SELF ADAPTATION IN EVOLUTIONARY ALGORITHMS

JAMES EDWARD SMITH

A thesis submitted in partial fulfilment of the requirements of the

University of the West of England, Bristol

for the degree of Doctor of Philosophy

Faculty of Computer Studies and Mathematics

University of the West of England, Bristol

July 1998

Abstract

Evolutionary Algorithms are search algorithms based on the Darwinian metaphor of "Natural Selection". Typically these algorithms maintain a population of individual solutions, each of which has a fitness attached to it, which in some way reflects the quality of the solution. The search proceeds via the iterative generation, evaluation and possible incorporation of new individuals based on the current population, using a number of parameterised genetic operators. In this thesis the phenomenon of Self Adaptation of the genetic operators is investigated.

A new framework for classifying adaptive algorithms is proposed, based on the scope of the adaptation, and on the nature of the transition function guiding the search through the space of possible configurations of the algorithm.

Mechanisms are investigated for achieving the self adaptation of recombination and mutation operators within a genetic algorithm, and means of combining them are investigated. These are shown to produce significantly better results than any of the combinations of fixed operators tested, across a range of problem types. These new operators reduce the need for the designer of an algorithm to select appropriate choices of operators and parameters, thus aiding the implementation of genetic algorithms.

The nature of the evolving search strategies are investigated and explained in terms of the known properties of the landscapes used, and it is suggested how observations of evolving strategies on unknown landscapes may be used to categorise them, and guide further changes in other facets of the genetic algorithm.

This work provides a contribution towards the study of adaptation in Evolutionary Algorithms, and towards the design of robust search algorithms for "real world" problems.

Self-Adaptation in Evolutionary Algorithms

Introduction

Evolutionary Algorithms are search algorithms based on the Darwinian metaphor of "Natural Selection". Typically these algorithms maintain a finite memory, or "population" of individual solutions (points on the search landscape), each of which has a scalar value, or "fitness" attached to it, which in some way reflects the quality of the solution. The search proceeds via the iterative generation, evaluation and possible incorporation of new individuals based on the current population.

A number of classes of Evolutionary Algorithms can (broadly speaking) be distinguished by the nature of the alphabet used to represent the search space and the specialisation of the various operators used, e.g. Genetic Algorithms ([Holland, 1975]: binary or finite discrete representations), Evolution Strategies ([Rechenberg, 1973, Schwefel, 1981]: real numbers) Evolutionary Programming ([Fogel et al., 1966]: real numbers), and Genetic Programming ([Cramer, 1985, Koza, 1989]: tree based representation of computer programs). This thesis is primarily concerned with Genetic Algorithms (GAs) based on a binary problem representation, since these can be used to represent a wide range of problems (ultimately, any problem that can be represented in a computer).

Although not originally designed for function optimisation, Genetic Algorithms have been shown to demonstrate an impressive ability to locate optima in large, complex and noisy search spaces. One claim frequently made is that the GA is a robust algorithm, i.e. that is it is relatively insensitive to the presence of noise in the evaluation signal. However it is recognised that the performance is greatly affected by the choice of representation, size of population, selection method and genetic operators (i.e. the particular forms of recombination and mutation and the rate at which they are applied). It is easily shown that poor choices can lead to reduced performance, and there is a further complication in the fact that both empirical (e.g. [Fogarty, 1989]) and theoretical (e.g.[Back, 1992a]) studies have shown that the optimal choice is dependant not only on the nature of the landscape being searched but of the state of the search itself relative to that landscape.

In this thesis a number of mechanisms are proposed and investigated within which the parameters governing search are encoded with the individuals and are able to adapt subject to the same evolutionary pressures and mechanisms as the candidate solutions. This is known as Self Adaptation of the search strategies.

These mechanisms allow the problem and parameter spaces to be searched in parallel, and by the choice of a suitable representation the range of strategies accessible is extended beyond those available to "standard" genetic algorithms.

Using the metric of the best value found in a given number of evaluations, it is shown that these algorithms exhibit superior performance on a variety of problem types when compared to a range of "standard" GAs. Furthermore, analysis of the reproduction strategies which evolve in the adaptive algorithms shows that the behaviours displayed correspond well to what is known about the nature of the search landscapes. Finally methods are suggested for utilising this correspondence to obtain on-line estimates of the nature of an unknown landscape, and to apply them to facets such as population sizing.

Organisation

This thesis is organised as follows:

In Chapter One the basic genetic algorithm is introduced and the various operators described. A review of previous work on optimisation of control parameters is given, followed by a discussion of a number of schemes that have been proposed for adapting operators and parameters during the course of evolution. A framework is presented within which these adaptive algorithms can be classified.

In Chapter Two previous work on recombination in genetic algorithms is discussed, and the concept of *linkage* between genes is introduced. A framework is developed for representing the linkage between genes, within which can be described various operators and the way in which they preserve information on linkage.

Building on this, in Chapter Three a Self-Adaptive mechanism is introduced based on the principles developed. Performance comparisons against a range of widely used crossover operators are presented, and the effect of changing other search parameters is investigated.

In Chapter Four, Kauffman's NK-family of search landscapes are introduced [Kauffman, 1993]. These are a set of tunable landscapes with well known statistical characteristics. An analysis is then given of the recombination strategies which evolve on different types of landscapes, in the light of their known properties. Results are also presented from performance comparisons with other recombination operators. Finally it is suggested that in addition to being effective in an optimisation setting, observations of the strategies evolving may be used to characterise the landscape being searched.

In Chapter Five the focus is turned to mutation, and the application of some ideas from Evolutionary Strategies to Steady State GAs is investigated. The effect of changing other search parameters and incorporating an element of local search is investigated and the performance compared with a variety of widely used and recommended fixed mutation rates. An analysis is made of the mutation rates evolved on different classes of landscapes, and with differing amounts of local search. Following this reasons are given for the empirically observed superior performance of the local search mechanism.

In Chapter Six two possible ways of merging the methods described above are considered. Both of these produce a single reproduction operator within which both the units of heredity (which determine recombination strategy) and the mutation rate(s) applied are allowed to evolve. Performance comparisons and a range of analysis tools are used to investigate the synergistic effects of co-evolving the recombination and mutation strategies.

Finally, in Chapter Seven, conclusions and a number of suggestions for future work are presented.

Some of the work contained in this thesis has previously been published elsewhere, specifically:
parts of Chapter One have been published in [Smith and Fogarty, 1997a],
parts of Chapters Two and Three have been published in [Smith and Fogarty, 1995],
parts of Chapter Four have been published as [Smith and Fogarty, 1996b],
parts of Chapter Five have been published as [Smith and Fogarty, 1996c] and
parts of Chapter Six are based on work published as [Smith and Fogarty 1996a].

Acknowledgements

I would like to thank my supervisor, Professor Terry Fogarty for his support during this work, and the staff of the ICSC for many fruitful and stimulating discussions.

Chapter One
Operator and Parameter Adaptation in Genetic Algorithms

1. A Background to Genetic Algorithms

Genetic Algorithms [Holland, 1975] are a class of population based randomised search techniques which are increasingly widely used in a number of practical applications. Typically these algorithms maintain a number of potential solutions to the problem being tackled, which can be seen as a form of working memory - this is known as the population. Iteratively new points in the search space are generated for evaluation and are optionally incorporated into the population.

Attached to each point in the search space will be a unique fitness value, and so the space can usefully be envisaged as a "fitness landscape". It is the population which provides the algorithm with a means of defining a non-uniform probability distribution function (p.d.f.) governing the generation of new points on the landscape. This p.d.f. reflects possible interactions between points in the population, arising from the "recombination" of partial solutions from two (or more) members of the population (parents). This contrasts with the globally uniform distribution of blind random search, or the locally uniform distribution used by many other stochastic algorithms such as simulated annealing and various hill-climbing algorithms.

The genetic search may be seen as the iterated application of two processes. Firstly *Generating* a new set of candidate points, the *offspring*. This is done probabalistically according to the p.d.f. defined by the action of the chosen reproductive operators (recombination and mutation) on the original population, the *parents*. Secondly *Updating* the population. This usually done by evaluating each new point, then applying a selection algorithm to the union of the offspring and the parents.

1.1. Some Definitions

The Genetic Algorithm can be formalised as:

$$GA = (P^0, \delta^0, \lambda, \mu, l, F, G, U) \qquad \text{(D1)}$$

where

$P^0 = (a_1{}^0, ..., a_\mu{}^0) \in I^\mu$	Initial Population
$I = \{0,1\}^l$	Binary Problem Representation
$\delta^0 \subseteq \Re$	Initial Operator Parameter set
$\mu \in N$	Population Size
$\lambda \in N$	Number of Offspring
$l \in N$	Length of Representation
$F : I \rightarrow \Re^+$	Evaluation Function
$G : I^\mu \rightarrow I^\lambda$	Generating function
$U : I^\mu \times I^\lambda \rightarrow I^\mu$	Updating function

For the purposes of this discussion it has been assumed that the aim is to maximise the evaluation function, which is restricted to returning positive values. Although the fitness function is usually seen as a "black box", this restriction will always be possible by scaling etc.

A broad distinction can be drawn between the two types of reproductive operators, according to the number of individuals that are taken as input. In this thesis the term *Mutation* operators is used for those that act on a single individual, producing one individual as the outcome, and *Recombination* operators for those that act on more than one individual to produce a single offspring. Typically, recombination operators have used two parents, and the term "crossover" (derived from evolutionary biology) is often used as a synonym for two-parent recombination.

Since all recombination operators involve the interaction of one individual with one or more others, the general form of these functions is:

$$R : I^\mu \times \delta \rightarrow I \qquad \text{Recombination Operator} \qquad \text{(D2)}$$

where the individuals taking part are drawn from {1... μ} according to some probability distribution p_r, as opposed to the general form:

$$M \quad : \quad I \times \delta \rightarrow I \qquad \text{Mutation Operator} \tag{D3}$$

Using $O \in I^{\lambda}$ to denote the set of offspring, an iteration of the algorithm becomes:

$$O_i^t = M\,R\,(P^t, \delta^t) \qquad \forall\, i \in \{1... \lambda\} \tag{D4}$$

$$P^{t+1} = U\,(O^t \cup P^t) \tag{D5}$$

The "canonical GA" as proposed by Holland is referred to as a Generational GA (GGA). It uses a selection scheme in which all of the population from a given generation is replaced by its offspring, i.e. $\lambda = \mu$, and U is defined by the p.d.f. $p_u\,(a_i^t)$ where

$$p_u(a_i^t) = \begin{cases} 0 & i \in P^t \\ 1 & i \in O^t \end{cases} \tag{D6}$$

Parents for the next round of recombination are chosen with a probability proportional to their relative fitness, R drawing individuals from {1...μ} with a probability distribution function p_r given by:

$$p_r(a_i^t) = \frac{F(a_i^t)}{\sum_{j=1}^{\mu} F(a_j^t)} \tag{D7}$$

Holland, and later DeJong also considered the effect of replacing a smaller proportion of the population in any given iteration (known as generations). At the opposite end of the spectrum to generational GAs lie "steady state" genetic algorithms (SSGAs) such as the GENITOR algorithm [Whitley and Kauth, 1988]. These replace only a single member of the population at any given time ($\lambda = 1$).

It can be shown that under the right conditions, the two algorithms display the same behaviour. In an empirical comparison [Syswerda, 1991] the two variants, were shown to exhibit near identical growth curves, provided that a suitable form of p_u was used to define U for the SSGA. In this case random deletion was used i.e.

$$p_u(a_i^t) = \begin{cases} \frac{n-1}{n} & i \in P^t \\ 1 & i \in O^t \end{cases} \tag{D8}$$

These results are well supported by a theoretical analysis [DeJong and Sarma, 1992] using deterministic growth equations (in the absence of crossover and mutation).This showed that the principal difference is that algorithms displaying a low ratio of λ:μ (SSGAs) display higher variance in their behaviour than those with higher ratios (e.g. 1 for GGAs). This variance decreases with larger populations, but was the basis behind early recommendations for generational algorithms (e.g. Holland's original work and the empirical studies in [DeJong, 1975]) where small populations were used. It was also found that the use of strategies such as "delete-oldest" (a.k.a. FIFO) reduced the observed variance in behaviour whilst not affecting the theoretical performance.

However in practice, it is usual to employ techniques such as "delete-worst" in steady state algorithms. This can be seen as effectively increasing the selection pressure towards more highly fit individuals, and it has been argued that it is this factor rather than any other which accounts for differences in behaviour [Goldberg and Deb, 1991].

In fact since the algorithm is iterative, the two selection distributions are applied consecutively, and so they can be treated as a single operator U with a p.d.f. given by

$$p_s(a_i^t) = p_r\,(a_i^t) * p_u\,(a_i^{t-1}) \tag{D9}$$

1.1. Building Blocks and the The Schema Theorem

Since Holland's initial analysis, two related concepts have dominated much of the theoretical analysis and thinking about GAs. These are the concepts of *Schemata* and *Building Blocks*. A schema is simply a hyper-plane in the search space, and the common representation of these for binary alphabets uses a third symbol- # the "don't care" symbol. Thus for a five-bit problem, the schema 11### is the hyperplane defined by having ones in its first two positions. All strings meeting this criterion are examples of this schema (in this case there are $2^3 = 8$ of them). Holland's initial work showed that the analysis of GA behaviour was far simpler if carried out in terms of schemata. He showed that a string of length l is an example of 2^l schemata, although there will not in general be as many as $\mu * 2^l$ distinct schemata in a population of size μ. He was able to demonstrate the result that such a population will usefully process $O(\mu^3)$ schemata. This result, known as Implicit Parallelism is widely quoted as being one of the main factors in the success of GAs.

Two features are used to describe schemata, the order (the number of positions in the schema which do not have the # sign) and the defining length (the distance between the outermost defined positions). Thus the schema H = 1##0#1#0 has order o(H) = 4 and defining length d(H) = 8.

For one point crossover, the schema may be disrupted if the crossover point falls between the ends, which happens with probability d(H) / (l -1). Similarly uniform random mutation will disrupt the schema with a probability proportional to the order of the schema.

The probability of a schema being selected will depend on the fitness of the individuals in which it appears. Using f(H) to represent the mean fitness of individuals which are examples of schema H, and fitness proportional selection as per (D7), the proportion m(H) of individuals representing schema H at subsequent time-steps will be given by:

m(H, T+1) = m(H,t) * probability of selection * (1 - probability of disruption).

In fact this should be an inequality, since examples of the schema may also be created by the genetic operators. Taking this into account and using the results above gives

$$n(H, t+1) \geq m(H, t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - \left(p_c \cdot \frac{d(H)}{l-1}\right) - p_m \cdot o(H)\right] \quad (1)$$

This is the *Schema Theorem*.

A related concept is the *Building Block Hypothesis (*see [Goldberg, 1989 pp 41-45] for a good description). Simply put, this is the idea that Genetic Algorithms work by discovering low order schemata of high fitness (building blocks) and combining them via recombination to form new potentially fit schemata of increasing higher orders. This process of recombining building blocks is frequently referred to as *mixing*.

1.2. Operator and Parameter Choice

The efficiency of the Genetic Algorithm can be seen to depend on two factors, namely the maintenance of a suitable working memory, and quality of the match between the p.d.f. generated and the landscape being searched. The first of these factors will depend on the choices of population size μ, and selection algorithm U. The second will depend on the action of the reproductive operators (R and M) and the set of associated parameters δ, on the current population.

Naturally, a lot of work has been done on trying to find suitable choices of operators and their parameters which will work over a wide range of problem types. The first major study [DeJong, 1975] identified a suite of test functions and proposed a set of parameters which it was hoped would work well across a variety of problem types. However later studies using a "meta-ga" to learn suitable values [Grefenstette, 1986] or using exhaustive testing [Schaffer et al., 1989] arrived at different conclusions. Meanwhile theoretical analysis on optimal population sizes [Goldberg, 1985] started to formalise the (obvious?) point that the value of μ on the basis of which decisions could be reliably made depends on the size of the search space.

The next few years of research saw a variety of new operators proposed, some of which (e.g. Uniform Crossover [Syswerda, 1989]) forced a reappraisal of the Schema Theory and led to the

focusing on two important concepts.

The first of these, Crossover Bias [Eshelman et al., 1989], refers to the differing ways in which the p.d.f.'s arising from various crossover operators maintain hyperplanes of high estimated fitness during recombination, as a function of their order and defining length. This is a function of the suitability of the p.d.f. induced by the recombination operator to the landscape induced by the problem encoding. These findings have been confirmed by more formal analysis on the relative merits of various recombination mechanisms [DeJong and Spears, 1990, 1992, Spears and DeJong, 1991].

The second concept was that of Safety Ratios [Schaffer and Eshelman, 1991] i.e. of the ratio of probability that a new point generated by the application of reproductive operators would be fitter than its parent(s) to the probability that it is worse. These ratios were shown empirically to be different for the various reproductive operators, and also to change over time. Again these reflect the match between the p.d.f.s induced by given operators on the current population to the fitness contours of the landscape.

This can be demonstrated by a thought experiment using the simplest case of the OneMax function:

$$f(a) = \sum_{i=0}^{l} a_i \qquad \text{(D10)}$$

In a random initial population, an average of half of the genes will have the non-optimal value, and so on average half of all mutations will be successful. However, as the search progresses, and the mean fitness of the population increases, the proportion of genes with the optimal value increases, and so the chance that a randomly distributed mutation will be beneficial is decreased. This was formalised in [Bäck, 1992a, 1993] where an exact form for the Safety Ratio was derived for a mutation probability p. This could not be solved analytically, but was optimised numerically and found to be well fitted by a curve of the form: $p_{opt} \approx \frac{1}{2(f(a)-1)-l}$

As can be seen, this is obviously time-dependant in the presence of any fitness-related selection pressure.

These considerations, when coupled with interactions with other Evolutionary Algorithm communities who were already using adaptive operators (e.g. the (1+1) [Rechenberg, 1973] and (μ λ) [Schwefel, 1977, 1981] Evolutionary Strategies), has led to an ever increasing interest in the possibilities of developing algorithms which are able to adapt one or more of their operators or parameters in order to match the p.d.f. induced by the algorithm to the fitness landscape.

In terms of the formalisation above, this equates to potentially allowing for time variance in the functions R,M and U, the parameter set δ, and the variables μ,and λ. It is also necessary to provide some set of update rules or functions to specify the form of the transitions $X^t \rightarrow X^{t+1}$ (where X is the facet of the algorithm being adapted).

As soon as adaptive capabilities are allowed, the size of the task faced by the algorithm is increased, since it is now not only traversing the problem space but also the space of all variants of the basic algorithm. This traversal may be constrained, and may take several forms, depending on the scope of change allowed and the nature of the learning algorithm. It may vary from the simple time-dependant decrease in the value of a single parameter according to some fixed rule (e.g [Fogarty, 1989]) to a complex path which potentially occupies any position in the hyper space defined by variations in R, M and δ, and which is wholly governed by self-adaptation and the updating process (e.g. [Smith and Fogarty, 1996a]).

If the exact nature of the search landscape is known, then it may be possible to follow the optimal trajectory through algorithmic space for a given problem. Unfortunately this will not generally be the case, and in practice it is necessary to provide some kind of learning mechanism to guide the trajectory. Inevitably there will be an overhead incurred in this learning, but the hope is that even allowing for this, the search will be more efficient than one maintaining a fixed set of operators and parameters.

In fact, the "No Free Lunch" theory [Wolpert and Macready,1995] tells us that averaged over all problem classes, all non-revisiting algorithms will display identical performance. Since for fixed operator/parameter sets there will be problems for which they are optimal, it follows that there must be other classes of problems for which the performance is very poor. The intention behind the use of adaptive operators and parameters within GAs is to create algorithms which display good "all-round" performance and are thus reliable for use as optimisers on new problems.

1.3. A Framework for Classifying Adaptation in Genetic Algorithms

The various methods proposed for incorporating adaptation into Genetic Algorithms, may be categorised according to three principles, namely *What* is being adapted (operators, parameters etc.), the *Scope* of the adaption (i.e. does it apply to all the population, individual members or just sub-components) and the *Basis* for change (e.g. externally imposed schedule, fuzzy logic etc.). In the following sections these principles are described in more depth, along with some examples of each type of adaptation.

1.3.1. *What* is being Adapted?

As has been described above, the genetic algorithm may be viewed as the iterated application of two processes: *Generating* new points in the landscape (via probabalistic application of recombination and/or mutation operators to the previous population), and *Updating* (via selection and resizing) to produce a new population based on the new set of points created and (possibly) the previous population.

The majority of the proposed variants of the simple genetic algorithm only act on a single operator, and furthermore it is true to say that most work has concentrated on the reproductive operators i.e. recombination and mutation. Whilst considerable effort has been expended on the question of what proportion of the population should be replaced at any given iteration, the function U, used to update the population, once chosen, tends to be static.

Much theoretical analysis of GA's has focused on the twin goals of exploration of new regions of the search space and exploitation of previously learned good regions (or hyperplanes) of the search space. The two terms may be explained by considering how the p.d.f.s governing generation of new points change over algorithmic time, remembering that the p.d.f.s are governed jointly by the actions of the reproduction and selection operators. An explorative algorithm is one in which a relatively high probability is assigned to regions as yet unvisited by the algorithm, whereas an exploitative algorithm is one in which the p.d.f. represents rather accumulated information about relatively fitter regions and hyperplanes of the search space. Thus the p.d.f. of an explorative algorithm will change more rapidly than that of an exploitative one.

Broadly speaking most adaptive algorithms work with the settings of either a Generational GA (GGA) or a "Steady State" GA (SSGA) which represent the two extremes of i) generating an entirely new population (through multiple reproductive events) and ignoring the previous population in the selection process or ii) generating and replacing a small number (usually one) of individuals (via a single reproductive event) at each iteration.

If the reproductive operators which are applied produce offspring that are unlike their parents, then the updating mechanisms of GGAs and SSGAs may be seen as favouring exploration and exploitation respectively. (Note that the equivalence of the growth curves for SSGAs and GGAs described earlier on page 10 was only in the absence of crossover or mutation).

However the extent to which offspring differ from their parents is governed not simply by the type of operator applied, but by the probability of its application as opposed to simply reproducing the parents. Thus for a given population and reproductive operator, it is possible to tune the shape of the induced p.d.f. between the extremes of exploration and exploitation by altering the probability of applying the reproductive operator, regardless of the updating mechanism.

It is this last point which has led to a focusing on the adaptation of the reproductive operators, since by changing the amount of disruption they induce it is possible to indirectly adapt the proportion

of the population which is replaced at each iteration, whilst using simple and efficient selection mechanisms. This allows the tuning of the updating mechanism to particular application characteristics e.g. the possibility of parallel evaluations etc.

A number of different strands of research can be distinguished on the basis of what the algorithms adapt, e.g. operator parameters, probabilities of application or definitions.

