right)$$

and $p_{ij} > 0$. ∎

8. MARKOV CHAIN ANALYSIS OF GENETIC ALGORITHM

In this section, we provide a Markov analysis of the genetic algorithm. The main aim of this analysis is to elucidate the roles of crossover and mutation operators in the genetic based search process. Particularly, the important role played by the mutation operator in bringing out the algorithm from what could have been absorbing states without this operator.

The genetic algorithm's behaviour can be modeled as a Markov process. The creation of a new population of solutions depends solely on the current population. Thus, the conditional probability of the search process to reach a particular population state from a given population state at any particular instant is unaffected by knowledge about previous transitions. The process is history insensitive and hence satisfies the Markovian criterion.

8.1. The State Space

The algorithmic states considered for the Markov analysis are predicates over schema representation of a population. Let the entire domain of binary strings of length m be grouped under non-overlapping schemas of length l. Thus each schema represents a subset containing 2^{m-l} strings. Each schema type is uniquely identified by its decimal value. Let $\mathcal{F} = \{0, 1, \ldots, 2^l - 1\}$ be the entire set of schema types. Let $\mathcal{F}_{P_i} \subseteq \mathcal{F}$ be the schema type characterisation of population P_i. The membership of a schema type k in $\mathcal{F}_{P_i}$ represents the presence of one or more strings from schema k in population P_i.

EXAMPLE 10. Let $\mathcal{F} = \{0, 1, 2, 3\}$ be the complete set of schemas of length 2. If $P_i = \{0010, 0101, 1000, 0011\}$, then $\mathcal{F}_{P_i} = \{0, 1, 2\}$ indicating the presence of one or more strings from the schema types 0, 1 and 2 in the population.

DEFINITION 11. *$\mathcal{S}$ is the Markovian state space, defined as the power set of $\mathcal{F}$ excluding the null set.*

$$\mathcal{S} = \mathcal{P}(\mathcal{F}) - \Phi.$$

The set $\mathcal{S}$ represents all possible population types which are characterised by the different schema types.

EXAMPLE 11. Let

$$\begin{aligned}\mathcal{F} &= \{0,1,2,3\},\\ \mathcal{S} &= \{\{0\},\{1\},\{2\},\{3\},\{0,1\},\{0,2\},\{0,3\},\{1,2\},\{1,3\},\{2,3\},\\ &\quad \{0,1,2\},\{0,1,3\},\{0,2,3\},\{1,2,3\},\{0,1,2,3\}\}.\end{aligned}$$

There are 4C1 single schema states, 4C2 two schema states, 4C3 three schema states and 4C4 four schema states. It is to be noted here that although $\{0,1\} \subset \{0,1,2\}$ the states represented by these sets are considered to be distinct.

Semantics of the States

DEFINITION 12. *Θ_i is an aggregate state indicating the representativeness of the poopulation in terms of the schema types and consists of the set of all Markov states with i number of schemas.*

EXAMPLE 12. From the previous example,

$$\begin{aligned}\Theta_1 &= \{\{0\},\{1\},\{3\}\};\\ \Theta_2 &= \{\{0,1\},\{0,2\},\{0,3\},\{1,2\},\{1,3\},\{2,3\}\};\\ \Theta_3 &= \{\{0,1,2\},\{0,1,3\},\{0,2,3\},\{1,2,3\}\};\\ \Theta_4 &= \{\{0,1,2,3\}\}.\end{aligned}$$

The aggregate state Θ_i indicates the representativeness of the population in terms of the schema types. Now $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_i, \Theta_n\}$ be the set of all such aggregate states.

DEFINITION 13. *$\mathcal{R} : \Theta \to I$ is a function which expresses the representativeness of the aggregate states contained in Θ, and is defined as*

$$\mathcal{R}(\Theta_i) = i.$$

The aggregate states of Θ can be ordered on the basis of their degree of representativeness. Thus, $\mathcal{R}(\Theta_i) < \mathcal{R}(\Theta_j)$ if $i < j$. If there are n such aggregate states, then in general

$$\mathcal{R}(\Theta_1) < \mathcal{R}(\Theta_2) < \cdots < \mathcal{R}(\Theta_i) < \mathcal{R}(\Theta_n).$$

Transitions

Transition from state s_i to state s_j in one step is possible if the probability $p(s_i \to s_j) > 0$. A step here connotes the generation of a new population from the current population by finite number of applications of randomized operators.

8.2 Crossover and Absorption

THEOREM 1. *Let X be the initial state of the population. Let crossover be the only randomised operator.*

(1) *If $X \notin \Theta_1$, then $\mathcal{A}$ is the set of absorbing states, where*

$$\mathcal{A} = \{s \mid s \in \Theta_2, d(s) = k, k = 2^m\}.$$

(2) *If $X \in \Theta_1$, then $\mathcal{A} = \Theta_1$ is the set of absorbing states.*

PROOF. If $d(S_a, S_b)$ is some power of 2, then S_a and S_b differ in one bit position. When parent schemas differ in just one bit, crossover yields the same pair as the offsprings. This is quite obvious from the definition of the crossover operator. If the initial population state is of single schema type, then the algorithm does not exit from state Θ_1, for in such a case crossover yields the same schema type. ∎