In some ways the simplest class are those algorithms which use a fixed set of operators and adapt the probability of application of those operators. Perhaps the two best known early examples of work of this type are the use of a time-dependant probability of mutation [Fogarty, 1989] and the use of varying probabilities of application for a few simple operators (uniform crossover, averaging crossover, mutation, "big creep" and "little creep") depending on their performance over the last few generations [Davis, 1989]. Many later authors have proposed variants on this approach, using a number of different learning methods as will be seem later. A similar approach which changes the population updating mechanism by changing the selection pressure over time can be seen in the Grand Deluge Evolutionary Algorithm [Rudolph and Sprave, 1995].

An extension of this approach can be seen in algorithms which maintain distinct sub-populations, each using different sets of operators and parameters. A common approach is to use this approach with a "meta-ga" specifying the parameters for each sub-population- e.g. [Grefenstette, 1986, Kakuza et al., 1992, Friesleben and Hartfelder,1993] to name but three. Although the searches in different (sub)populations may utilise different operators, each can be considered to have the full set available, but with zero probability of applying most, hence these are grouped in the first category. A similar approach is taken to population sizing in [Hinterding et al., 1996]. All of this class of algorithms are defined by the set:

$$GA = (P^0, \delta^0, l, F, G, U, \Gamma) \tag{D11}$$

where $\Gamma: \delta \rightarrow \delta$ is the transition function such that $\delta^t = \Gamma(\delta^{t-1})$. As has been described, the function Γ will often take as parameters some "history" of relative performance, and the number of generations iterated. Note that here λ and μ are subsumed into δ.

A second class of adaptive GA's can be distinguished as changing the actual action of the operator(s) over time. An early example of this was the "Punctuated Crossover" mechanism [Schaffer and Morishima, 1987] which added extra bits to the representation to encode for crossover points. These were allowed to evolve over time to provide a 2 parent N-point recombination mechanism, where N was allowed to vary between zero and the length of the string. The Lego and related Apes mechanisms [Smith and Fogarty,1995, 1996a, 1996b] evolve the "units of heredity" which determine recombination and mutation mechanisms, through the use of genetic encoded "links" between loci on the representation.

In both of these cases, the form of the reproductive operators are constant but their expression is dependant the particular individuals to which they are applied. The definition of the population thus changes to $P^t \in I^\mu \times X^\mu$ where the form of X will depend on the algorithm: for example in Schaffer and Morishima's work $X = \{0,1\}^{l-1}$ with a 1 in position j denoting that crossover should take place after locus j. The information governing the crossover operator is genetically encoded, and subject to mutation and recombination along with the problem encoding. The transition function here is simply the genetic algorithm itself i.e. $\Gamma = UMR$.

Algorithms which encode their operators and/or parameters in the genotype and use transition functions of this form are generally referred to as *Self-Adaptive.*

Within the category of algorithms which change the form of their operators rather than simply the probability of application fall those algorithms which alter the mechanisms governing the updating of the working memory by continuously changing its size (e.g.[Smith R., 1993, Smith R. and Smuda, 1995, Arabas et al., 1994,). In the first two examples, the form of p_u is determined by comparing pairs of individuals as matched during crossover. In the work of Arabas et al., each individual is given a "lifespan", g, at the time of its creation, according to its relative fitness. The p.d.f. governing the creation of a new population thus changes according to the current state of the

population

i.e. $P = (a_1,..., a_\mu, g_1,...g_\mu)$ and $p_u(a_i^t) = \begin{cases} 0 & g_i \geq t \\ 1 & g_i < t \end{cases}$ (D12)

This can be contrasted to the SSGA with FIFO deletion where the lifespan of every individual is μ evaluations.

An alternative approach is to alter the representation of the problem itself: this can be viewed as an attempt to make the landscape suit the p.d.f.s induced by the operators rather than vice-versa. Work on "Messy GA's" ([Goldberg et al., 1989] and many subsequent publications) is based on finding appropriate linkages between genes, using a floating representation where the order of the variables is not fixed. Since the "cut and splice" recombination operator tends to keep together adjacent genes, this can be seen as moulding the landscape to suit the operators. Similarly the ARGOT strategy [Schaefer, 1987] adaptively resizes the representation according to global measures of the algorithm's success in searching the landscape, and more recently work on co-evolving representations [Paredis, 1995] can be seen in this restructuring light, as can the Adaptive Penalty Functions of [Eiben and van der Hauw, 1997, Eiben et al. 1998].

1.3.2. What is the *Scope* of the Adaptation?

The terminology of [Angeline, 1995] defines three distinct levels at which adaptation can occur in evolutionary algorithms. *Population-level* adaptations make changes which affect the p.d.f. contribution from each member of the current population in the same way. *Individual-level* adaptations make changes which affect the p.d.f. contribution from each member of the population separately, but apply uniformly to each of the components of the individuals' representation. At the finest level of granularity are *Component-level* adaptations, where the p.d.f. contributions from each component of each member may be changed individually.

Inevitably, the scope of the adaptation is partly determined by what facet of the algorithm is being adapted, and the nature of the transition function. Changes in population size [Smith, R. and Smuda 1995], or problem restructuring (e.g. [Schaefer, 1987, Eiben and van der Hauw, 1997, Eiben et al. 1998]) for example can only be population level adaptations. However changes in some operators can occur at any of the three levels (for example mutation rates in Evolutionary Strategies), and so the notion of Scope forms a useful way of distinguishing between algorithms.

The forms of Γ defined by different adaptive algorithms are all attempts to tune the p.d.f.s arising from the action of genetic operators on the population to the topography of the problem landscape. Earlier an algorithmic space was described encompassing any given variant of the GA, and adaptation was defined as a traversal of this algorithmic space. For population level changes the trajectory will be a single path, as all members of the algorithm's population at any given time will share a common set of operators and parameters. For individual and component level adaptations, the path will be that of a cloud, as each member follows a slightly different course. This cloud is composed of a number of traces which may appear and disappear under the influence of selection.

1.3.2.1. Population-Level Adaptation.

Recall from above that Genetic Algorithms can be viewed as the iterated probabalistic application of a set of operators on a population, and that in most cases these operators are static - that is to say that their forms and parameters are fixed and apply uniformly to the whole population i.e. they are *global*. Population-level adaptation algorithms can be typified as using a fixed set of global operators, but allowing their parameters to vary over time. The most important parameter is of course the probability of application. The various "meta-ga" algorithms mentioned above, and the competing sub-populations of the breeder GA e.g. [Schlierkamp-Voosen and Mühlenbein, 1994] belong firmly to this level. In general algorithms using adaptation at the population level will take the form of (D11). They may differ in the set of operators used, but are principally distinguished by the transition function Γ.

There are a number of theoretical results which provide support for this kind of approach, e.g. regarding the time-variance of the optimal mutation rate [Mühlenbein, 1992, Hesser and Manner, 1991] (see also the discussions Bäck's results, and of Schaffer & Eshelman's experimental results regarding the time dependencies of Safety Ratios for mutation and crossover, on page 12).

In [Fogarty, 1989] an externally defined form is used to reduce the mutation rate over time, in a similar fashion to the cooling schedules used in Simulated Annealing. However a more common approach is to adjust one or more parameters dynamically in accordance with the performance of the algorithm or some measured quantity of the population.

A well known, and popular approach (e.g. [Davis, 1989, Corne et al., 1994, Julstrom, 1995]) is to keep statistics on the performance of offspring generated by various reproductive operators relative to their parents. Periodically "successful" operators are rewarded by increasing their probability of application relative to less successful operators. This approach does require extra memory, since it is usually found necessary to maintain family trees of the operators which led to a given individual, in order to escape from credit allocation problems.

Other authors have proposed control strategies based on simpler measures. Eshelman and Schaffer use the convergence of the population to alter the thresholds governing incest prevention, and the time spent without improvement to govern the probability of restarting the GA using vigorous mutation [Eshelman and Schaffer, 1991]. This latter is similar to many strategies for tracking changing environments, where a drop in the performance of the best member of the current generation is used to trigger a higher rate of mutation e.g. [Cobb and Grefenstette, 1993].

In [Lee and Takagi, 1993] fuzzy rules are learned and used to control various parameters based on the relative performance of the best, worst and mean of the current population. This concept of observing the fitness distribution of the population and then altering parameters according to a set of rules is also used in [Lis, 1996] where the mutation rate is altered continuously in order to keep a fitness distribution metric - the "population dispersion rate" within a desired range.

A more complex approach, and one which at first appears to belong to the component level is that of [Sebag and Schoenauer, 1994] who maintain a library of "crossover masks" which they label as good or bad. Inductive learning is used to control application of crossover and the periodic updating of mask strengths. However the rules learnt to control crossover are applied uniformly and so this belongs firmly in the category of population-level adaptation.

Perhaps the most obvious method of achieving population-level adaptation is to dynamically adjust the size of the population itself by creating or removing members according to some global measurement. Two approaches have been recently reported. In [Smith R., 1993, Smith R.and Smuda, 1995] the size is adjusted according to estimated schema fitness variance, whereas in [Hinterding et al., 1996] three populations of different sizes are maintained, with periodic resizing according to their relative successes.

Finally many of the proposed methods of adapting representations mentioned above plainly fall into this category.

1.3.2.2. Individual Level Adaptation

An alternative approach to adaptation is centred on consideration of the individual members of the population rather than the ensemble as a whole. Thus as a simple example, a global level adaptation may vary the probability of mutation for the whole population, whereas an individual level algorithm might hold a separate mutation probability for each member of the population. Considering the p.d.f. governing generation as the sum of the contributions from each member, then population level changes affect the way in which the contributions are determined uniformly, whereas individual level adaptations affect the p.d.f. contributions for each member separately.

A frequently claimed justification for this approach is that it allows for the learning of different search strategies in different parts of the search space. This is based on the not unreasonable assumption that in general search space will not be homogeneous, and that different strategies will be better suited to different kinds of sub-landscape.

As a crude metaphor, imagine a blind robot, equipped only with an altimeter, dropped at random

on the earth and trying to reach the highest point it can. If on a large flat region, it would be better employed using a wider search (more uniform p.d.f. i.e. exploration), whereas if dropped by chance in the Himalayas such a search might rapidly lead it out of the mountain range. In the latter case a more localised search (i.e. exploitation) would be far preferable.

Algorithms at this level have been proposed which fall into a number of different categories in terms of the basis and scope of adaptation.

Perhaps the most popular class are algorithms which encode some parameters for an operator into the individual and allow these to evolve, using the updating process itself as the basis for learning. An early example of this was the "Punctuated Crossover" mechanism of [Schaffer and Morishima, 1987]. This added a mask to the representation which was used to determine crossover points between two parents during recombination, and which was evolved along with the solutions (as described on page 14 above). The results reported were encouraging, but this may have been due to the high number of crossover points evolved (compared to the algorithm used for comparison). In [Levenick, 1995] a similar mechanism is investigated, but with the added bits coding for changes in crossover probability at those points rather than deterministic crossing.

A more complex version of these, which uses discrete automata to encode for the crossover probabilities at each locus, is discussed in [White and Oppacher, 1994]. In this algorithm the representation is extended to store the state of the automata at each locus i.e. $a_i = \{0,1\}^l \times \{0,... n\}^{l-1}$. Unlike the two algorithms just mentioned, the transition function Γ, which maps $a_i^t \rightarrow a_i^{'t}$ is not simply the combination of recombination and mutation acting on the genotypes prior to the action of selection. Rather a set of rules are applied based on the relative fitness of offspring and parents, which update the state of each automata.

An alternative method for controlling recombination strategies was used in [Spears, 1995] where a single bit was added to the genotype and used to decide whether two-point or uniform crossover would be used when an individual reproduced. Again self-adaptation was used for the transition. This control method differed from the above mechanisms in that they allow the form of the operator itself to change, whereas this algorithm effectively associates two operators with each individual and associates probabilities of 0 or 100% of application with each.

It is noticable that this was compared with a population level mechanism where the relative proportion of bits encoding for the two operators was used to make a global decision about operator probabilities. Although these two alternatives should provide the same ratio of usage of the two operators, the latter mechanism was found to work far less well, suggesting that there is indeed a merit in attaching reproductive strategies to particular individuals. Certainly the two methods can yield different p.d.f.s for the next generation of points.

Perhaps more common has been the adoption of the idea of self adaptive mutation rates from Evolutionary Strategies. The addition of extra bits to the genome to code for the mutation rate for that individual was first investigated for GGAs in [Bäck, 1992b]. It was found that the rates evolved were close to the theoretical optimum for a simple problem, providing that a sufficiently rigorous selection mechanism was used. The second level of adaptation in [Hinterding et al., 1996] uses a similar encoding to control the standard deviation of the Gaussian distributed random function used to mutate genes. In [Smith and Fogarty, 1996c] self adaptive mutation was translated into a Steady State algorithm, with the addition of a $(\lambda,1)$ "inner GA". This was found to be necessary to provide the selection pressure for self adaptation, but also provided a kind of local search.

However not all algorithms using adaptation at the individual level rely on endogenous control and self-adaptation. As with population level adaptations, a number of other algorithms can be used to control the behaviour of individuals.

A typical example is found in [Srinivas and Patnaik, 1994] where the probabilities of applying mutation or crossover to an individual depend on its relative fitness, and the degree of convergence (of fitnesses) of the population. This fitness-dependant control of the p.d.f. contribution from each individual is used to control the global balance of exploitation and exploration, so that the information in relatively fitter individuals is exploited whereas exploration is achieved by associating "broader"

p.d.f.s with the less fit members. Another approach based on relative local fitness in a structured population uses a pattern of rules to control the reproduction strategy (replication, crossover or mutation) based on metaphors of social behaviour [Mattfeld et al., 1994]. Similarly the Grand Deluge Evolutionary Algorithm of [Rudolph and Sprave, 1995] adapts the "acceptance" threshold governing the updating process separately for each point, based on relative local fitness over a period of time. Effectively all of these exogenously controlled algorithms take the form:

$$GA = (P^0, \overline{\delta^0}, l, F, G, U, \Gamma) \quad \text{(D13)}$$

where there is now a vector of sets of parameters $\bar{\delta} = (\delta_1, \ldots, \delta_\mu)$

Similarly measures of relative fitness and convergence are used to determine the "lifespan" given to a newly created individual in [Arabas et al., 1994]. This novel approach controls the contents and size of the working memory by assigning a fixed lifetime to each individual after which it is removed from the memory (see page 14). This is one of the very few algorithms reported providing dynamic population sizing, and the results reported suggest that this is a promising line of research.

1.3.2.3. Component-Level Adaptation

This is the finest level of granularity for adaptation: here algorithms allow different reproduction strategies for different parts of the problem representation. Again these ideas originate from Evolutionary Strategies research, which has progressed over the years from a single parameter controlling the mutation step size for the population, through individual step-sizes for each member, to having a separate mutation step size encoded for each component being optimised.

This can be considered as a means of allowing the focus of search to be directed, and the principal advantage is that it allows for a much finer degree of tuning of the p.d.f. contribution associated with each individual. All of the work done at this level has used self-adaptation as the method of learning parameters.

This approach was tried in the self-adaptive mutation mechanism of [Bäck, 1992b] and compared with individual level adaptation. Essentially a real-valued mutation rate was added for each locus i.e. $a_i = \{0,1\}^l \times \Re^l$. The results suggested that in certain circumstances the results were advantageous, but that on other landscapes the learning overheads associated with all the extra mutation parameters slowed the search down. The conclusions seemed to be that such a mechanism could be very effective if the components were properly defined i.e. the level of granularity was chosen suitably.

An attempt to solve some of these problems can be seen in the Lego mechanisms [Smith and Fogarty, 1995, 1996b]. These are somewhat similar to the "Punctuated Crossover" algorithms of Schaffer and Morishima in adding extra bits to the representation to determine whether two adjacent genes may be broken by crossover. However the emphasis is different, in concentrating on finding blocks of co-evolved linked genes. The recombination mechanism also differs in that blocks of genes may be chosen from the whole population (rather than just two parents) when a new individual is formed. This evolution of linkage has to be seen therefore as an adaptation of recombination strategy at the component level, since individuals have no meaning in the context of parents other than as contributing to a "genepool" from which new individuals are assembled. This emphasis on evolving successful components rather than individuals was taken further in the Apes algorithm [Smith and Fogarty, 1996a]. This included component level adaptation of mutation rates by attaching a mutation rate to each block, which is also self-adapted.

1.3.3. What is the Basis for Adaptation?

The final distinction between classes of adaptive genetic algorithms is perhaps the most important, namely the basis on which adaptation is carried out. This in turn hinges on two factors, firstly the evidence upon which adaptation is carried out, and secondly the rules or algorithm which define how changes are effected. In terms of our notation this second factor is the definition of the transition function Γ, and the evidence translates to the choice of parameters or inputs for that function.

The first distinction between kinds of adaptive algorithms is drawn on the basis of the evidence that they consider.

In one camp are all those algorithms which take as their evidence differences in the relative performance of strategies. These include Self-Adaptive and meta-GAs. Those algorithms which adjust operator probabilities based on records of performance, such as those of [Davis 1989, Corne et al. 1994, Julstrom 1995] fall into this category as their probabalistic nature allows the concurrent testing of multiple strategies (the difference being that they use predefined rules to adjust the strategies). Also into this category fall those algorithms which draw conclusions based on observations of strategy performance such as Sebag and Schoenauer's Inductively Learned Crossover Masks, and White and Oppacher's Automata Controlled Crossover.

Into the other camp fall those algorithms which adapt strategies based on empirical evidence which is not directly related to the strategy followed. This might be convergence statistics (in terms of the population fitness, or the allele distribution), observed relative schema fitnesses etc. Although some use simple rules e.g. previously learned Fuzzy Rule-Sets [Lee and Takagi, 1993] or predefined schedules [Fogarty, 1989], most base the derived trajectory on an attempt to fulfil some criterion or maintain a population statistic. Examples of this latter are the "Population Dispersal" metric of [Lis, 1996], and "Schema Fitness Variance" [Smith, R. and Smuda, 1995].

Consideration of the nature of the transition function itself suggests another important distinction that can be drawn between two types of algorithm. These have been labelled as "Uncoupled vs. Tightly Coupled" [Spears, 1995], or "Empirical vs. Absolute" update rules [Angeline 1995]. Essentially in the first type of algorithm, the transition function Γ is externally defined - previously referred to as exogenous algorithms. In the second type (endogenous algorithms) the operator parameters (and possible definitions) are encoded within individuals, and Γ is itself defined by the genetic operators *U*, *M* and *R*, i.e.the GA itself is used to determine the trajectory. This latter approach is more commonly known as Self-Adaptation.

To clarify this point, in all algorithms a set of evidence is considered, on the basis of which the trajectory of the algorithm is decided. If the evidence is not relative strategy performance, then the control mechanism is necessarily uncoupled from the generation-updating mechanisms of the evolutionary algorithm itself. If the evidence is relative strategy performance, then the control mechanism may be externally provided (as in, for example, Davis' work) or may be a function of the genetic operators of the algorithm itself i.e. Self Adaption.

In most uncoupled adaptive algorithms the evidence takes the form of statistics about the performance of the algorithm, such as the fitness distribution of the population, "family trees", or simply the amount of evolutionary time elapsed. The important factor is that the mechanism used to generate the new strategies based on this evidence is *externally provided* in the form of a learning algorithm or a set of fixed rules.

A number of Self-Adaptive algorithms have been described above, and their reported success is perhaps not surprising in the light of the considerable research into these topics in the fields of Evolution Strategies and Evolutionary Programming. The rationale is essentially two-fold, firstly that if strategies are forced to compete through selection, then what better proof of the value of a strategy than its continued existence in the population. Secondly, and perhaps less tritely, the algorithmic space being searched has an unknown topography, but is certainly extremely large, very complex and highly epistatic, since there is a high degree of dependency between operators. Evolutionary Algorithms have repeatedly been shown to be good at searching such spaces...

This same rationale is used for many of the meta-ga strategies, the "competing sub-populations" of the Breeder GA [Schlierkamp-Voosen and Mühlenbein, 1994], and the population sizing adaptation described in [Hinterding et al., 1996]. However it is more correct to describe these as Uncoupled Adaptations since the strategies are not coded for directly in the populations, and the evidence for change is the relative performance of (sub) populations of individuals in a given time period. In other words, although it is still true that $\Gamma = UMR$, in this case the operators and parameters defining Γ are those of the meta-algorithm rather than the "ground level" GA. Since the problem representation, and the representation used in the meta-ga are highly unlikely to be the same, the

operators and parameters will also be different.

1.4. Discussion

The framework proposed above categorises algorithms according to three principles. The first two of these are what features of the algorithm are susceptible to adaptation, and the granularity at which this is done. Separate from these, a further important distinction was drawn according to the basis for change. This took two forms, namely the type of evidence used as input to the strategy-deciding algorithm, and that algorithm itself.

There are a number of possible ways of representing this categorisation, in Figure 1, a simplified taxonomic tree is shown to illustrate the main branches of Adaptive Genetic Algorithms (algorithms are referred to by their first authors).

The first branch of the tree is evidence for change (or more formally the inputs to the transition function). If this is the relative performance of different strategies then there is a further subdivision according to whether the transition function is tightly coupled (Self Adaptation) or uncoupled. If the input to the transition function is some feedback other than relative strategy performance, then the nature of the transition function is necessarily uncoupled from the evolutionary algorithm, so the second branch point does not apply. Below all these branches are leaves corresponding to the scope of the adaptation.

The nature of the field, and the absence of a standard test suite of problems make it impossible to compare algorithms on the basis of reported results and declare a universal winner. Indeed the very considerations that drive research into adaptive algorithms make such statements meaningless. It is however possible to draw a few conclusions.

Firstly, those algorithms based on observing the relative performance of different strategies appear to be most effective. This seems to follow naturally from the fact that GA theory is currently not sufficiently advanced either to permit the specification of suitable goals in terms of other metrics, or (more importantly) how to achieve them. It is ironic that perhaps the most widely quoted paper on adaptive strategies, the externally defined time-decreasing mutation rate of [Fogarty, 1989] is also one of the most widely misquoted works in the field, since this was concerned specifically with initially converged populations.

Secondly, there appears to be a distinct need for the maintenance of sufficient diversity with the population(s). It is the experience of several authors working with adaptive recombination mechanisms that convergence makes the relative assessment of different strategies impossible. Variety within the population is vital as the driving force of selective pressure in all Evolutionary Algorithms, and will be doubly so in Self-Adaptive algorithms. This is less of a problem for algorithms manipulating mutation rates as mutation is generally a force for increased diversity.

Thirdly, there are powerful arguments and empirical results for pitching the adaptation at an appropriate level. The success of individual level adaptive reproduction schemes appears convincing, and there is promise in the various methods proposed for identifying suitable components via linkage analysis which would allow adaptation at an appropriate level. However as Angeline points out, and Hinterding demonstrates, there is scope for adaptation to occur at a variety of levels with the GA

Finally adaptation can be seen as searching the space of possible configurations for the GA. This space is one about which little is known, and needless to say the landscape will depend on what is being optimised (e.g. best solution in a given time, mean performance etc.). However it is the fact that little is known about it which makes it so similar to those problem landscapes of interest. All the arguments used in favour of adaptive GA's in the first section of this chapter would appear to apply equally to the search of this space.

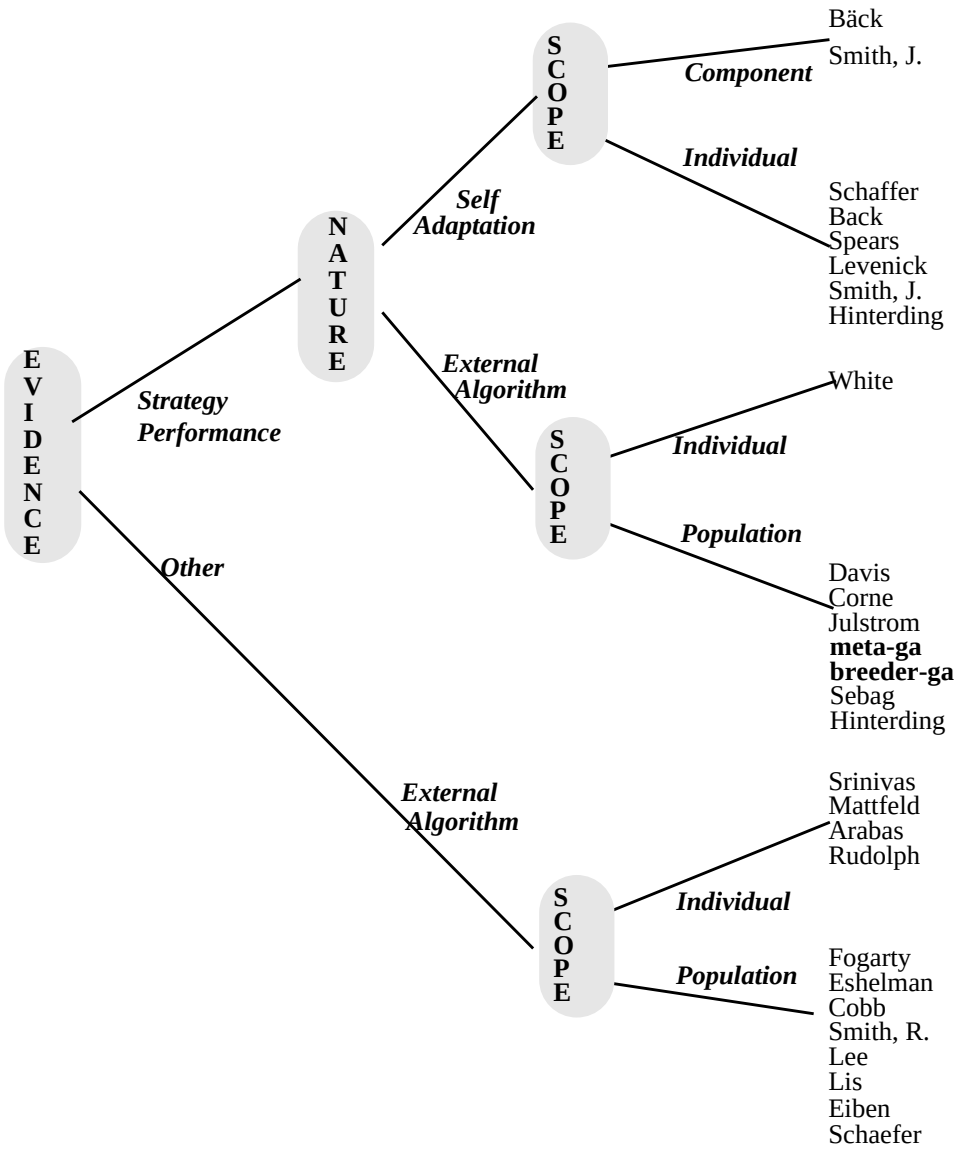


Figure 1: Taxonomy of Adaptive Genetic Algorithms

Chapter Two
Recombination and Gene Linkage

2. Introduction

Having previously discussed the rationale behind adaptive genetic algorithms, and provided a basis for the choice of self-adaption as the preferred method, this chapter concentrates on the recombination operator. It is this operator which principally distinguishes genetic algorithms from other algorithms, as it provides a means by which members of the population can interact with each other in defining the p.d.f. which governs the generation of new points in the search space.

2.1. The Importance of Recombination

Since Holland's original formulation of the GA there has been considerable discussion of, and research into, the merits of crossover as a search mechanism. Much of this work has been concentrated on contrasting the behaviour of algorithms employing recombination as one of the reproductive operators as opposed to those using only mutation-based operators. Into this second class fall algorithms such as Evolutionary Strategies (either (1+1)[Rechenberg, 1973] or (1,λ) [Schwefel, 1977]), Evolutionary Programming (Fogel et al. 1966), various Iterated Hillclimbing algorithms and other neighbourhood search mechanisms such as Simulated Annealing [Kirkpatrick et al.,1983].

The conditions have been studied in which sub-populations using crossover would “invade” populations using mutation alone by the use of a gene for crossover [Schaffer and Eshelman, 1991]. It was found that providing there was a small amount of mutation present then crossover would always take over the population. This was not always beneficial to the search, depending on the amount and “locality” of epistasis in the problem encoding.

They also confirmed empirically that the safety ratio of mutation decreased as the search continued, whereas the safety ratio of crossover (especially for the more disruptive uniform crossover) increased as the search progressed. For both algorithms the ratio was never above unity, i.e. offspring are more likely to be worse than better when compared to their parents.

Some reasons why mutation is decreasingly likely to produce offspring which are fitter than their parents were discussed earlier (see page 12). The probability that crossover will display an increasing safety ratio on a given problem representation depends on a number of factors. Although the state of GA theory is not sufficiently advanced to be prescriptive, two closely related factors which have been widely recognised and investigated are:

1. Crossover is much less effective than mutation at resisting the tendency of selection pressure towards population convergence - once all of the population has converged to the same allele value at a locus, crossover is unable to reintroduce the lost allele value[s], (unlike mutation). As a result of this, as the population converges and identical parents are paired with increasing frequency, crossover has less and less effect, sending the safety ratio towards unity. This was demonstrated in experiments using the adaptive “Punctuated Crossover” mechanism [Schaffer and Morishima, 1987] which allowed a traditional G.A. to learn binary decisions about effective crossover points at different stages of the search. It was found that as the search continued the number of encoded crossover points increased rapidly, although the number of productive crossover operations did not.

2. Until the population is almost entirely converged, crossover has a greater ability to construct higher order schemata from good lower order ones (the Building Block hypothesis). This ability is modelled theoretically in [Spears, 1992] where it is shown that crossover has a higher potential than mutation. The nature (in terms of order and defining length) of lower order schema that will be combined depends on the form of the crossover operator (this is discussed further in section 2.2), and so the quality of the higher order building blocks constructed will depend in part on a suitable match between the epistasis patterns in the representation and the operator.

The quality will also depend on the accuracy of the estimates of the fitnesses of competing low order schemata. As time progresses, and more instances of the low order schemata are sampled, these estimates are likely to become more accurate. If crossover can combine small building blocks with

high probability, then the Safety Ratio is likely to rise initially as these schema fitness estimates improve.

As the size of the building blocks increases over time there is a secondary effect that population convergence biasses the sampling which can have deleterious effects. However as noted, (and unlike mutation) the probability of constructing new high order hyperplanes depends on the degree of convergence of the population. Therefore whether the combination of relatively fit low order hyperplanes is beneficial or not, the effect on the Safety Ratio will diminish over time as the population converges.

Although Safety Ratios are a useful tool for understanding the differences between operators, they can also be misleading as they consider relative rather than absolute differences in fitness between offspring and parents. As an example on their "trap" function, Schaffer and Eshelman found that the ratio diminished over time for mutation and increased over time for Uniform Crossover, but the former had a far better mean best performance in terms of discovering optima.

The Royal Road landscapes [Mitchell et al., 1992, Forrest and Mitchell, 1992] were created with building blocks specifically designed to suit the constructive powers of crossover. Mitchell et al. compared results from algorithms using both crossover and mutation with those from iterated hillclimbing techniques. It was found that contrary to the authors' expectations, Random Mutation Hill Climbing outperformed algorithms using crossover on these "Royal Road" functions. Analysis showed that this was due to the tendency of crossover based algorithms (especially those using a low number of crossover points) to suffer from "genetic hitch-hiking" whereby the discovery of a particularly fit schema can lead to a catastrophic loss of diversity in other positions.

These issues have in fact been studied for many decades prior to the invention of genetic algorithms, or even modern computers. The origins of sexual recombination, and what benefits it confers, have long been studied by Evolutionary Biologists (see [Maynard-Smith, and Szathmary, 1995 chapter 9] for a good overview).

From the point of view of the "selfish gene" [Dawkins, 1976], it would initially appear that "parthenogenesis" (asexual reproduction) would appear to confer an advantage, since all of an organism's genes are guaranteed (saving mutation) to survive into the next generation. It might be expected that a gene for parthenogenesis that arose by mutation in a sexually reproducing population would soon dominate. However, when a finite population is considered, the effects of Muller's Ratchet [Muller, 1964] come into play. Simply stated, this is the effect that deleterious mutations will tend to accumulate in a population of fixed size. In practice this tendency is counteracted by selection, so that an equilibrium is reached. The point (i.e. the mean population fitness) at which this occurs is a function of the selection pressure and the mutation rate. By contrast, if recombination is allowed, then Muller's ratchet can be avoided, and this has led to one view of the evolution of recombination as a repair mechanism - *"It is now widely accepted that the genes responsible for recombination evolved in the first place because of their role in DNA repair"* [Maynard-Smith, 1978, p36].

A second strand of reasoning used by biologists is similar to the Building Block hypothesis, and shows that in changing environments populations using recombination are able to accumulate and combine successful mutations faster than those without [Fisher, 1930]- this is sometimes known as "hybrid vigour".

2.2. Recombination Biases

Considerable effort has been put into deriving expressions which describe the amount and type of schema disruption caused by various crossover operators and how this is expected to affect the discovery and exploitation of "good" building blocks. This has been explored in terms of positional and distributional biases, experimentally [Eshelman et al., 1989, Eshelman and Schaffer, 1994] and theoretically [Booker, 1992, Spears and DeJong, 1990, 1991].

In brief, an operator is said to exhibit positional bias if the probability of disrupting schema H of a given order $o(H)$ is a function of the defining length $d(H)$. Thus for example One Point Crossover exhibits high positional bias since for a fixed o, the probability of a randomly chosen crossover point

falling between the ends of a schema *H* of order *o* will be directly proportional to *d(H)* according to *d(H) / (l-1)* as was seen earlier (page 11). In fact the probability of disrupting a schema is purely a function of *d(H)*, and since all genes between the end of the representation and the crossover point are transmitted together, is independent of the value of *o(H)*.

Distributional Bias is demonstrated by operators in which the probability of transmitting a schema is a function of its order. For example, Syswerda defines Uniform Crossover [Syswerda, 1989] as picking a random binary mask of length *l*, then taking the allele in the *i*th locus from the first parent if the corresponding value in the mask is 0, and from the second if it is 1. Since each bit in the mask is set independently, the probability of picking all *o* genes from a schema in the first parent is purely a function of the value of *o* and is independent of *d(H)*: $p(survival) = p^{o(H)}$, where $p = 0.5$. This can be generalised by considering masks which have a probability p ($0 < p < 1.0$) of having the value 0 in any locus, and so the amount of bias exhibited can be tuned by changing p.

Positional bias, the tendency to keep together genes in nearby loci, is responsible for the phenomenon of genetic hitch-hiking, whereby the genes in loci adjacent to or "inside" highly fit schemata tend to be transmitted by recombination along with them. This assigns to the hitch-hiking schemata an artificially high mean fitness since they will tend to be evaluated in the context of highly fit individuals. This is known as "spurious correlation" and can lead to a loss of diversity, and premature convergence to a sub-optimal solution, as the proportions of the hitch-hiking schema increase according to (1).

2.3. Multi-Parent and Co-evolutionary Approaches

Recently the investigation of multi-parent crossover mechanisms, such as Bit Simulated Crossover (BSC) [Syswerda, 1993], has led to more discussion on the role of pair-wise mating and the conditions under which it will have an advantage over population based reproduction. Empirical comparisons [Eshelman and Schaffer,1993] showed that for carefully designed "trap" functions pair-wise mating will outperform BSC. This was attributed to the high amount of mixing occurring in the BSC algorithm, compared to the delayed commitment shown by the pair-wise operators which enables the testing of complementary middle order building blocks. However it was concluded that "*the unique niche for pair-wise mating is much smaller than most GA researchers believe, and that the niche for 2X* (sic) *is even smaller*"

Work on multi-parent recombination techniques (with fixed numbers of parents) [Eiben et al.,1994, 1995] showed that for many standard test bed functions *n*-parental inheritance (with *n* greater than 2 but less than the size of the population) can be advantageous, although they identified problems with epistasis and multi-modal problems. Two of the operators they introduced for non-order based problems are "scanning crossover" and "diagonal crossover". The former is similar to BSC in performing a new selection process from the pool of *n* potential parents at each locus, thus effectively co-evolving each position. In the absence of any selective pressure p_r (this form is known as "uniform scanning") this will introduce a distributional bias whose strength will depend on the number of parents in the pool (Eiben et al. refer to this number as the "arity" of the operator). Diagonal crossover is a (n-1)-point generalisation of traditional one-point crossover, where sections are read from the n parents successively. This will exhibit positional bias to a degree dependant on the arity.

The idea of co-evolving separate populations of sub-components was used in [Potter and DeJong, 1994] as an approach to function optimisation, with similar findings that the efficiency of the approach is lessened as the amount of epistasis increases, but that if the problem decomposition is suitable then improvements over the "standard" operators are attainable. However the system reported relies on user-specified problem decomposition, and each of the sub-species is optimised sequentially. This leads to questions as to which of the candidates from the other populations to use when creating a full solution for evaluation. Subsequent work has suggested that in fact the choice of partner for evaluation is highly problem specific [Bull, 1997].

These investigations on the beneficial effects of pair-wise mating have been paralleled in the field of Evolutionary Strategies, which have evolved from single parent algorithms to population

based models where recombination plays an important part in the creation of offspring. In [Bäck et al.,1991] a variety of recombination operators are listed which include single parent, two parent and multi parental strategies.

One obvious way of distinguishing between reproductive operators is on the basis of the number of parents involved in the generation of a child Another way is in terms of the way that genetic material is chosen to be exchanged, and the effect that has on the likelihood of sampling particular hyperplanes (schemata). As was seen above, a common analysis is to consider two types of bias, positional and distributional. All of these, will have an effect on the p.d.f governing the way that new elements of the search space are generated and sampled.

On this basis a "reproduction space" can be defined, with axes defined by these three factors, which contains all of the algorithms discussed above. In two dimensions the two types of bias are shown as extremes of the same axis, which is obviously a simplification since some operators (e.g. n-point crossover with n = l or 2) will display moderate amounts of both types. These operators are therefore shown as two linked points. Despite the simplification, this diagram is useful for the purposes of illustration

.

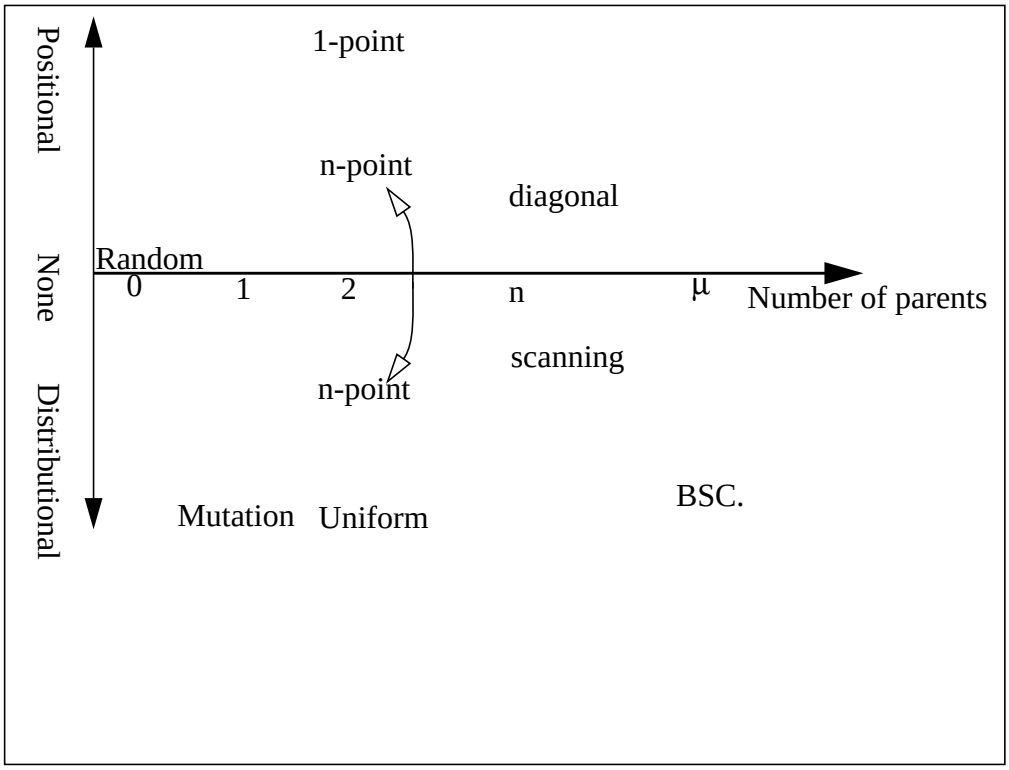


Figure 2: Representation of Space of Reproduction Operators

For any non-trivial new problem it will not be known in advance what the fitness landscape looks like, and it is impossible to decide in advance which is the most effective search algorithm to employ. Furthermore as the search continues it is likely that the optimal search method will describe a trajectory through the algorithmic space as discussed previously. This has been the rationale behind many of the adaptive recombination operators discussed in the previous chapter.

2.4. Gene Linkage

In previous sections an explanation of the operation of GAs was given by the Building Block Hypothesis, which described evolution as a process of discovering and putting together blocks of co-adapted genes of increasing higher orders. The "messy GA" [Goldberg et al., 1989] was an attempt to explicitly construct an algorithm that worked in this fashion. The use of a representation that

allowed variable length strings, and removed the need to manipulate strings in the order of their expression, began a focus on the notion of gene linkage (in this context gene will be taken to mean the combination of a particular allele (bit) value in a particular locus (position)).

Subsequent variants (the "fast messy GA" [Goldberg et al., 1993] and the "Gene Expression Messy GA" [Kargupta, 1996]) rely on first order statistics to identify the blocks of linked genes that they manipulate, as do a number of other schemes e.g. [van Kemenade,1996]. An alternative scheme was proposed in [Harik and Goldberg, 1996], which attempted to co-evolve the positions and values of genes using a representation which consider loci as points on a circle, with the (real-valued) distance between points representing their linkage.

Simply stated, the degree of linkage between two genes may be considered as the probability that they are kept together during recombination. Thus for a problem composed of a number of separable components, the genes in the building blocks coding for the component solutions will ideally exhibit high inter-block linkage but low intra-block linkage.

In the succeeding sections a model of recombination operators based on boolean gene linkage will be introduced, and it will be demonstrated that standard operators can easily be described and generalised within this model. Some linkage learning operators are also described in these terms, and a new learning operator based on the model is proposed.

2.4.1. A Model for Gene Linkage

In this section a model of gene linkage is developed that permits the description of a number of different recombination operators in terms of the way that they manipulate and preserve linkage between loci within an individual. Although the effect of mutation is to disturb allele values, so that locus linkage and gene linkage are not strictly synonymous over evolutionary time, the low rates of mutation used in GAs mean that for the purposes of this discussion this distinction is irrelevant for most of the operators described here, and will be ignored.

This model associates with each member of the population a boolean array A of size l x l which encodes for gene linkage within the problem representation. Recombination is considered in part as a function which acts on these arrays: $R(A) \rightarrow A'$, such that the contents of the parents' arrays after the operator is applied determine which genes in the parents will be transmitted together, and hence which schemata will be preserved. When the allele values from loci in a selected parent are copied into the offspring, the corresponding columns of that parent's linkage array are copied into the array of the child, (even though in many cases they will be completely rewritten by the application of the recombination operator in the next iteration).

For conventional operators a recombination operator will take a fixed number of parent individuals as its inputs, and the same transformation is applied to the linkage array of each (as will be described in the next section). In this case the offspring will initially have the same linkage array as its parents. A number of adaptive algorithms have been proposed in which the linkage patterns are learned, and for these operators the parents will in general have differing arrays, as will the resultant offspring.

A recombination operator is thus defined by its arity, the function R, and a representation of the random choices made by the operator, such as the crossover points in N point crossover, and the choice of parents in uniform or scanning crossover.

To give a simple example, which will be formalised later, One Point Crossover works by taking two parents and selecting a crossover point at random, which effectively splits each parent into two blocks of linked loci. In both parents all of the array elements corresponding to inter-block links are set *true*, and those corresponding to extra-block links (i.e. links "across" the crossover point) are set *false*. In other words the effect of this operator is to rewrite both arrays completely, according to a randomly chosen number (the crossover point).

In this model, each element A_{ij} of the linkage array takes the value *true* or *false* according to whether there is a link from locus i to locus j. In many cases an unambiguous definition of the recombination operators requires a symmetrical array, but this is not always the case. Two loci are

considered to be linked (i.e. remain together under the recombination operator) if and only if both of the relevant array elements are set *true*. For the purposes of this discussion the problem representation will be assumed to be linear.

Having defined the linkage array, it is useful to formally define some terms (using standard discrete mathematical notation), starting with a boolean function that tells us whether two loci are linked.

$$\forall i,j \in \{1, \ldots, l\} \bullet \quad Linked(i, j) = A_{ij} \wedge A_{ji} \tag{D14}$$

One property common to all operators is that since each locus is by definition linked to itself, then the principal diagonal of the array is composed entirely of *true* elements, i.e.

$$\forall i, j \in \{1, \ldots, l\} | (i = j) \bullet Linked(i, j) \tag{2}$$

This may be read as *for all i, j, between 1 and l* ***such that*** *i = j* ***it holds*** *i is linked to j.*

In order to specify a recombination operator, it is also necessary represent the arity, *a*, and the random choices made when the operator is applied. This is done via a randomly created vector *x* of size *l.* The contents of *x* and the way that they are specified vary from operator as will be seen.