8.3. Mutation and Communication

THEOREM 2. *With the application of mutation, all the states communicate.*

PROOF. Crossover generates a new population from the current one. Mutation operator is applied independently over all the elements of the population generated by crossing over. Let the probability of transition from state s_a to s_b be given by $p(s_a \rightarrow s_b)$. The statement of the theorem asserts that $p(s_a \rightarrow s_b) > 0$. Let $s_a \in \Theta_m$ and $s_b \in \Theta_n$. Since $R(\Theta_m) = m$ and $R(\Theta_n) = n$, the number of distinct schema types in s_a and s_b would be m and n, respectively. Let the population be of a finite size say $2N$. This means that there are $2N$ mutation trials in a population generation. Let $s_a = \{a_1, a_2, \ldots, a_m\}$ and $s_b = \{b_1, b_2, \ldots, b_n\}$. Let $p_{a_i b_j}$ represent the probability of transition from schema a_i to schema b_j in a single mutation step. Let $n_{a_i b_j}$ be the number of mutation trials in which there is a transition from schema a_i to b_j. From the multinomial distribution,

$$p(s_a \rightarrow s_b) = \sum_{\left\{\forall n_{a_i b_j} \mid \sum_{\substack{1 \le i \le m \\ 1 \le j \le n}} n_{a_i b_j} = 2N\right\}} \left(\frac{2N!}{\prod n_{a_i b_j}!}\right) \prod p_{a_i b_j}^{(n_{a_i b_j})}.$$

Clearly $p(s_a \rightarrow s_b) > 0$. ∎

EXAMPLE 13. Let s_a be represented by an instance $\{a\}$ where $\{a\} \in \Theta_1$ and s_b as $\{b, c\}$ where $\{b, c\} \in \Theta_2$. Let the population be of a finite size $2N$.

$$p(\{a\} \rightarrow \{b, c\}) = \sum_{\{\forall n_1, n_2 \mid n_1 + n_2 = 2N\}} \frac{2N!}{n_1!\, n_2!} p_{ab}^{n_1} p_{ac}^{n_2}$$

8.4. Limiting Behaviour

THEOREM 6. *If a genetic algorithm with crossover and mutation as randomized operators, saves the best solution across generations, then for a global optimization problem, as the number of generations tends to infinity, the algorithm would hold the globally optimal solution with probability one.*

PROOF. If we consider a schema to constitute a set of subschemas (schema within a schema) the Markov model and the properties of crossover and mutation operators are preserved at the subschema level. This allows us to view a binary solution string within a hierarchy of schemas. From Lemma 3, it is clear that all the algorithmic states are communicating and hence, are persistent. The same argument can be extended to the subschema levels. The domain of binary strings of a fixed length is a finite set. From the above arguments, it is clear that the algorithm is inherently capable of generating all possible solution strings and hence, the theorem. ∎

8.5. Scope of the Analysis

In this analysis, it is assumed that for crossover a uniformly random selection of parent strings are made from the current population. In practical genetic algorithms, parent strings are selected on the basis of some fitness criterion. Fitness based parent selection does not change the limiting behaviour. However, finite time behaviour is bound to be influenced considerably by such a selection criterion. The main emphasis of the analysis has been to demonstrate the roles played by crossover and mutation as randomized operators influencing the strongly convergent limiting behaviour of the algorithm. The analysis of limiting behaviour shows the capability of the algorithm towards sustained search of the domain and hence, its suitability in a global optimization context. In this sense, genetic algorithms are highly suited for application in global optimization problems. Issues regarding the optimal choice of the number of generations to be evolved and the size of the population, requires an analysis of finite time behaviour.

9. CONCLUSION

In this paper, we have identified certain properties of the randomized operators in genetic algorithms. The algorithm's behaviour has been modeled in a Markovian framework. A limiting behaviour analysis of genetic algorithms with crossover and mutation as randomized operators based on their properties reveals the interplay of these operators in establishing a strong convergence property of the algorithm. Further work is necessary on the finite time behaviour analysis of the genetic algorithm.

REFERENCES

1. S. Arunkumar and T. Chockalingam, Randomized heuristics for the mapping problem, *Intl. Jnl. of High Speed Computing* (to appear).
2. T. Chockalingam and S. Arunkumar, A randomized heuristics for the mapping problem: The genetic approach, In *Parallel Computing*, North-Holland (to appear).
3. B.W. Kernighan and S. Lin, An Efficient heuristic procedure for partitioning graphs, *Bell Systems Technical J.* **49** (2), 291–308 (1970).
4. P. Sadayappan and F. Ercal, Cluster partitioning approaches to mapping parallel programs onto a hypercube, In *I Intl. Conf. on Supercomputing, Athens, (June 1987)*, pp. 475–497.
5. S.Y. Lee and J.K. Aggarwal, A mapping strategy for parallel processing, *IEEE Trans. on Computers* **C-36** (4), 433–442 (April 1987).
6. J.H. Holland, Genetic algorithms and the optimal allocation of trials, *SIAM J. Computing* **2** (2) (June 1973).
7. B. Goldberg, *Genetic Algorithms*, Addisson-Wesley, U.S.A., (1988).
8. C.L. Bridges and D.E. Goldberg, An analysis of reproduction and crossover in a binary coded genetic algorithm, In *Proc. of the Second Intl. Conf. on Genetic Algorithms and their Applications*, pp. 9–13, (1987).
9. D.E. Goldberg and P. Segrest, Finite Markov chain analysis of genetic algorithms, In *Proc. of the Second Intl. Conf. on Genetic Algorithms and their Applications*, pp. 1–8, (1987).
10. K. Efe, Heuristic models of task assignment scheduling in distributed systems, *Computer* **15**, 50–56 (June 1982).
11. D.A. Plaisted, A heuristic algorithm for small separators in arbitrary graphs, *SIAM J. Computing* **19** (2), 267–280 (April 1990).