2.4.2. Static Recombination Operators

2.4.2.1. N-Point Crossover

For *n*-point crossover operators, the elements of *x* are integers which represent the number of crossover points to the left of the corresponding loci. They are monotonically increasing, with the changes in value corresponding to the randomly chosen crossover points, i.e.

$$\forall i \in \{1, \ldots, l\} \bullet ((i < l) \wedge x_i \in \{0, \ldots, n\} \wedge (0 \leq x_{i+1} - x_i \leq 1)) \vee (x_i = n) \tag{D15}$$

Using definition 15 with $n = 1$ (and by necessity arity 2) allows the specification of One Point Crossover (1X) as:

$$R_{1X}(A) \rightarrow A' | \ \forall i, j \in \{1, \ldots, l\} \bullet \ ((x_i = x_j) \wedge Linked'(i, j)) \vee (\neg A'_{ij} \wedge \neg A'_{ji}) \tag{D16}$$

where the function Linked'() is taken to refer to the array after the operator is applied.

This definition states that two loci are linked if there are the same number of crossover points (0 or 1) to the left of them in the representation. If there are not the same number of crossover points, then according to D16 both of the elements in the array which determine the link are set false. This ensures an unambiguous specification of *A'*.

Setting $n > 1$ in (D15) and increasing the arity accordingly to *n+1, and* using D16 to specify *R* produces Diagonal Crossover [Eiben et al. 1995].

In the case where $n > 1$, but the arity is two (i.e "traditional" two-point and *n*-point crossover) it is necessary to take account of the fact that alternate blocks are also linked. The random vector $\bar{x}$ is still defined as per (D15), and the linkage of alternate blocks is catered for by noting that this equates to the x_i values yielding the same result mod(2) if they are linked. The generalised two-parent *n*-point crossover is defined by (D15) and:

$$R_{2,nX}(A) \rightarrow A' | \forall i, j \in \{1, \ldots, l\} \bullet ((mod(2, x_i) = mod(2, x_j)) \wedge Linked'(i, j)) \vee (\neg A'_{ij} \wedge \neg A'_{ji}) \tag{D17}$$

Although Diagonal Crossover picks *n* crossover points at random and then selects the blocks from *n+1* parents successively, it would be simple to generalise this to any number of parents, so that if the arity *a* is less than *n* more than one block is picked from some parents (e.g. D17 when $a = 2$). If the arity is greater than *n+1*, then this generalisation is still valid as long as more than one child is created per crossover application. Thus the general form of this class of recombination operators where *n* crossover points are picked at random and offspring created from *a* parents is:

$$R_{a,nX}(A) \rightarrow A' | \forall i, j \in \{1, \ldots, l\} \bullet ((mod(a, x_i) = mod(a, x_j)) \wedge Linked'(i, j)) \vee (\neg A'_{ij} \wedge \neg A'_{ji}) \tag{D18}$$

where the vector of crossover points, $\bar{x}$, is defined as per D15.

2.4.2.2. The Uniform Crossover Family of Operators

The case of Uniform Crossover can be represented in one of two different ways. The essence is that at every locus a random choice is made as to which parent the gene should be copied from. Although the more usual case has a 50% probability of selecting from either parent, some authors (e.g [Spears and DeJong, 1991]) have championed the cause of parameterised uniform crossover, so that genes are copied from the first parent with probability *p*. The arity of Syswerda's original version, and most implementations, is two.

The first difference between this family of operators and the *n*-point crossovers above is immediately apparent: in the former *n* crossover points were chosen at random, and the vector $\bar{x}$ filled in accordingly. For the Uniform Crossover family of operators, a set of *l* random values must be chosen. A second apparent difference is that in the former the number of crossover points is known, whereas in the second it may vary between 0 and *l-1*. However it should be noted that since the values are chosen at random from {1...*l*} for *n*-point, the value *n* is in fact an upper bound on the number of crossover points, as they may coincide, and also because a crossover point at either extreme of the representation has no effect.

One representation of two parent Uniform Crossover which makes this distinction clear, simply notes that two adjacent loci will remain linked in both parents if the corresponding random numbers are both either less than or greater than *p*. Using a vector *r* to denote a random vector where each r_i is selected uniformly from [0,1], the elements of *x* are given by:

$$x_1 = 0, \forall i \in \{2, \ldots, l\} \bullet ((r_{i-1} < p) \wedge (r_i < p) \wedge (x_i = x_{i-1})) \vee ((r_{i-1} \geq p) \wedge (r_i \geq p) \wedge (x_i = x_{i-1})) \vee (x_i = x_{i-1} + 1) \quad \text{(D19)}$$

Using (D19) to determine the vector *x*, the function R is then defined exactly as per (D17). It is evident that this model could be used to generate a class of multi-parent operators by taking any values for the arity, *a*, and using (D18) to specify *R*. This class of operators would have a crossover point between adjacent loci with probability $2p(1\text{-}p)$, and hence a mean number of $2p(l\text{-}1)(1\text{-}p)$ crossovers per genome. However for $a>2$ this class is no longer isomorphic to Syswerda's original specification.

An alternative representation, which is more easily generalised to models with arity *a*, such as Scanning Crossover [Eiben et al. 1994], is for the elements of *x* to be drawn from the set {0...*a-1*} according to some probability distribution. This might be uniform, fitness proportionate, or parameterised as above. The generalised representation becomes:

$$\Upsilon_{UX}(A) \rightarrow A' | \forall i, j \in \{1, \ldots, l\} \bullet ((x_i = x_j) \wedge Linked'(i, j)) \vee (\neg A'_{ij} \wedge \neg A'_{ji}) \quad \text{(D20)}$$

Using *a* to denote the arity, the elements of *x* are drawn from {0,...*a-1}* as:

$$x_i = \begin{cases} 0 & r_i \leq p \\ 1 & r_i > p \end{cases} \quad \text{Parameterised Uniform Crossover} \quad \text{(D21)}$$

$$x_i = int(r_i \cdot a) \quad \text{Uniform Scanning} \quad \text{(D22)}$$

$$\zeta_i = j \bullet \sum_{k=0}^{j-1} fit_k < r_i \cdot \sum_{k=0}^{a-1} fit_k \leq \sum_{k=0}^{j} fit_l \quad \text{Fitness Proportionate Scanning} \quad \text{(D23)}$$

This definition of *x* for Fitness Proportionate Scanning is an implementation of the "roulette wheel" algorithm, and the variable *k* is taken to be an index into the set of parents. Other selection methods could easily be implemented with the corresponding changes to (D23).

2.4.2.3. Other Static Reproduction Operators

All of the recombination operators above utilise a new randomly generated vector $\bar{x}$ to transform the parents' linkage arrays each time they are applied. Although the definition of the operator in terms of its effect on the linkage array is constant, the resultant arrays *A'* may be different for each application. By contrast mutation based search and Bit Simulated Crossover employ strategies which

can be modelled as having a completely fixed and identical linkage array for each individual.

Algorithms which employ no recombination select an individual from the population and copy it whole. The effect of this "asexual reproduction" is to treat each individual as having a completely linked genome, i.e. the linkage set consists of every locus. This can be modelled as:

$$\Re_{asexual}(A) \rightarrow A' | \forall i, j \in \{1, \ldots, l\} \bullet Linked(i, j) \wedge Linked'(i, j) \quad \text{(D24)}$$

By contrast, the action of Bit Simulated Crossover is to treat each individual as having a completely unlinked genome i.e. all the entries in the linkage array are false:

$$\Re_{BSC}(A) \rightarrow A' | \forall i, j \in \{1, \ldots, l\} \bullet \neg A'_{ij} \wedge \neg A_{ij} \wedge \neg A'_{ji} \wedge \neg A_j \quad \text{(D25)}$$

2.4.3. Adaptive Recombination Operators

In Section 2.4.2 a number of recombination operators were described in terms of the generation of a set of random numbers which are used to specify a vector *x* which instantiates one application of the operator. For the n-point and Uniform crossover families the specification of *R* is such that the linkage arrays in all *a* parents are completely and identically rewritten according to the contents of *x*, i.e. no linkage information is accrued over time.

However, it has repeatedly and persuasively been argued (e.g. [Goldberg et al., 1989],[Thierens and Goldberg, 1993],[Harik and Goldberg, 1996],) that designers of genetic algorithms ignore linkage at their peril. As was mentioned above part of the rationale behind the development of the messy-ga and its variants was to identify and amass building blocks and intrinsic to this was the notion of linkage.

The messy-ga itself uses a variable length encoding which allows for under and over-specification, coupled with mechanisms such as "competitive templates" for filling in missing parts of a representation, all of which make it hard to analyse in terms of the model above.

The latest variant, the "Gene Expression Messy GA" uses first order statistics to identify linked blocks and reproduction operators which work on these linkage sets. The algorithm maintains linkage sets for each gene in much the same way as described above. The recombination operator works by selecting a number of genes from one parent at random. For each member of this "Exchange Set", (provided certain first order statistical criteria are met), the gene and its linkage set are copied into the opposite parent. The problem of "block conflict" (see next section) is surmounted by not insisting that the linkage sets are coherent. This can be represented by dropping the linkage criterion (14) and allowing each row of the array to be considered separately.

Because the arrays are potentially different in each member of the population, it is no longer sufficient to consider a single array A. Instead A will refer to the "donor" parent, B to the recipient and C to the offspring.Ignoring the statistical criteria, and denoting the Exchange set as X, the rows in the linkage array of the offspring *C*, (and the corresponding genes) are given by:

$$\forall i \in \{1, \ldots, l\} \bullet (((i \in X) \wedge \forall j \in \{1, \ldots, l\} \bullet (C_{ij} = A_{ij})) \vee \forall j \in \{1, \ldots, l\} \bullet (C_{ij} = B_{ij})) \quad \text{(D26)}$$

Unfortunately, the inconsistencies introduced also provide scope for spurious linkage correlations to propagate. Although this algorithm is claimed to guarantee convergence to the optimal solution for problems where the building blocks are of a known order *k*, there are two problems. Firstly, as shown in [Thierens and Goldberg,1993], the population size required to guarantee convergence to the optimal solution scales exponentially with *k*. Secondly, and more importantly for practical implementations, *k* is, in general, unknown. This has the unfortunate effect that if *k* is underestimated, and there is deception of order ~*k* in the problem, then the algorithm is guaranteed to converge to a sub-optimal result.

Another variant of this approach, which explicitly attempts to learn values from the range [0,1] to enter into the linkage array is seen in [Harik and Goldberg,1996]. Again, this uses a dynamic encoding where genes have the form (position, value), and a cut and splice crossover operator. The model used allowed some calculations to be made about the bias of the operator in terms of the linkage distribution, provided certain assumptions about the proportion of optimal building blocks were

made. Unfortunately when run in the context of a "real" GA, i.e. with selection, the algorithm did not display any appreciable learning of appropriate linkage. This was attributed to the effects of convergence. Essentially it was found that using the crossover operator alone as a means of exploring the space of linkages was too slow, since once selection has converged the problem encoding, mixing or linking genes has no effect. As Goldberg put it "all our attempts to evolve linkage without intervention (inversion, crossover, etc.) have failed because the alleles want to converge before the linkage gets tight." [personal communication, June, 1996]. This arises partly because the algorithm was trying to learn real valued linkages, rather than restricting itself to a boolean space.

In the Punctuated Crossover mechanism [Schaffer and Morishima, 1987], self adaption is used to govern the evolution of the linkage array.In their algorithm the problem representation is augmented by the addition of a binary flag between each pair of adjacent loci, which encodes for a crossover point. Transcription starts with one parent, and successive genes (and crossover bits) are copied from that parent until a crossover point is encountered in either parent. At this point genes start to be copied from the other parent, ***regardless*** of whether it coded for crossover at that point. This continues until a full child is created, and a second child is created as its complement.

This is equivalent to associating with each individual a tridiagonal linkage array, of the form:

$$A_{ij} = \begin{cases} 0 & |i-j| > 1 \\ 1 & i = j, j+1 \\ 0, 1 & i = j-1 \end{cases} \tag{D27}$$

The values are subject to mutation at the same rate as the problem bits, which corresponds to the generation of the random vector and the transforming of the arrays.

In order to define the linkage array, *C*, of the first offspring, it is useful to keep a running count of the number of times crossover has occurred so far during transcription. This will be denoted by *n*, and for brevity the parents' linkage arrays will be subscripted by this value. If the parents are *A* and *B*, then $A_{n,ij} = A_{ij}$ if mod(2,*n*) = 0, and B_{ij} if mod(2,*n*) = 1. The genome and linkage array inherited by the first offspring are given by:

$$C_{ij} = \begin{cases} A_{0,\,ij} & (i = 1) \wedge (j = i+1) \\ A_{n,\,ij} & (i > 1) \wedge (j = i+1) \wedge A_{n,\,ij} \wedge A_{n+1,\,ij} \\ A_{n+1,\,ij} & (i > 1) \wedge (j = i+1) \wedge (\neg A_{n,\,ij} \vee \neg A_{n+1,\,ij}) \\ 1 & i = j, j+1 \\ 0 & |i-j| > 1 \end{cases} \tag{D28}$$

Unfortunately this implementation did not preserve much linkage information from either parent: it was demonstrated in the original paper that it was possible for two parents with "medium" numbers of crossover points to create two offspring, such that the linkage array attached to one child contained all the crossover points from both parents.

2.5. Linkage Evolving Genetic Operator

On the basis of the observations made in the previous sections, a number of properties can be identified that a new recombination operator should possess:

1. The operator should be able to adapt its form to the landscape being searched rather than pursuing a fixed search strategy.

2. The operator should be able to traverse the whole space of recombination operators, i.e. it should be able to vary both the bias it displays and its arity.

3. The operator should preserve as much linkage information as possible.

It was shown in Section 2.4 that both the N-point and Uniform families of Crossover Operators can be defined in terms of linkage arrays, with the randomness in each instantiation captured by a random vector *x*. Similarly other adaptive operators can be defined in terms of linkage arrays, with

the random choice of parents and mutation taking the place of *x*. It follows that one way of achieving all of the goals listed above is to encode the linkage array within the genome of each individual and to use self adaptation to govern the growth of different linkage sets, and hence recombination strategies.

In brief, the proposed operator will work by attaching a linkage array to each member of the population, which partitions the loci in that member into a number of linkage sets. The process of recombination then consists of selecting a number of these sets from the population such that between them all the loci of a new individual are exactly specified. The genes specified by those sets, together with the relevant portions of the linkage array, are then copied into the offspring.

The Self Adaptation of the recombination strategies thus proceeds through a process of rewarding successful strategies (by association with fitter members of the population), and generating new strategies from them. New linkage arrays are generated by the recombination process (which is able to produce offspring with different arrays to their parents) and also by the application of mutation to the linkage arrays.

It was demonstrated in the previous section that standard recombination operators, and asexual reproduction, can be expressed in terms of the effect that they have on linkage arrays. The definitions D16-20, 24, 25 completely specify *A*', and state that for two unlinked loci *i* and *j*, A'_{ij} and A'_{ji} are both false. In terms of which loci are inherited from
the parent, a sufficient condition for an equivalent implementation occurs if either one of
the two elements is false. This means that not only would an operator implementing linkage arrays span a wide space of recombination operators, but there is a kind of neighbourhood structure to the mapping from arrays to implementations i.e. similar arrays code for operators with similar arity and bias.

The proposed approach is similar to that of Schaffer and Morishima, but a tighter restriction is placed on linkage inheritance. In their algorithm, transcription will swap from one parent to the second if there is a crossover point encoded on the second, even if the equivalent genes are linked in the first parent. This leads to problems with inherited linkage as noted on page 30. The algorithm proposed here will only copy complete sets of linked genes from a parent. When the next set of genes is chosen for copying, consistency of linkage sets is enforced. This is obviously a tighter restriction, and so the whole population is searched for potential linkage sets. The arity of the operator is thus undefined, and will depend on the sets chosen.

A number of questions immediately arise at this point, such as should the algorithm be restricted to tridiagonal arrays implementing only adjacent linkage (as per Schaffer & Morishima), and should symmetry be enforced on the array.

The first of these questions is by far the most important since it affects the kind of tightly linked blocks that the model will be able to directly represent. At first it might appear that the restriction to adjacent linkage could impose unacceptable constraints, and will tend to exhibit positional bias. There are three major reasons why it was decided to implement adjacent linkage only:

1. Although Tridiagonal arrays can only explicitly code for two loci to be linked if they are part of the same block, it is also possible for two (or more) unlinked genes to be inherited from the same parent if both of the blocks containing them are selected. The likelihood of this happening will depend on the particular mix of arrays present in the population at any given time.

In the previous section it was shown that for most operators the sets of loci which are inherited together are determined by the arity and the choice of a set of random numbers, from which $\bar{x}$ and hence *R* and *A'* are defined. In this case the element of randomness arises from the way in which blocks are selected.

2. It has already been noted that learning linkage is a second order effect, and that there is often a problem with allele convergence preventing the algorithm from finding good "blocks". By only considering adjacent linkage, the size of the problem space is reduced from $O(l^2)$ to $O(l)$.

3. When the practical implementation is considered, problems of conflict resolution are immediately found with the use of non-adjacent linkage. These can be demonstrated by the following

simple example with a population of size 2, where for ease of illustration genes are being picked sequentially from left to right:

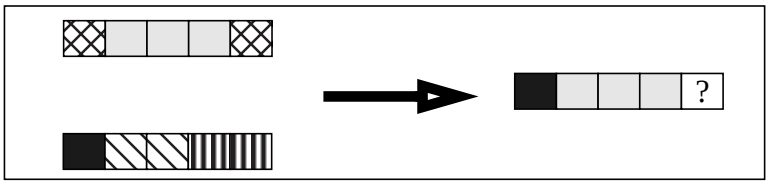


Figure 3: Block Conflict with Non-Adjacent Linkage

In Figure 3 the shading represents the different linkage sets present in the two parents. If the first linkage set from the bottom parent is picked, followed by the second set from the top, loci 1 to 4 in the offspring are specified. There is now a problem, since it is not possible to complete the offspring by selecting a gene for locus 5 without destroying a linkage set. This is because locus 5 is linked to locus 1 in the top parent and locus 4 in the bottom one. In this case, the problem will occur whenever sets from the two parents are mixed, i.e. 50% of the time. Furthermore, consideration shows that the same problems can arise regardless of the order of selecting genes

This means that implementing non-adjacent linkage, would require an arbitration process via which would be decided the parts of the accrued linkage information to be ignored during recombination. Although this is a trivial example, consideration shows that as the population size and length are increased the problem is likely to get worse.

By contrast the restriction to adjacent linkage guarantees completion of any child. Consider any block of linked genes in any parent. Subsequent blocks in that parent will not contain links to genes at previous loci, by virtue of the adjacency constraint. Thus an individual can always be completed by selecting the remaining blocks from the same parent without fear of conflict.

Finally comes the question of whether symmetrical arrays should be maintained. Effectively doing this would mean only considering the lower half of the array. As noted above, Schaffer and Morishima reported that it was possible for the linkage information to be very unevenly inherited in this case. There are also other factors to be considered in deciding on a model.

The most obvious outcome of symmetry is to reduce by a factor of two the number of elements available to be disrupted by mutation. If there is a probability p of mutating a value in the linkage array, then the probability of a set of order n surviving mutation is given by $p_{survival}(n) = (1-p)^n$ for the symmetrical case and $p_{survival}(n) = (1-p)^{2n}$ in the asymmetrical case. This suggests that the action of mutation is to enforce a greater distributional bias on the asymmetrical case, with greater propensity to maintain smaller blocks.

However there is a balancing factor which acts in favour of larger blocks in the asymmetrical case. This is the fact that it is possible to build up blocks during recombination in a way which is impossible for symmetric arrays. This is demonstrated by the following simple example.

Let the array A for a parent a define the linkage sets {1,2,3},{4,5},{6},{7,8,}... and let the array B for parent b define the linkage sets {1},{2,3,4,5},{6,7}....

Imagine starting to create an offspring c by selecting the sets {1,2,3} and {4,5} from a and then the set {6,7} from b. Inspection of the linkage sets shows that one or both of A_{56} and A_{65} must be false, but in general there is a 1 in 3 chance that A_{56} is true, which by definition of the operator holds for C_{56}. A similar argument holds for B_{65} and so C_{65}. Taking these two together there is a 1 in 9 chance that both C_{56} and C_{65} are true, i.e. that the linkage sets of C now look like {123},{4567}...

In practice the probability of blocks joining may be less than this. The sketch above assumed that unlinked loci have an equal distribution of cases of (0,1),(1,0) and (0,0) and that the probabilities in A and B are independent. In a real population these assumptions start to become weaker over evolutionary time, both as a result of the stochastic effects of a finite population and of the effects of convergence (see Appendix A).

However, during the important early stages of the search, this ability to create longer blocks by juxtaposition acts in opposition to the pressure towards shorter blocks arising from the effects of mutation on the arrays. It makes it more possible for the algorithm to develop and maintain longer blocks. This allows the asymmetrical algorithm to explore reproduction strategies with less recombination than is possible for the symmetrical case.

In practice mutation is applied with higher probability to the problem representation than the linkage arrays, and so identical linkage sets may contain different allele values in members of the same population. As evolutionary time progresses the fitter instances will be selected, and subjected to mutation, and so more examples of individuals containing the same linkage sets with different allele values will be evaluated.

It is by this means that the competing schemata (defined over the partition corresponding to the linkage set) are assessed by the GA. The ability to maintain and explore longer blocks in the asymmetrical version therefore allows the algorithm to search higher order schema, and reduces the chances of falling prey to deception.

Finally, both the higher probability of mutation and the ability to create blocks by juxtaposition mean that algorithms using asymmetrical arrays can undergo a much faster evolution of linkage. This turns out to be a benefit since other authors working in the field have experienced problems with linkage evolving at a much slower rate than the problem encoding (see above).

2.6. Conclusions

In this chapter a description was given of recombination in genetic algorithms, and a formalisation was developed to describe various operators in terms of a “linkage array”. Based on this a description was formulated for a new adaptive operator which would combine the desirable properties of self-adaptation with taking advantage of learned knowledge about linkage between genes which many operators discard. The next chapter describes the implementation of this operator.

Chapter Three
Implementation of the Lego Operator.

3. Introduction

In this chapter the implementation of the new recombination operator and its performance on a suite of test problems is described. Its sensitivity to changes in other facets of the GA is investigated, and an analysis made of the recombination strategies evolved on different problems.

3.1. Representation

The Lego operator is implemented by adding two boolean arrays to the representation of each individual, denoting whether each locus is linked to its neighbour on the left and right respectively. For the sake of clarity these arrays have length l, although in practice only $l - 1$ of the positions are used since the problem representations are not toroidal. This gives an extended representation for the population as

$$P^{t} = (a_1^t, \ldots a_\mu^t) \in (I \times I \times I)^\mu \tag{D29}$$

Denoting the three arrays coding for the problem, left linkage and right linkage as α, L and R respectively, an individual comprises the 3-tuple $a_i^t = (\alpha_i^t, L_i^t, R_i^t)$. This will be referred to as the genome of the individual, and a single gene at locus j consists of the relevant entries from all three arrays: $a_{i,j} = (\alpha_{i,j}, L_{i,j}, R_{i,j})$. The linkage array $A(i)$ for the ith member of the population is given by:

$$A_{i,jk} = \begin{cases} 0 & |j-k| > 1 \\ 1 & j = k \\ R_{i,j} & j = k-1 \\ L_{i,j} & j = k+1 \end{cases} \tag{D30}$$

The linkage criterion, (D14), for two genes j and k, (where without loss of generality $j < k$) becomes

$$Linked(j,k) \equiv (j = k-1) \wedge L_k \wedge R_j. \tag{D31}$$

The operator is not restricted to selecting from only two parents, so the genome can be considered as being comprised of a number of distinct linkage sets, or blocks. Unlike the case for two parent recombination, alternate blocks on the same parent are not linked. The linkage set of a locus i is defined as the set of all linked loci i.e.:

$$\begin{aligned} S(i) = \{ j \in \{1,\ldots,l\} \mid & (j = i) \\ & \vee ((j < i) \wedge \forall k \in \{j,\ldots,i-1\} \bullet Linked(k, k+1)) \\ & \vee ((i < j) \wedge \forall k \in \{i,\ldots,j-1\} \bullet Linked(k, k+1)) \\ & \bullet j \} \end{aligned} \tag{D32}$$

When an offspring is created, blocks are chosen sequentially from left to right. In the first locus, all blocks are eligible, so one is chosen according to the p.d.f. p_r from a parent denoted p_1. This block is then copied whole, that is to say that every gene from the linkage set of gene $a_{1,1}$, is copied into the new individual.

If there are $s_1 = |S_{11}|$ elements in this first set, then, by definition of the linkage set, in both the offspring and a_1, $\neg Linked(s_1, s_1+1)$. Providing that $s_1 < l$ there is now a new competition to select the next block.

Because the whole purpose of the new operator is to preserve information carried in the linkage array, the selection of partial blocks is forbidden. Thus the selection of the next parent is restricted to those with eligible blocks. If the selection of blocks from the population is fitness dependant this will involve a dynamic calculation of p_r. to yield a modified p.d.f. p_r'. The p.d.f. for parental selection for the next block will look like:

$$p''_r(a_i^t) = \begin{cases} 0 & Linked(a_{i,s1}^t, a_{i,s1+1}^t) \\ p'_r(a_i^t) & \neg Linked(a_{i,s1}^t, a_{i,s1+1}^t) \end{cases} \quad \text{(D33)}$$

Having chosen a parent p_2 the block S_2 (s1+1) is copied into the new individual and the process repeats until the offspring is fully specified. As noted above, it is guaranteed to be possible to complete an offspring.

The Lego recombination operator has the form $R: (I \times I \times I)^{\mu} \rightarrow I$. Using O_i to denote the ith offspring produced and $O_{i,j}$ to denote the jth locus of that offspring, the full definition of the operator is given by:

$$\Im_{i,j} = \begin{cases} X_{1,j} & j = 1 \\ X_{k,j} & Linked(X_{k,j-1}, X_{k,j}) \quad 1 \le j \le l, 1 \le k \\ X_{k+1,j} & \neg Linked(X_{k,j-1}, X_{k,j}) \end{cases} \quad \text{(D34)}$$

where the parents X_k are selected from the population using the modified p.d.f. $p''_r(a_k^t)$.

Since the linkage arrays L_i and R_i of the population define the recombination strategy, and are themselves potentially changed during both recombination and mutation, the transition function for the recombination function is given by $\Gamma_R(A_i) = MR(A_i)$ R is defined by D34 and the effect of mutation, M, on the linkage is given by:

$$M(A) \rightarrow A' \mid \forall i, j \in \{1, \ldots, l\} \bullet ((|i-j| = 1) \wedge (x_{ij} \le p) \wedge (A'_{ij} = \neg A_{ij})) \vee (A'_{ij} = A_{ij}) \quad \text{(D35)}$$

where p is the mutation rate and x_{ij} is a random number sampled afresh from [0,1].

The effect of enforcing eligibility on the selection pressure will depend on the implementation of the algorithm. With a Generational GA it is common to select into an intermediate population from which parents can be chosen at random, but this will not be possible with a Steady State GA. In order to make the Lego operator consistent an intermediate population was not implemented, rather the fitnesses of competing blocks was taken into account during the selection part of recombination as per (D33). The effect of enforcing eligibility is to create a niching effect at the block level as will be seen later.

In practice, the new operator works by considering the population at any given time as a gene pool comprised of blocks of linked genes defined over certain loci. These blocks potentially vary in size from a single gene to an entire chromosome.

If all of the blocks are of unit length then this reduces to considering the problem as a set of separate populations of alleles (one for each locus) which is the basis of Syswerda's Bit Simulated Crossover.

However by allowing links to form between neighbouring genes, it should be possible for good middle order building blocks to be preserved, allowing the preservation and exploitation of longer schemata. This is shown schematically in Figure 4.

Initially the boolean values in the linkage arrays are set using a biassed random choice of value so as to create a range of different block sizes in the initial population, representing a variety of recombination strategies. It was found that using a bias value of around 80% probability (of *true*) gave a good mix of block sizes in the population.

In order to implement the mutation scheme, there is a small probability of changing the value of a link bit via the application of bit-flipping mutation at a low rate.

In Appendix A, the effects of iterated recombination and mutation are examined. It is shown that for an infinite population, in the absence of any selection pressure with respect to linkage, the system will rapidly evolve to a steady state. This steady state will retain all four combinations of links between two adjacent genes, and will thus retain the ability to adapt to changing circumstances. The

proportion of genes linked at the steady state is entirely independent of the starting conditions, being solely a function of the mutation rate applied. Thus any deviation from this pattern indicates a selection pressure for or against linkage.

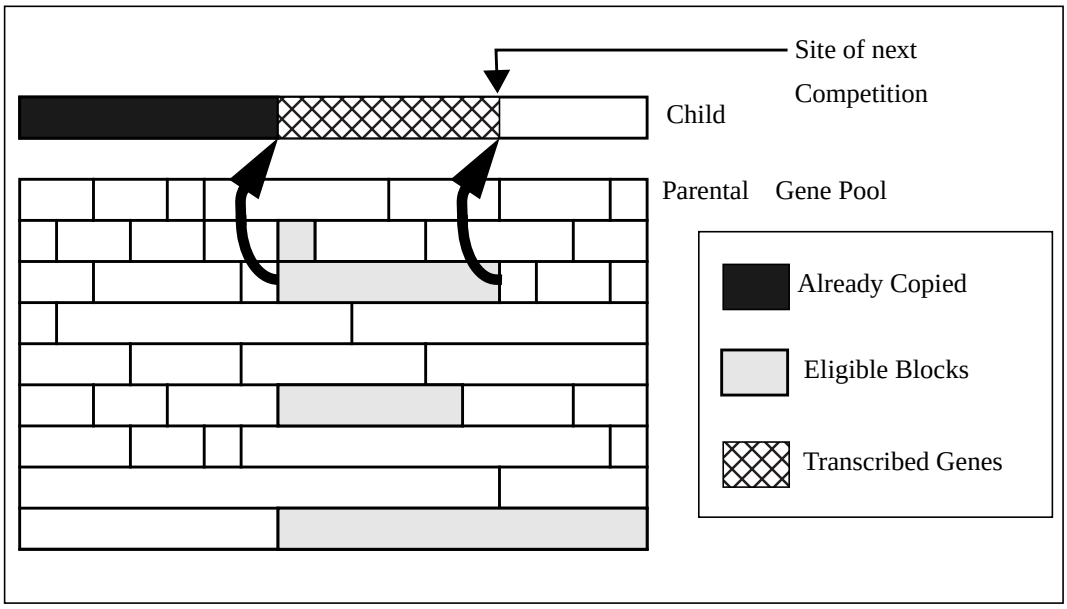


Figure 4: Lego Recombination Operator: 2nd Block Chosen

3.2. Testing the Operator

Having designed the new recombination operator it was decided to test its performance in the context of function optimisation, and compare it with a range of commonly used operators. For these purposes a suite of well studied test problems which displayed a range of epistasis and multi-modality were selected. All of the problems selected had known maximum values, and the algorithms were compared on the basis of the time taken to find the optimum. It should be recognised that there is no single commonly accepted method of comparison, and that very different results might be obtained if a different performance criteria were used.

3.2.1. Experimental Conditions

The linkage evolving genetic operator (Lego) described above was compared against other recombination operators, which represent a diversity of strategies. The operators used were 1-point crossover, uniform crossover (with 50% probability of selecting from either parent), BSC and also Asexual reproduction (i.e. mutation only).

A generational GA was used with a population of size 100 and selection was via a deterministic tournament of size two. In each case the relevant algorithm was run until the problem's optimum had been reached, or a fixed number of evaluations had been done, and the results show the generation in which this occurred, averaged over fifty independent runs (the same sets of seeds were used for each of the algorithms tested).

The one-point, uniform and bit simulated crossover operators were applied with a 70% probability after preliminary experimentation to establish a reasonable application rate for this population size. Lego was applied with a 100% probability.

Experiments were run across a spread of (bit flipping) mutation rates in order to investigate the sensitivity of the various operators to this parameter. For all these experiments the link mutation probability was set to 0.5% per bit.

In addition the new operator was tested with a variety of link mutation rates in order to determine the sensitivity of the operator to this parameter (again mutation here changed the value of a bit rather than picking at random).

3.2.2. The Test Suite

The problems were chosen to display varying amounts of epistasis and deception. They were as follows:

1. OneMax: This was run with a genome of length sixty four. This is a simple hill-climbing problem (the fitness of a phenotype is the number of bits set to one) with no epistasis or deception. It should be ideally suited to Bit Simulated Crossover as each locus can be solved independently.

2. Royal Road: This is a sixty four bit problem effectively identical (given the performance measure) to the R1 function [Forrest and Mitchell, 1992], and similar to Eshelman and Schaffer's "plateau" functions [Schaffer and Eshelman, 1991]. It is comprised of eight contiguous blocks of eight bits, each of which scores 8 if all of the bits are set to one. Although there is no deception in this problem there is an amount of epistasis. This problem was designed to be "ga-easy" but Forrest and Mitchell reported that it had been solved faster by Random Mutation Hill-Climbing. Schaffer and Eshelman reported for their plateau function that *"The crossover gene takes over and the population performance is about the same as with mutation alone but this performance level was reached much sooner"*. With this relatively small population the time taken to solve the problem is heavily affected by the number of the 8 building blocks present in the initial population, especially for the 1-point crossover and so for the latter 500 runs were used. A maximum of 400,000 evaluations were run.

3. Matching Bits: This is a twenty bit multi-optima problem where each pair of adjacent bits of the same parity contribute 1 point to the score. There are thus two maxima at (0,0...0) and (1,1,....1) which score 19. There are a large number of sub-optimal peaks: for example there are 34 sub- optima scoring 18 at Hamming distances from one of the optima of between two and ten.

In addition to being multi-modal, this problem has high local epistasis of order two. The analysis in [Eshelman and Schaffer, 1993] suggests that the difficulty of the problem for the more disruptive operators will depend on the speed with which the search drops one of the two possible solutions. One point crossover would be expected to outperform the other operators on this problem due to its high positional bias. A maximum 150,000 evaluations were run.

4. "Trap" function: This is based on Schaffer & Eshelman's Trap function [Eshelman and Schaffer, 1993] and is composed of ten contiguous sub-problems, each consisting of 5 bits. The score is 2.0 for all five zeroes, 0.0 for a single one, and +0.25 for each subsequent one. There is thus a sub-optima of 1.0 for each sub-problem. The epistasis in this problem is still local but is of a higher order than for the previous problem. Although there are fewer relatively fit sub-optima the deception is higher than for the matching bits problem as for each sub-problem simple hill climbing will lead away from the global optima. Results were given for two point, uniform and BSC when run with the CHC algorithm, which show that two point crossover substantially outperforms uniform which in turn outperforms BSC if the sub-problems are contiguous. However when the sub-problems are distributed the rank order changes from (two-point, uniform, BSC) to (uniform, BSC, two point). This is a natural outcome of the high positional bias of one or two point crossover. A maximum 400,000 evaluations were run.

3.3. Results

3.3.1. Comparison with other operators.

A summary of the results obtained showing the mean number of evaluations taken at the optimum mutation setting for each operator is given in Table 1. The figure in brackets is the mutation rate at which this performance was achieved. A "-" indicates that no runs found the optimum. Figure 5 shows the sensitivity of the operators to mutation rates by plotting mean number of generations to reach the optimum against bit mutation rate for each problem.

It should be noted that for each operator the optimal mutation rate is generally different. Equally, for each function, if the operators are ranked according to the performance at a single mutation rate (as it common practice) the rank orders obtained are highly dependant on the choice of

mutation rate..

Table 1: Generations to find optimum at optimal mutation rate

	Recombination Operator (mutation rate%)				
Function	*1-point*	*Uniform*	*Lego*	*Asexual*	*BSC*
OneMax	32.84 (0.4)	18.5 (0.0)	22.32 (0.6)	89.0 (0.6)	15.06 (0.0)
Royal Road	272.40 (1.2)	335.68(0.6)	275.40 (1.2)	699.0 (0.8)	342.08 (0.8)
Matching Bits	68.64 (5.0)	173.22 (7.0)	89.28 (8.0)	99.3 (5.0)	259.38 (8.0)
Trap	514.2 (0.25)	-	1874.35 (2.5)	3250 (1.5)	-

OneMax: The algorithms using the BSC, Uniform Crossover and Lego find the optimum far faster than those using the other operators, and do so over a far broader range of mutation rates. The results show that there is a correspondence between the speed of finding the solution at a given mutation rate and the ability of the operator to find a solution at high mutation rates. The two operators with no inbuilt positional bias (Uniform Crossover and BSC), which have the potential to solve the problem as a parallel set of sub-problems, do so most efficiently in the absence of mutation, and their performance tails off slowly with increasing mutation rates. By contrast the Lego operator has to learn to lose positional bias, and does not perform well with zero mutation. Inspection of the results showed this was because on some runs with no mutation hitch-hiking caused premature convergence to suboptimal alleles in a few loci.

Royal Road: 1 point crossover and Lego show very similar performance on this function followed by BSC and Uniform crossover. The latter two only work well over a relatively small band of mutation rates. As before the asexual reproduction is far slower and only works effectively over an even narrower spread of low mutation rates.

Matching Bits: Here BSC and Uniform Crossover only perform well over a very tight band of mutation rates, and are not competitive. Again in the presence of mutation the performance of Lego and the optimal operator (1 Point Crossover in this case) are very similar. Asexual reproduction is competitive, but only over a tight band of mutation rates.

Trap function: As suggested above One Point crossover is far superior (both in terms of speed of finding a solution and stability in the face of different mutation rates) to the other operators in this format. Even so, good results were only obtained over a range of 1-3% probability. The algorithm using no recombination only found the optimum occasionally, and at low mutation rates. The Lego algorithm performed better than all the others except 1 Point, but again only over a tight band of mutation rates.

Experiments with the problem representation changed so that the sub-problems are interleaved showed that the performance of the one point operator was dramatically worsened (as found in the incremental version by Schaffer and Eshelman[11]) as was to a lesser extent the Lego operator. The other operators were unaffected, since they show no positional bias.

Overall these results suggest that the Lego operator works well on uni-modal problems whether their epistasis is zero(OneMax) or high (Royal Road), but less well on multi-modal ones.

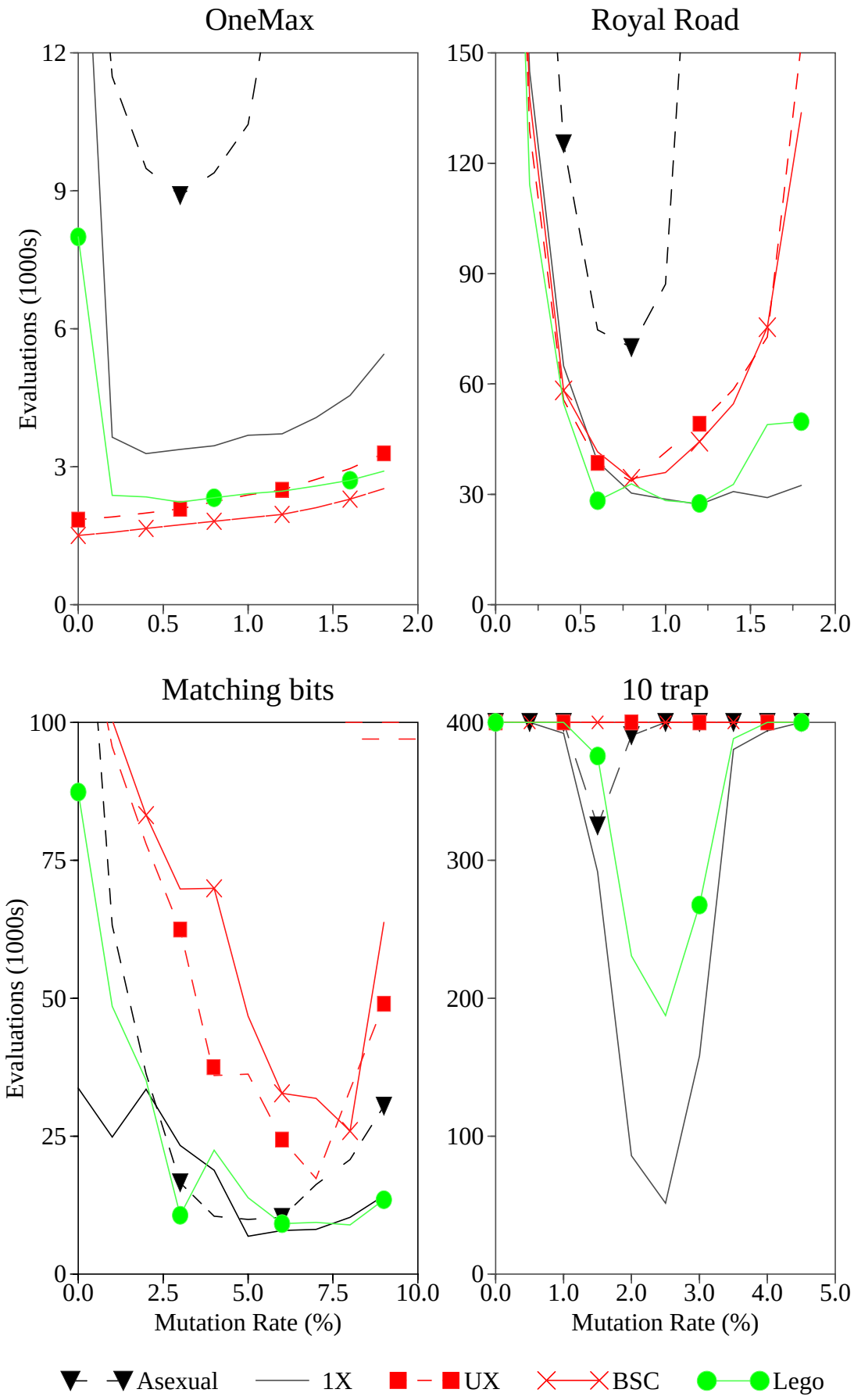


Figure 5: Generations to find Optimum vs. bit mutation rate (%)

3.3.2. Analysis of Operation - Sensitivity to Parameters.

In order to test the sensitivity of the operator to the rate of link mutation it was run with rates of 0%, 0.1%,0.5% and 1%. It was found that over the range of mutation rates there was little difference for the One Max, Matching Bits and Royal Road problems. However the results for the trap function, shown in Figure 6, are instructive.

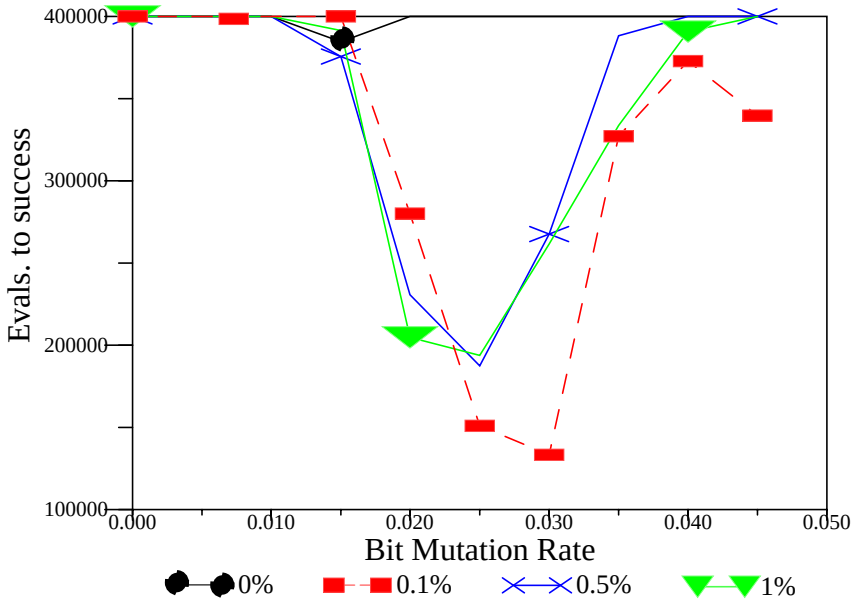


Figure 6: Effect of varying link mutation rate: 10 trap function

It can be seen that the performance is best with a small rate (0.1%) of link mutation but that as the rate increases above this the efficiency drops off.

In the previous section it was found above that this problem was best solved by One Point Crossover, where the high positional bias enables the preservation of partial solutions. For the same reasons, it would be expected that the individuals with large amounts of linking would come to dominate the Lego population.

However for highly linked populations, the primary action of link mutation is to break links between genes. Therefore as the probability of mutation increases, so does the rate at which the offspring created from a highly linked genepool are reduced by mutation into individuals with smaller inherited blocks. If these block boundaries lie within the sub-solutions, then they can be disrupted.This explains why lower rates of link mutation give a better performance.

The question of why the performance is not therefore best with no link mutation at all (where it would be expected that an asexual reproduction strategy would be adopted) is more subtle, but may be explained with reference to Schaffer and Eshelman's findings in [Schaffer and Eshelman, 1991]. They found that for deceptive functions the sub-population with the gene for crossover enabled died out during the initial stages of the search but later flourished when re-introduced by mutation.

Once the population has converged onto a sufficient number of the sub-solutions, then crossover will provide a much faster method of combining these partial solutions than mutation alone can do. If there is an initial selection pressure in favour of linkage, long blocks start to predominate the genepool and the mechanism effectively loses the ability to perform crossover. For zero link mutation there is no way of breaking links between blocks once they are established, and so a largely asexual strategy persists. By contrast, with link mutation at a low level, its effect over time is to disrupt the long blocks that build up, and effectively cause the re-introduction of crossover, allowing the mixing of sub-solutions

The alternative explanation is that there is no selective bias with respect to linkage, and the steady state situation in Appendix A holds. If this were the case the proportion of linked genes would decay to less than 50%.

This can be examined by classifying each individual according to the proportion of its loci which are linked (this is known as its “linkage rate”), and observing the proportion of the total

population falling within various ranges of linkage as the search progresses. Figure 7 shows such plots, as well as the performance of the best of the current generation (for which the maximum value is 100) for link mutation rates of 0% and 0.5%.

These results are the mean of twenty independent runs at a mutation rate of 2.5%.

The first point to note is that both plots show the majority of the population falling into classes with mean linkage rates well above 50%, demonstrating that there must be a positive selective pressure in favour of linkage on this problem.

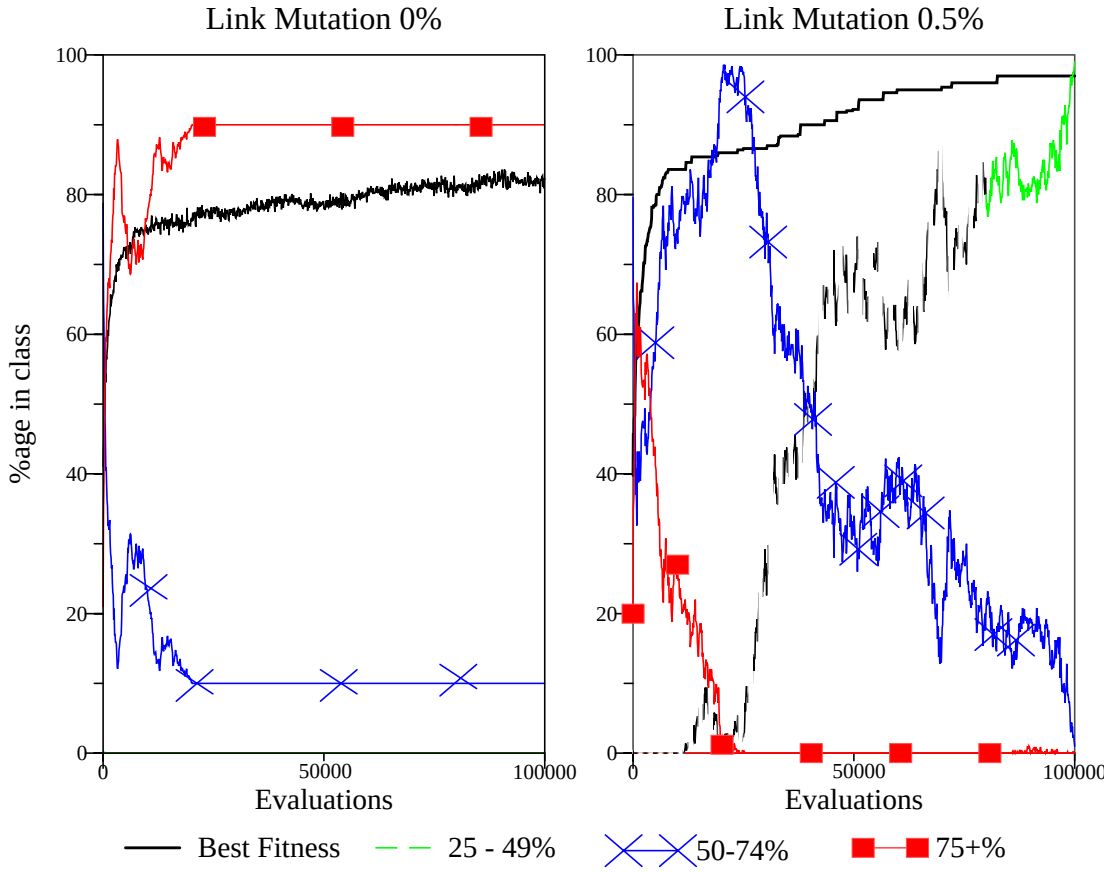


Figure 7: Population Analysis: Trap Function

The plot with no link mutation shows the highly linked population rapidly taking over the search. The search itself starts to stagnate as the safety ratio of mutation decreases and it becomes less and less likely that mutation will create good new partial solutions to the problem without destroying existing ones.

In contrast, the plot for a link mutation rate of 0.5% shows that the sub-population with high linkage initially flourishes and there a very few individuals with low linkage survive. After about 10,000 evaluations, individuals with linkage in the range 50-74% dominate the population, many of which will have been created through the disruption of successfully evolving fully linked individuals. The effect of link mutation is to introduce more and more crossover as the search progresses, with a corresponding rise in the number of less linked individuals. This enables the highest found to keep on increasing, as partial solutions from different individuals (which may well be differently mutated versions of the same ancestor) are combined

Finally it should be noted from Figure 6 that the best performance with no link mutation occurs at a lower rate of bit mutation (i.e. at a lower but less risky rate of search) than when there is some crossover taking place (link mutation rates of 0.1% and 0.5%).

This confirms that recombination serves (at least) two functions. The first is to provide a means of generating new search points via recombination between parents from different regions of the genotypic landscape. The second is to provide a kind of “Repair Mechanism” via the recombination of differently mutated offspring of the same parent. Both are commonly viewed by biologists as being potential pressures behind the evolution of sex (see page 23 for further discussion).

3.3.3. Analysis of Operation - Strategy Adaptation

Classifying each member of the population by its mean linkage, and tracing the fortunes of classes over time can also be used to investigate how the search adapts to different problem types.The proportions of the population falling into a number of aggregate linkage classes (e.g. 0-25% linkage) were measured over the course of the search, and these were averaged over twenty independent runs. For each problem the optimal bit mutation rate (as established by experiments) for a link mutation rate of 0.5% was used. It was decided to show the proportions of linked genes in this manner rather than just plot a mean linkage since it gives a better view of the range of strategies present.

In all cases once the problem is nearly solved, or the population has converged over a significant number of loci, then there is no longer any selective pressure on linking at those loci, and so the amount of linkage will drift to equilibrium under mutation and recombination. From Appendix A we can see that the equilibrium linkage proportion for this link mutation rate is 45.3% For this reason the plots shown in Figure 8 only display the initial parts of the search during which the performance of the best member in the population is increasing significantly

It should be noted that the plots classify individuals according to the number of linked genes as opposed to the number of link bits set in the individuals, whereas the bias in the initialisation is merely on the number of link bits set. Thus although the individuals in the initial population have approximately 80% of their link bits set, they will on average only be 64% linked i.e. those individuals shown with 75-99% linkage have arisen through a positive selective advantage in favour of linking (certain) adjacent bits.

i) For the One Max problem the sub-populations with 25-49% linkage rapidly take over, at the cost of the more highly linked sub-populations. Given a random initialisation, the starting population will have on average 50% of its bits correct, and so under selection alone mutation will on average be deleterious in effect after the first generation. The most successful strategy is one which adapts to utilise more recombination, so that each bit position can be solved separately. The speed at which this happens shows that this is an evolutionary adaptation rather than drift under mutation.

ii) For the Royal Road the sub-population with 50-74% linkage initially dominates, then there is a gradual takeover by individuals in the 25-49% class (although the 50-74% linked sub-population is still not extinct after 400 generations). This demonstrates an initial selective pressure for linkage which decreases over time as opposed to i) where there is strong pressure against and iii) and iv) where there is pressure for linking.

The results in Table 1 show that strategies undergoing any type of recombination drastically outperform asexual reproduction in this setting. If the advantage of crossover was due to the ability to combine separately evolved blocks of eight ones, then it would be expected that one point crossover (and by analogy a highly linked population) would be by far the best strategy, due to its positional bias. However the good results for BSC over a range of mutation rates (see figure 5) suggest that benefits of crossover result as much from the “repair” mechanism suggested above, and that positional bias can cause deleterious hitch-hiking effects.

This ties in with Forrest & Mitchell's results that the problem was most efficiently solved by what they called “Random Mutation Hill-Climbing” (RMHC) rather than algorithms with crossover. RMHC is akin to (1+1) Evolutionary Strategies in which an individual is mutated and is replaced by the resulting offspring only if the latter has equal or higher fitness. This highly conservative selection mechanism avoids Muller’s Ratchet by not allowing deleterious mutations to accumulate. Its reported success, compared to the poor results reported here for asexual reproduction, suggests that important difference is the selection mechanism in RMHC.

iii) for the highly epistatic and multi-modal Matching Bits problem, individuals in the 50-74% linkage class initially dominate the search, indicating a selective pressure towards links between bits. Once the optimum has been found and the population starts to converge (~10,000 evaluations), the mutational drift towards lower linkage accelerates slightly. The initial increase in the number of linked genes in the gene-pool effectively reduces the number of parents contributing to each individual so reducing the likelihood of joining material from a large number of different sub-optima.

iv) For the deceptive trap function the same effect occurs, although dominance of long blocks in the genepool, as represented by the highly linked individuals is more pronounced. Around 35000-40000 evaluations there is a plateau in the peak fitness, coincident with the genepool almost wholly consisting of highly linked individuals. This is followed by a rapid rise in the number of less linked individuals, which is accompanied by a phase of periodic increases in the population's best fitness. This corresponds to the re-introduction of crossover by mutation (as discussed above) allowing the bringing together of sub-solutions from different parents.

During the first stages of the search the two predominant bands are 50-74 and 75-99% linkage. Although calculation of the mean block length is complex, an upper bound can be set by considering the first block in the chromosome. If it is assumed that the linkages are independent and have an equal probability p, the mean length of the first block is given by: $\text{mean} = \sum_{i=1}^{50} i \times p^{i-1} \times (1-p) \approx 1/(1-p)$. For a linkage probability of 75%, this gives an estimated mean block length of 4 bits, rising to 5 for 80% linkage and 6.67 for 85% linkage.

Calculating the mean length of blocks starting halfway along the genome yields equivalent values of 3.978, 4.887 and 6.12.

These values are close to the order(5) of the building blocks which need to be assembled. Further analysis would be needed to determine whether the overall mean block size was less than 5, leading to convergence onto deceptive blocks.

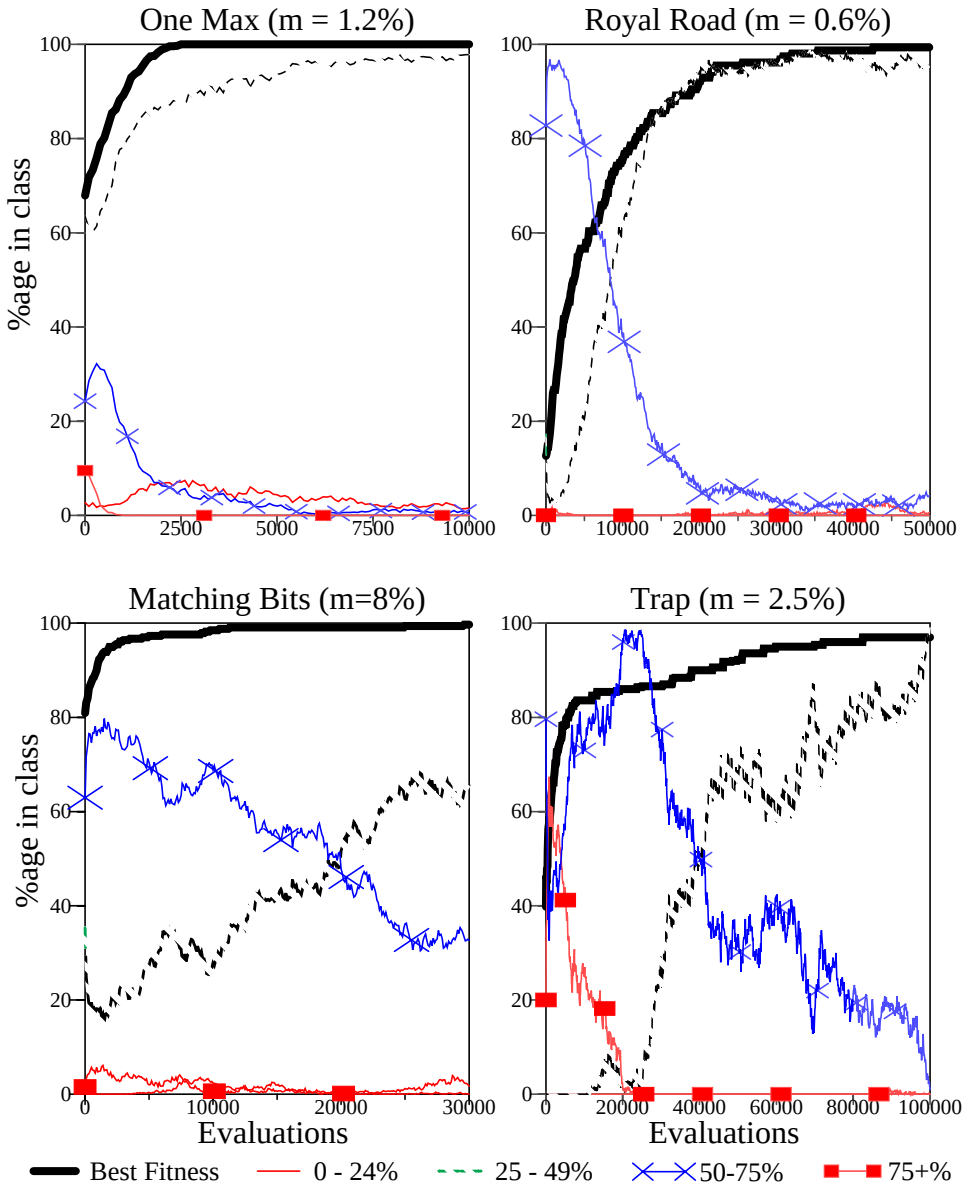


Figure 8: Population Analysis: Test Suite.

3.3.4. Summary of Generational Results

The results above show that in terms of the time taken to find an optimum the Lego operator performs reasonably well. On both of the uni-modal problems used its performance is very similar to that of the best fixed strategy for that problem. This is despite the fact that the best fixed operators are very different for the two problems. For the zero epistasis OneMax problem, BSC shows the best performance, and the linkage analysis shows that the Lego populations adapts to "lose" linkage and adopt a very BSC like strategy with very low positional bias. By contrast the highly epistatic Royal Road problem is better solved by One Point Crossover. This shows high positional bias, and the linkage plots show that there is a definite selective pressure in favour of linkage.

For the multi-modal problems the situation is less clear. Lego outperforms Asexual, BSC and Uniform Crossover on both problems, and is comparable to 1 Point Crossover on the Matching bits problem, but is worse on the Trap function. In both cases there is a strong evolution of highly linked populations, demonstrating a clear evolutionary pressure towards the keeping together of co-adapted genes.

Interestingly other authors using multi-parental or co-evolutionary approaches have identified problems with searching multi-modal landscapes.This may well be a result of the greater mixing effect of multi-parental recombination leading to faster (premature) convergence onto one peak. In [Eiben et al. 1994] it is suggested that poor performance may occur when the parents lie close to different peaks and are not contributing information about the same region of space. The amount of mixing which occurs under these circumstances depends on the distributional bias, which for most multi-parent operators increases with their arity. It has been noted that they work better in a SSGA than a GGA [Eiben et al. 1995]. Similarly Uniform Crossover and BSC, which both exhibit distributional bias, are usually implemented in the less exploratory setting of a SSGA where there is more chance of a long schema being preserved.

It must be emphasised at this stage that the results presented above are for a relatively small test suite, and that only one measure has been chosen as a basis for comparison. In fact using a measure of speed of convergence rather than quality of converged solution should be detrimental to any adaptive algorithm since the space searched is larger, and there will be learning overheads imposed by the adaptation.

The plotting of the results obtained over a wide range of bit mutation rates shows that the operator is on the whole less sensitive to mutation rates than the other operators tested. As has been noted it would have been simple to choose a single mutation rate to report which showed a more flattering gloss. The recombination strategies which emerge are certainly comprehensible in the light of what is known about the structure of the landscapes searched, and appear to match well to the linkage patterns of the most successful "static" operator.

3.4. Steady State vs. Generational Performance

Following the testing of the Lego recombination operator in a Generational GA, and noting the observations above, the performance of the Lego operator was examined in a Steady State GA. As far as possible all the parameters of the GA were kept the same. As before a population of 100 was used, and the two-parent crossover operators were used with a 70% probability. Again parent (block) selection was by deterministic two-way tournament. Initially a "delete-oldest" (a.k.a. Fifo = First In First Out) strategy was used to determine the updating p.d.f. p_u: this yields the same expected lifetime as a generational GA. The same link mutation rate (0.5%) was applied. The results plotted in Figure 9 are the mean of twenty runs.

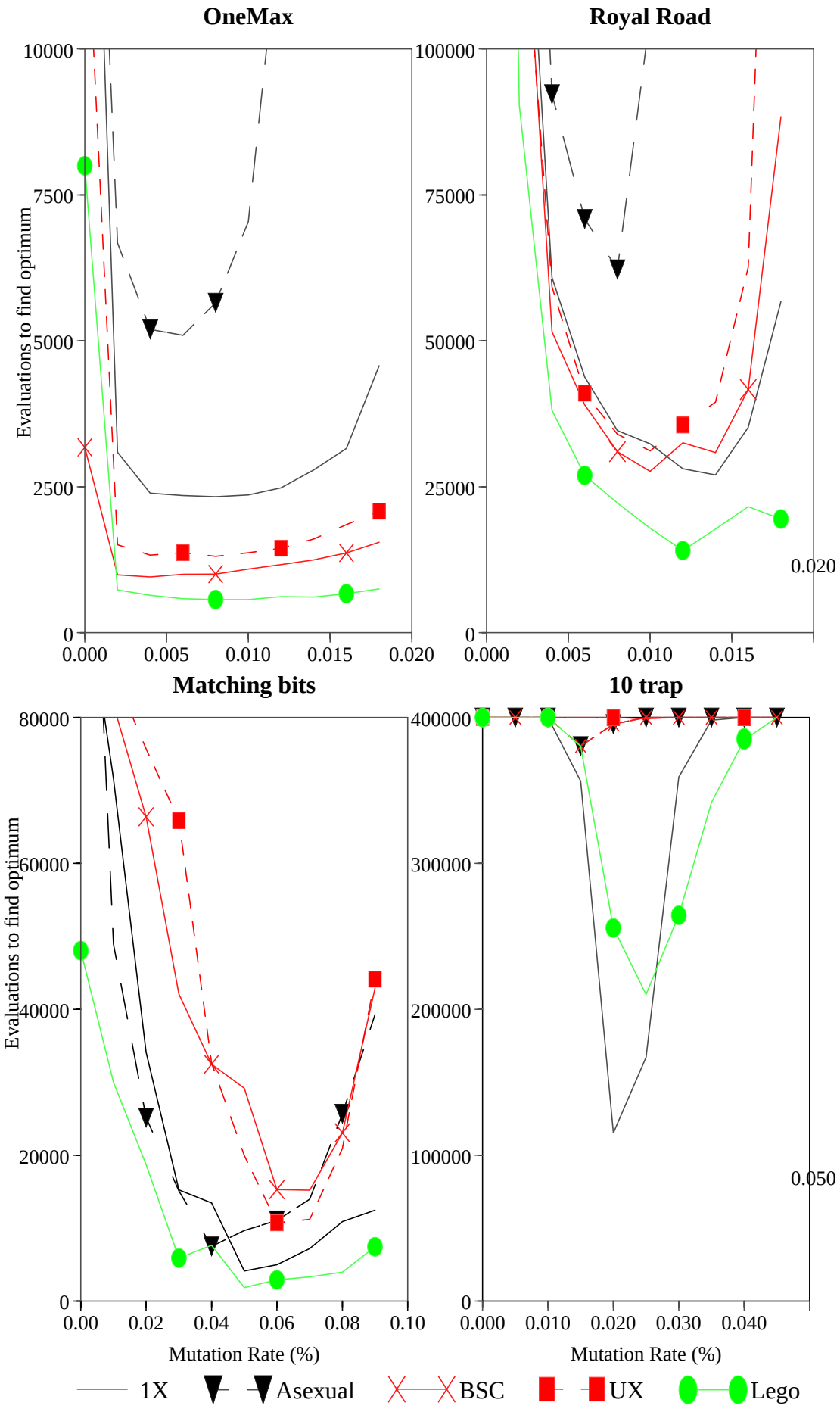


Figure 9: Performance Comparison: Steady State GA.

The results show the relative performance of the Lego operator is improved in this setting. It is now the “fastest” operator on OneMax, Royal Road and Matching Bits problems, and shows less sensitivity to the mutation rate, as long as it is non-zero.

The algorithm with 1 Point crossover is still the fastest at solving the 10-Trap problem, but unlike the generational case, the Lego operator is able to solve the problem over a wider range of mutation rates than 1 Point.

Figure 10 shows a comparison of the Lego algorithm’s performance in the setting of a GGA vs. a SSGA using “delete-oldest” (FIFO) and “delete-worst” (WO). For both GGA and SSGA results are also presented using linear scaling relative to the current generation, according to the formula:

expected number of offspring = 1.0 + (fitness - average) / (average - worst)

The delete-worst strategy (WO) does not perform at all well on any but the simplest problem. This is due to the extra selection pressure leading to premature convergence.

When tournament selection is used, the results for the SSGA are always better than those for the GGA, both in terms of the time taken to find the optimum for any given mutation rate, and of reduced sensitivity to the value of the mutation rate.

When fitness proportionate selection is use with Linear Scaling, there is a less clear distinction between GGA and SSGA results. On the Royal Road and Matching Bits problems the results are very similar, but the SSGA is better on the OneMax and Trap problems. In fact there will be a subtle difference in the selection pressure due to the incremental selection of parents in the SSGA as opposed to the selection of a whole population in the GGA. For the latter Baker’s SUS algorithm was used [Baker 1987]. As for the “roulette wheel” this assigns an expected number of members to each member of the population and then performs stochastic selection to create a parental gene-pool. However the SUS algorithm performs selection without replacement, which gives a more accurate distribution. Because the SSGA performs incremental parental selection, it necessarily performs selection with replacement, which will yield a distribution more skewed towards the fitter members and hence a slightly higher selection pressure.

Comparing the two different selection methods (tournament vs. Linear Scaling with SUS), shows that there is little difference other than on the OneMax problem, where the constant selective differential provided by the tournament is an advantage.

Overall there can be seen to be a clear improvement when Lego is implemented in a SSGA rather than a GGA, a result which mirrors reported findings with other multi-parent and other disruptive recombination operators as discussed above

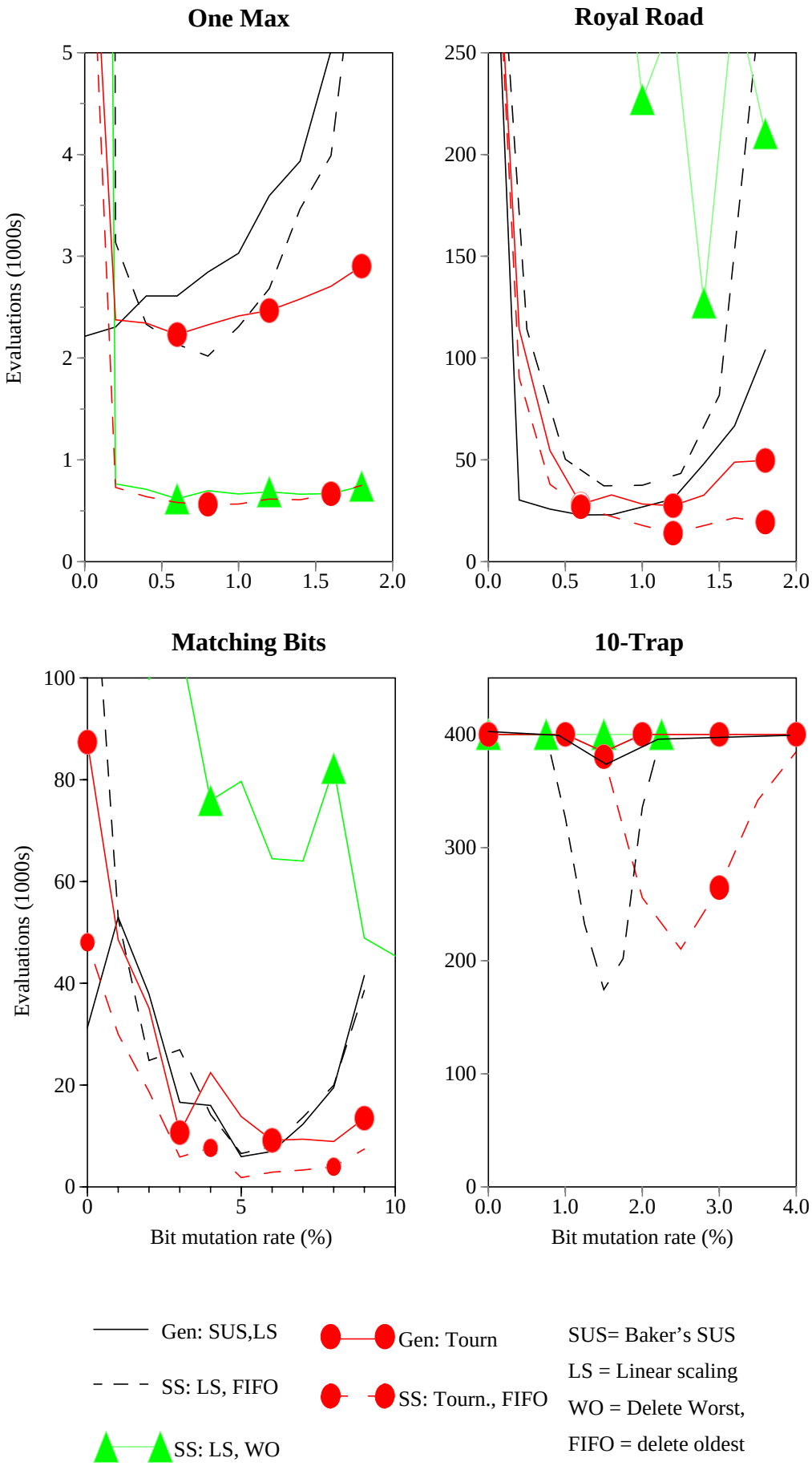


Figure 10: Performance Comparison SSGA vs. GGA, Lego Operator

3.4.1. Population Analysis - Steady State

Figure 11 shows the evolution of the linkage strategies with a population of 100, Fifo deletion and link mutation at 0.5% probability, averaged over 20 runs

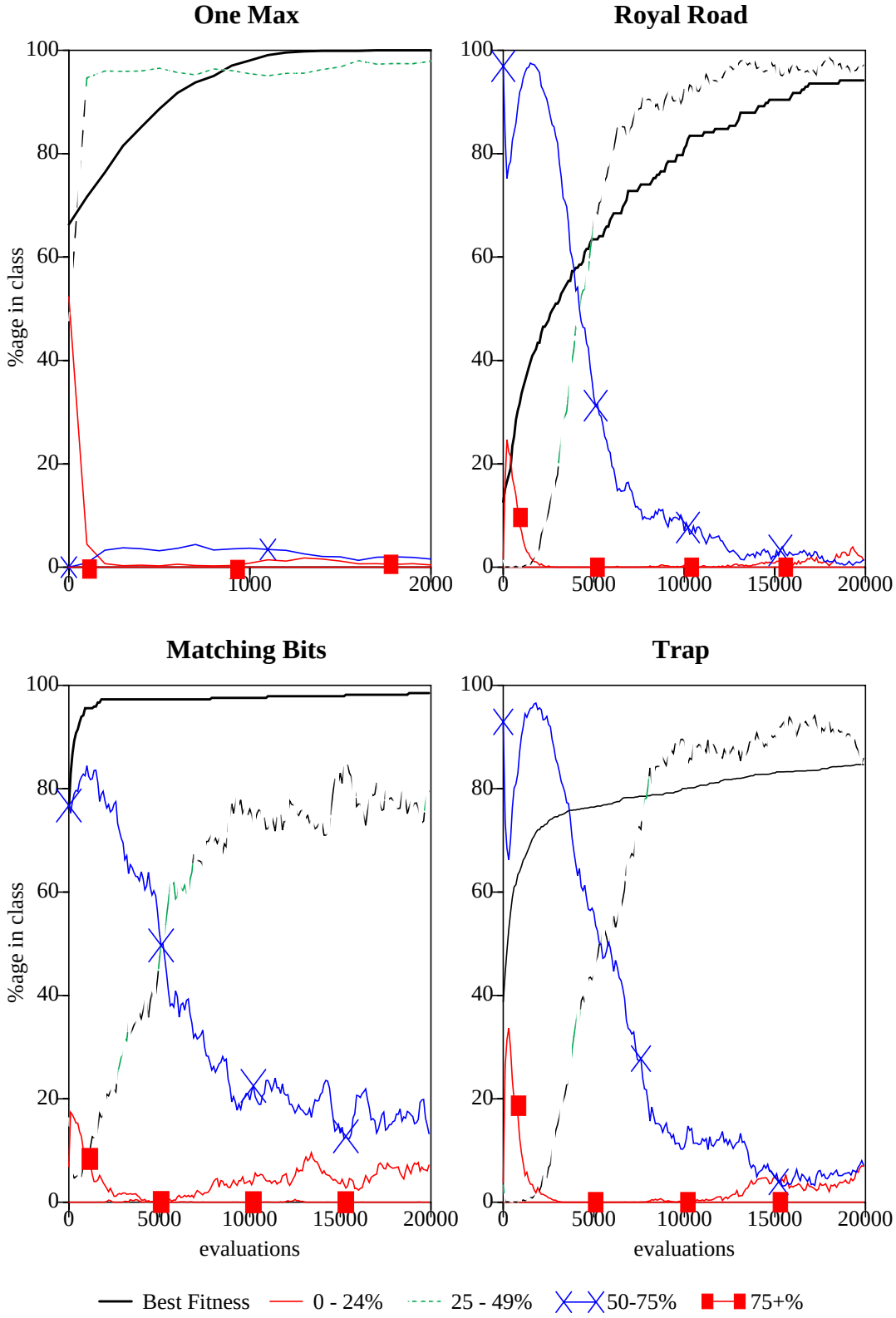


Figure 11: Population Analysis: SSGA 0.5% Link Mutation

Again the same patterns emerge as for the GGA. The OneMax problem is solved by rapid evolution of a highly disruptive recombination strategy. The Royal Road problem initially supports moderate linkage, but then a more successful strategy takes over with more recombination. The same is true, but with a slower takeover by smaller blocks for the matching bits and trap functions.

In Figure 12 the same experiments are run, but with a smaller (0.1%) probability of link mutation. Again the same patterns emerge, but the drift towards lower linkage once the population has started to converge is slower. This fits in with the results found in Appendix A for a single locus.

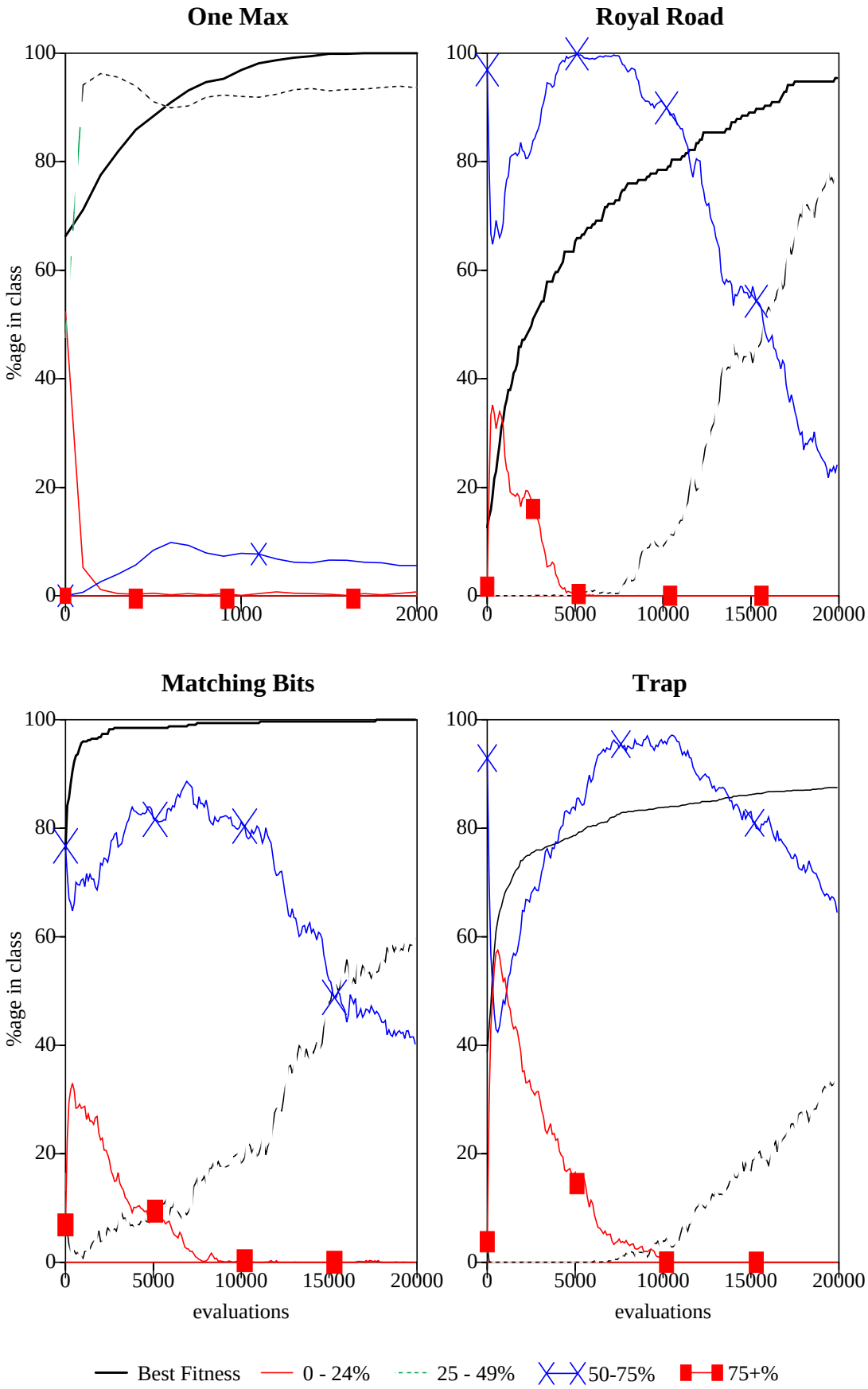


Figure 12: Population Analysis: SSGA 0.1% Link Mutation

3.4.2. Varying the Population Size

The final parameter which might be expected to affect the algorithm is the population size. In Figure 13 the linkage analysis is plotted for twenty runs of a SSGA using a population of 500. All other parameters are as for the previous section.

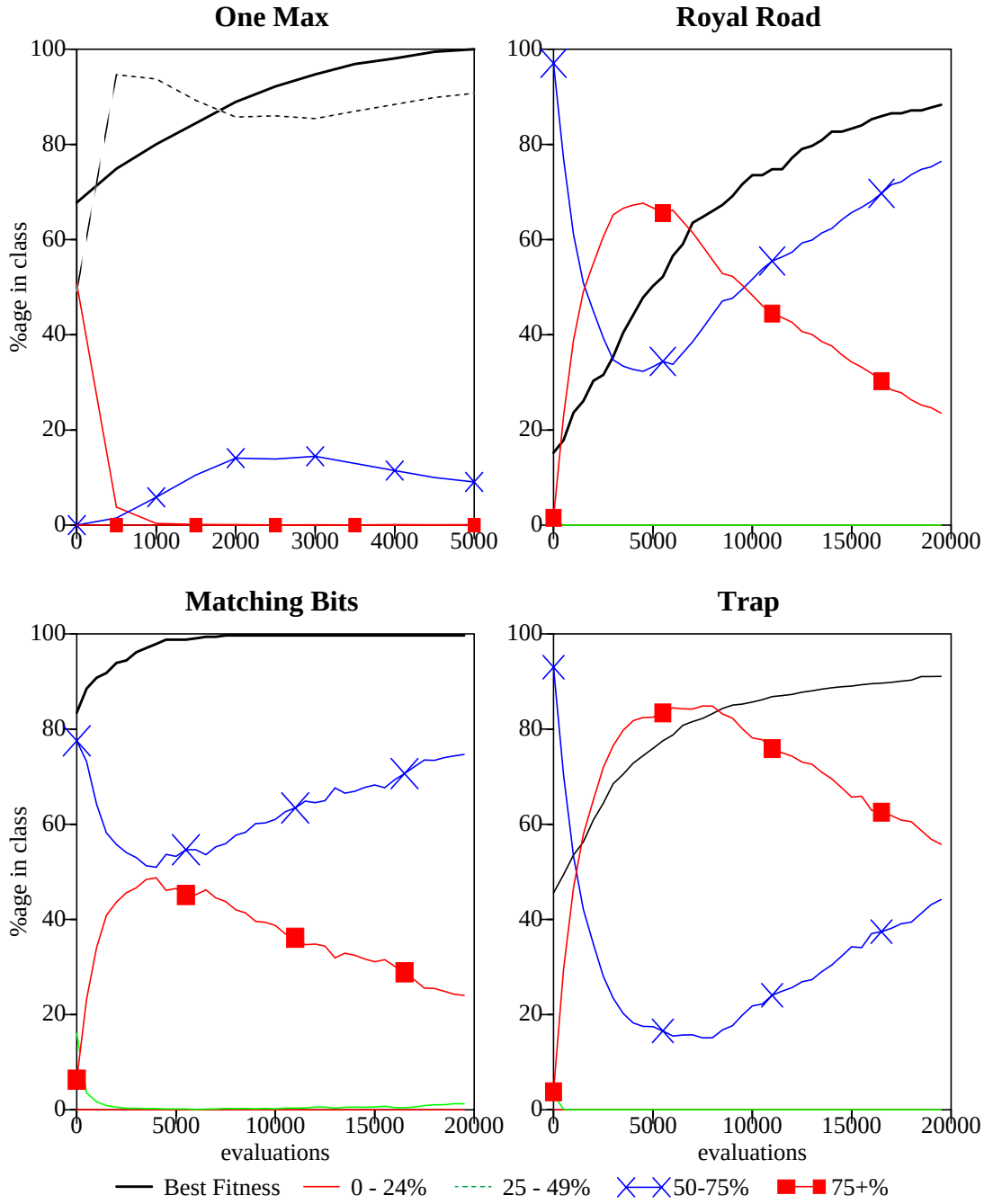


Figure 13: Population Analysis: Size 500, Fifo deletion, Link Mutation 0.1%

These plots demonstrate the same behaviour as before, but the patterns are far clearer. The larger population allows a better estimate to be made of the value of a particular strategy, since there are more likely to be several examples in the population.

This means that there is less chance of a successful strategy being "lost" because the linkage array happens to be attached to a less fit individual. Looking through the plots in the order OneMax- Matching Bits -Royal Road- Trap shows an tendency towards the evolution of more highly linked individuals. This represents an increasing evolutionary pressure towards recombination strategies which are less disruptive of schema and show more positional bias.

3.5. Conclusions

In this chapter the implementation and testing of the new Lego operator was described. Given the performance metric chosen, its behaviour was shown to be comparable to the optimal operator on a set of test problems, although the optimal operator for each problem was not the same. Some decrease in performance was noted on the multi-modal problems, particularly the deceptive Trap function, when compared to One Point Crossover. It was noted that the metric chosen would be less helpful to adaptive algorithms due to the overheads of learning search strategies. Despite this, when tested in a SSGA, the Lego operator gave the best performance on three of the four problems.

Further investigations into the effects of changing the selection and updating methods revealed that the Lego operator worked better in a steady state setting than generational, especially when tournament selection was used. It was noted that other authors working with multi-parent recombination operators have observed improved performance in a SSGA compared to a GGA, and that those operators which exhibit more distributional bias against the preservation of schema with long defining lengths (e.g. Uniform Crossover) are more usually implemented in SSGAs.

An analysis of the behaviour of the algorithm was made by classifying individuals according to their mean linkage, and observing the numbers falling into various categories over time. These experiments showed the operator to be relatively insensitive to changes in other details of the GA. Comparisons with the theoretical behaviour of the linkage in the absence of any selective pressure confirms that some kind of adaptation is indeed taking place.

In order to investigate the phenomenon more fully, and to provide a more systematic basis for change between problems, it was decide to implement Kauffman's NK landscapes and observe the behaviour of the algorithm. This work is described in the next chapter.

Chapter Four
Analysis of Evolution of Linkage Strategies

4. Introduction

In the previous chapter the Linkage Evolving Genetic Operator (Lego) was introduced, and its performance compared to that of a range of common recombination operators in the context of function optimisation. It was found that this potentially disruptive mechanism works better in a "steady state" setting, a result which was also found in work on other multi-parent recombination techniques [Eiben et al., 1995]. This chapter concentrates on analysing how the nature of the search adapts to different types of landscapes.

In the performance comparisons a test suite of well known problems was used, but although they offered a range of different problem difficulties, they did not provide a good basis for systematic examination of the algorithm's performance. In this chapter Kauffman's NK model is used to provide a means of creating and characterising the landscapes searched in terms of the amount of interaction between loci (epistasis). This is described in the next section.

4.1. The NK Landscape Model

Kauffman's NK model of fitness landscapes has been used by a number of researchers to investigate properties of the genetic algorithm e.g. [Hordijk and Manderick, 1995, Manderick et al., 1991]. The abstract family of landscapes defined can be shown to be generic members of the AR(1) class of combinatorial optimisation landscapes. The reader is directed to [Kauffman, 1993] for a fuller description.

In brief the fitness of an individual is considered to be the average of the fitness contributions of its N loci. The contribution from each locus is dependent on the value of the allele at that locus and K other loci, and this "epistasis" is considered to be so complex that it is modelled by creating a look-up table of random fitnesses. This table has an entry for each of the 2^{K+1} possible combinations of values (assuming a binary representation) of bits in that locus and the K other loci on which it depends. Each landscape is thus defined by a lookup table of size $N^* 2^{K+1}$. In the model used here, the value in each cell of the table was an integer drawn with a uniform distribution from [0,1000].

For K=0, there is no epistasis and only one optimum, as there is a single optimal allele at each locus independently. As K is increased changes in the allele value of one locus affect the fitnesses of more and more other loci, and the landscape becomes more and more rugged with increasing numbers of lower optima and decreasing correlation between the fitness of neighbouring points in the landscape. At the limit of K = N -1 the landscape is entirely uncorrelated.

A further refinement can be made by considering the nature of the epistatic interactions. These can be *adjacent* (i.e. the K bits to the right of the locus are considered) or *random* (i.e. for each locus K values are initially defined at random as being the epistatic loci for that position). Note that for the adjacent epistasis the chromosome is considered to be a circle for this purpose, and that when K = 0 or K = N-1, the two types of epistasis are the same.

Some general features of this class of landscapes are listed below:

1. The mean fitness of the optima in a landscape is a function of the degree of epistasis (K) not of the string length N. This appears to hold as long as K is not O(N), and is a more general result than simply being the natural outcome of normalising the fitness contributions from each locus.

2. For K= 0, rank order statistics show that the mean value of the optima is 66% of the maximum possible. Kauffman reports this to hold whatever the distribution used to generate the random values.

3. As K increases, the value of the optima initially increases above the value for K = 0. However above moderate values of K (i.e. ~8-15) the value starts to fall again as a "complexity catastrophe" sets in. Kauffman suggests that there are two regimes, one where K is small with respect to N (K ~ O(1)) and another where K grows with N (K~O(N)). This appears to hold whether the epistasis is local or random.

4. Another measure of landscape ruggedness is the “mean walk length” i.e the average number of steps to a local optimum via one-mutant neighbours (Hamming neighbours). For K= 0, a randomly selected point on the landscape will on average differ in half of its loci from the allele values at the optimum.Thus the mean walk length is N/2 steps. As the number of optima increases, a randomly chosen point will necessarily be nearer to it’s local optimum. Thus for static N, the mean length decreases as K increases, reaching ~ln(N) for K = N - 1.

However for static K, the mean length increases as N increases. This is a result of the fact that the number of sub-optima is predominantly a function of the epistasis, and so increasing the size of the landscape increases their separation.

5. Analysis into the use of fitness-distance correlation as a measure of problem difficulty for GA’s [Jones and Forrest, 1995] shows that the correlation between fitness and distance drops rapidly with increasing epistasis, from -1.0 at K= 0 to 0.0 at K = N -1. In [Weinberger, 1990] an examination was made of the autocorrelation between the fitness of pairs of points and their Hamming distance. It was found that this decreased exponentially, and the logarithm of the rate of change, “the correlation length”, was suggested as a measure of landscape structure. The correlation length decreases rapidly with increasing K. In [Manderick et al. 1991] the correlation length of a landscape is defined precisely as the distance *h* where the autocorrelation function $\rho(h) = 0.5$.

Kauffman provides an empirical investigation of the structure of fitness landscapes by plotting the relative fitness of local optima against their Hamming distance from the fittest found optima for the value of N = 96. He summarises the results as *“for K small - for instance, K = 2- the highest optima are nearest one another. Further, optima at successively greater Hamming distances from the highest optimum are successively less fit.... our landscape here possesses a kind of Massif Central, or high region, of genotype space where all the good optima are located. As K increases this correlation falls away, more rapidly for K random than for K adjacent”* (Kauffman, 1993 pp61-62).

4.2. Experimental Conditions: Analysis

In order to investigate the behaviour of the algorithm the population was scanned after every 20 evaluations, and each individual assigned to one of the nominal classes (0-24%, 25-49%, 50-74% and 75+%) according to the proportion of its genes which were linked. It was felt that using this way of displaying the population’s linkage would provide more graphical illustration than simply plotting the mean population linkage.

The algorithm was run for 10,000 evaluations with both interaction models. The landscapes used string lengths (N) of 16 and 32, and values for K of 0,2, 4, 8 and 15. For each value of K the averages of 20 runs were made, each run with a different landscape.

Following the results in the previous chapter, a SSGA with deterministic tournament selection, FIFO replacement, and a population of 100 was used. Both problem representation and linkage genes were subjected to (value flipping) mutation at a rate of 0.1% probability per locus.

4.3. Population Analysis

4.3.1. Adjacent Epistasis

Figures 14 and 15 show the behaviour of the algorithm on landscapes with adjacent interactions for four values of K on landscapes with N = 16 and 32 respectively. As can be seen the linkage patterns are very similar for the different values of N, the principal difference being that the time scales for adaptation are longer in the larger search spaces.

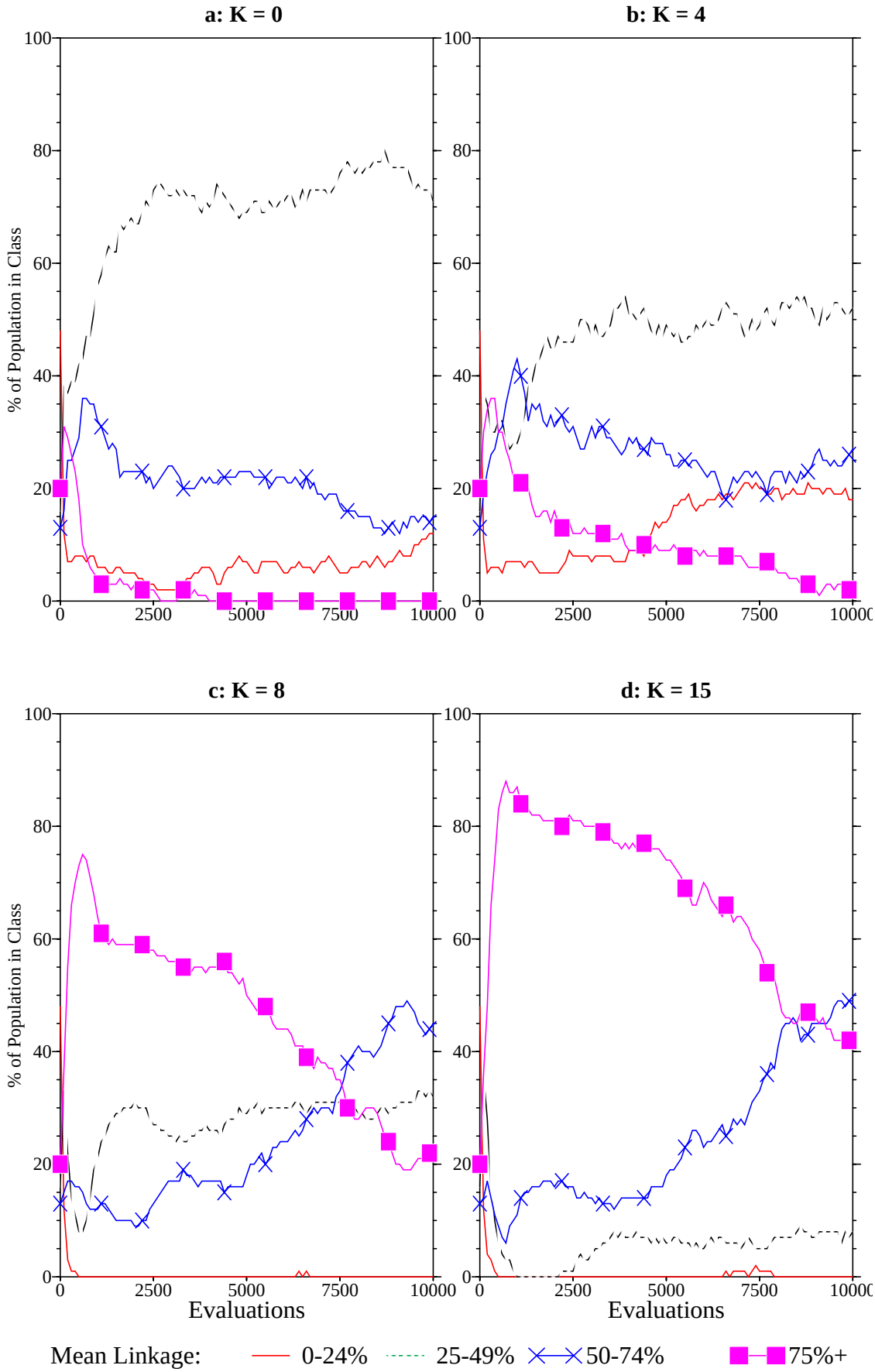


Figure 14: Linkage Evolution with Adjacent Epistasis: N = 16

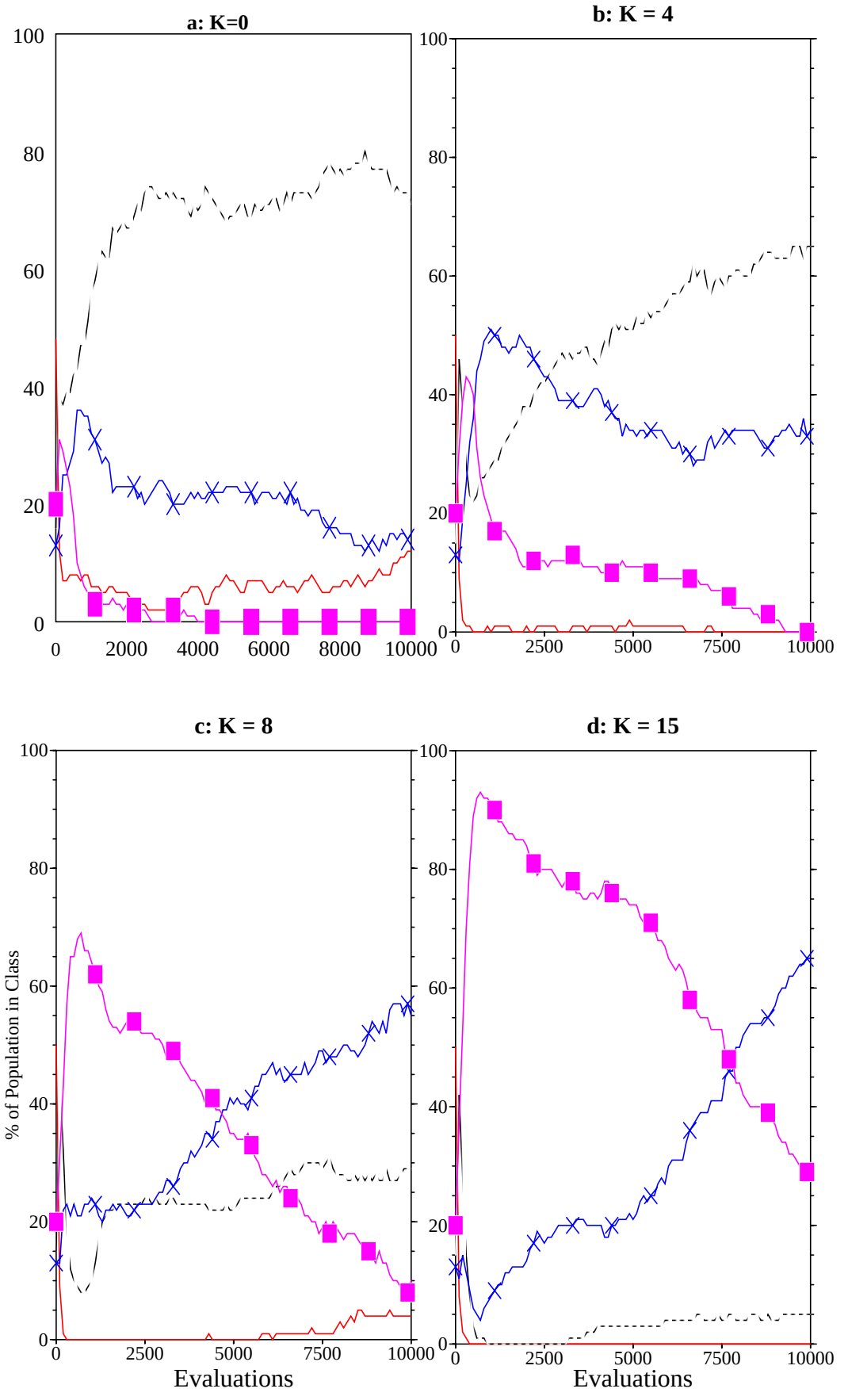


Figure 15: Linkage Evolution with Adjacent Epistasis: N = 32

In each case there is a slight initial “spike” in the number of highly linked chromosomes (the 75+% population), as a result of the “free” links joining together

However, as can be seen from Figures 14a and 15a, for the smooth unimodal problem (K =0), the population is taken over rapidly by individuals with very low linkage, i.e. the gene pool is dominated by small blocks as the problems of optimising the fitness at each loci are completely separable.The decline in the number of long blocks can be explained by the concept of genetic hitch-hiking discussed earlier. Even for blocks which are initially composed of genes which all have optimal allele values, the effect of mutation over time will be to reduce some of these to sub-optimal values. The probability that a block of optimal genes survives mutation decreases exponentially with the length of the block.Thus the hitch-hiking effect is enough to create a selective pressure against linkage.

For the value of K = 4, (Figures 14b and 15b) where the landscape is more rugged, the nature of the population has changed, and the snapshots reveal far more individuals falling into the higher linkage classes. This represents a gene pool composed of longer blocks. This means that the creation of a new individual will involve the recombination of “information” from fewer parents, and will be more likely to preserve long schemata. However recombination still provides an advantage for this type of landscape.

For the very rugged problems with K = 8 (Figures 14c and 15c) and K = 15 (Figures 14d and 15d) the trend of increasing linkage is even more pronounced. The populations become dominated by individuals composed of very long blocks (75+% linkage). Many of these blocks will be incompatible with each other, giving a much reduced, or zero probability of recombination i.e. the search proceeds mainly via mutation. This happens because the landscapes contain so many optima in uncorrelated positions that the chance of the successful juxtaposition of sub-solutions via recombination becomes very small.

To paraphrase Kauffman, the peaks “spread out” as K increases. For a relatively diverse population, there will be potential parents from several peaks. As the “massif central” diverges, so the mean Hamming distance between selected parents from different peaks will increase. Ignoring for the moment incompatibilities between competing blocks in the genepool, (or alternatively assuming a large enough population) the number of parents contributing blocks to a new individual is inversely related to its linkage. Thus, in a diverse population, the linkage in the genepool determines the mean Hamming distance of a new individual generated by the recombination mechanism from any of the points previously sampled by the algorithm.

In short, the more highly linked the population, the less explorative the p.d.f. generated by the recombination operator, and the shorter the “steps” that the operator takes on the landscape away from the parents. However as was noted above, the correlation length of the landscape decreases with K, so this ties in well. At low K, lower linkage enables more mixing to occur, effectively creating children further from the parents. This is a viable strategy since the correlation length of the landscape is relatively long, and so this explorative p.d.f. is likely to generate a point of good fitness. However as the correlation distance falls, then strategies which generate individuals at high Hamming distances from the parents are less likely to be successful compared to those which take smaller steps, i.e. the linkage increases on average.

It should be noted that these arguments only hold for relatively diverse populations in which more than one optima are being explored. As the populations converge to a single optimum, the linkage becomes irrelevant in preserving schemata, and drifts to an equilibrium.

4.3.2. Random Epistasis

Figure 16 shows the population analysis with random epistasis on landscapes with N = 16 and varying K. The pattern of increasing linkage with K is the same as in the case of adjacent epistasis. It should be noted that this holds despite the fact that the nature of the blocks in the Lego model no longer maps directly onto the groups of linked genes defining the blocks of high fitness. This shows that the adaptation is occurring as a response to a more general feature of the landscape, e.g. fitness-

distance correlation.

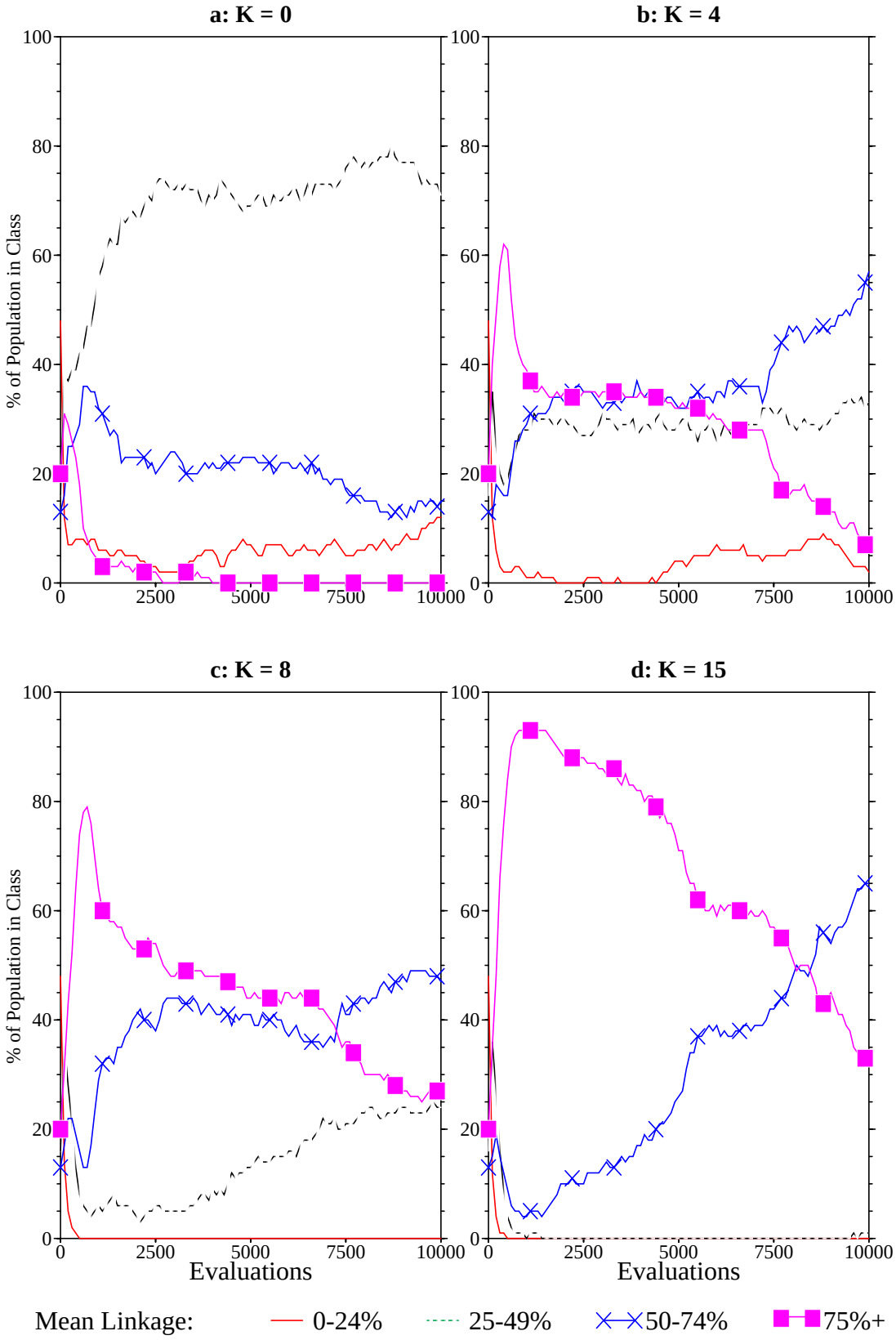


Figure 16: Linkage Evolution with Varying Random Epistasis: N = 16

The second part of the statement quoted from Kauffman was that he observed the correlation between peaks to fall away more rapidly for random interactions than adjacent. The results above show that on both types of landscapes, as the correlation length decreases, so there is an increasing selection pressure in favour of linkage. This leads to a decrease in the numbers of small blocks in the genepool as represented by individuals in the lower linkage classes.

If Kauffman's hypothesis is correct, it should translate into a faster decline in the fortunes of the smaller blocks in the gene pool with random epistasis than with adjacent epistasis. If so for any given value of K between 2 and 15, there would be a noticeable difference between the results for the two different types of interactions. Figure 17 compares the behaviour of the algorithm with the two types of interaction for K 2,4 and 8.

For the K=2 landscapes, (Figures 17a and 17b) the linkage evolution is very similar for both types of epistasis, although there is more linkage preserved in the first 2-3000 evaluations on the random-K landscapes.

As K is increased to 4, the random interactions (Figure 17d) show populations with consistently higher linkage than adjacent interactions (Figure 17c). Unlike the populations on the adjacent-K landscape, those on the random-K landscape show a virtually zero incidence of individuals exhibiting low gene linkage (0-24%). This shows that recombination has proved comparatively less successful on the random-K landscape as a result of the greater distance between peaks. This would fit in with the hypothesis that the peaks are further spread on the random landscape.

For the very rugged landscapes (K = 8, Figures 17e and 17f), both searches are initially dominated by highly linked individuals (long blocks). On the adjacent-K landscape, this proportion peaks at about 80% of the population before remaining consistent at around 60%, before declining rapidly after about 4000 evaluations, with a rise in the proportion of individuals in the 25-49% linkage class to around 30% of the population

By comparison, the highly linked sub-population on the random-K landscapes shows a steadier decline after the initial phase. For much of first 5000 evaluations there is lower (~50%) dominance by the most highly linked class After this the takeover is by individuals from the next lower (50-74%) linkage class. However, the proportion of individuals falling into the lowest linkage class is far lower for the random-K landscapes than for the adjacent-K landscapes throughout most of the time-scale shown.

Overall then, then results obtained back up Kauffman's observations for the cases K=2 and 4, but are inconclusive for the case K=8.

In fact Kauffman's observations are based on a sample of points taken for N=96, where exhaustive testing was not done. He also present results on the mean walk length to a local optima for landscapes of varying N and K and both random and adjacent epistasis. These show that the mean walk length (from a random starting position) is always lower on adjacent-K landscapes for K=8, which suggests that the local optima are in fact more uniformly distributed over the search space for adjacent than random epistasis.

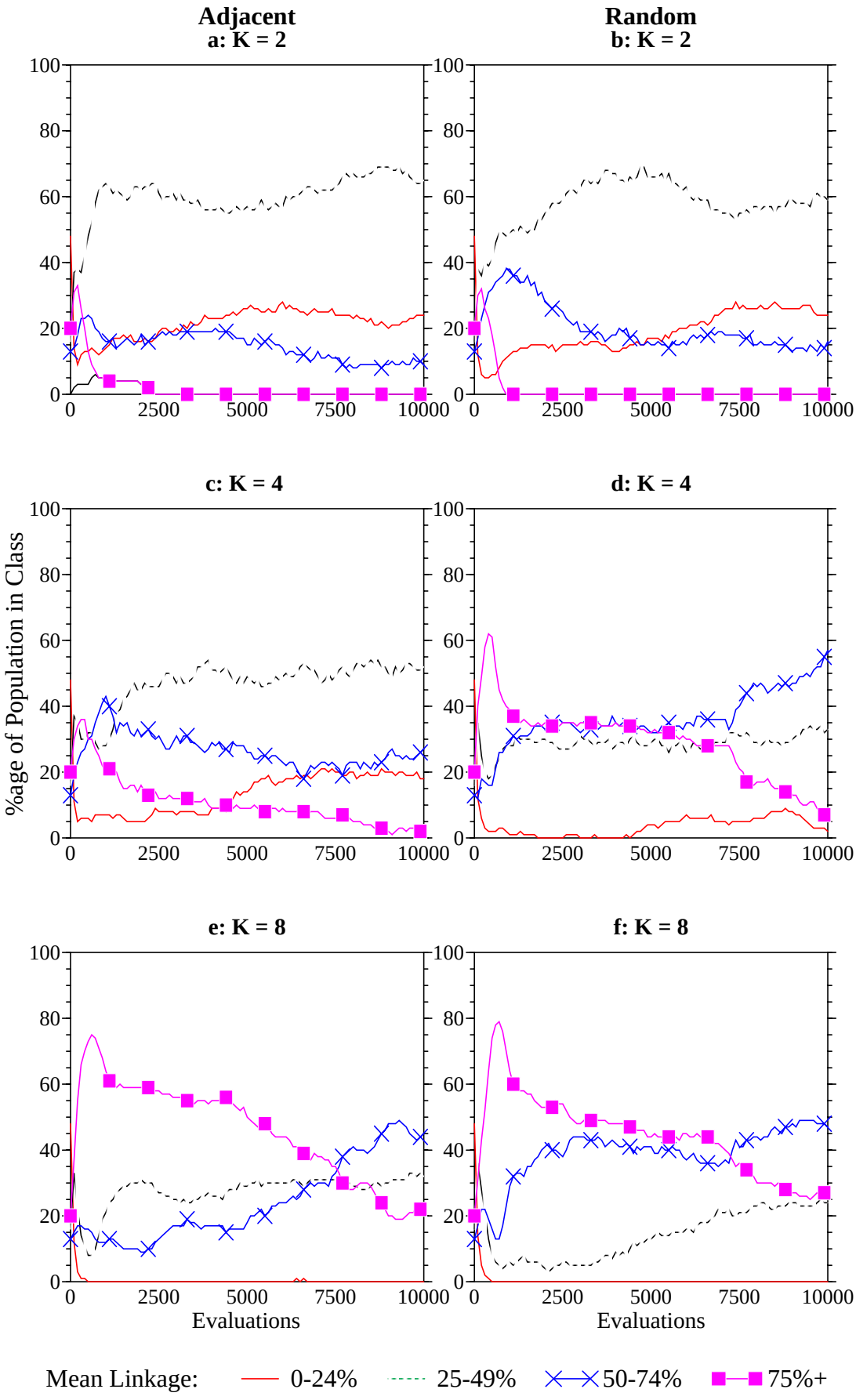


Figure 17: Linkage Evolution: Random vs. Adjacent Epistasis

4.3.3. The Effects of Convergence

In order to investigate the behaviour of the algorithm on the two types of landscapes more fully, the experiments were repeated, but this time the number of parents contributing blocks to the make-up of each new individual was measured as an average over each period of 20 evaluations. This provides an alternative measure of the evolving search strategies since there are two scenarios in which the degree of linkage can be misleading. The first is the case of a highly converged population. The second is the opposite case where an individual may be reproduced whole due to a lack of competition at each of its block boundaries. In this case even a moderately linked population could preserve discrete niches through lack of inter-niche recombination.

In Figure 18, the absolute (total) and distinct (ignoring identical copies) numbers of parents are plotted for K values of 2,4 and 8.

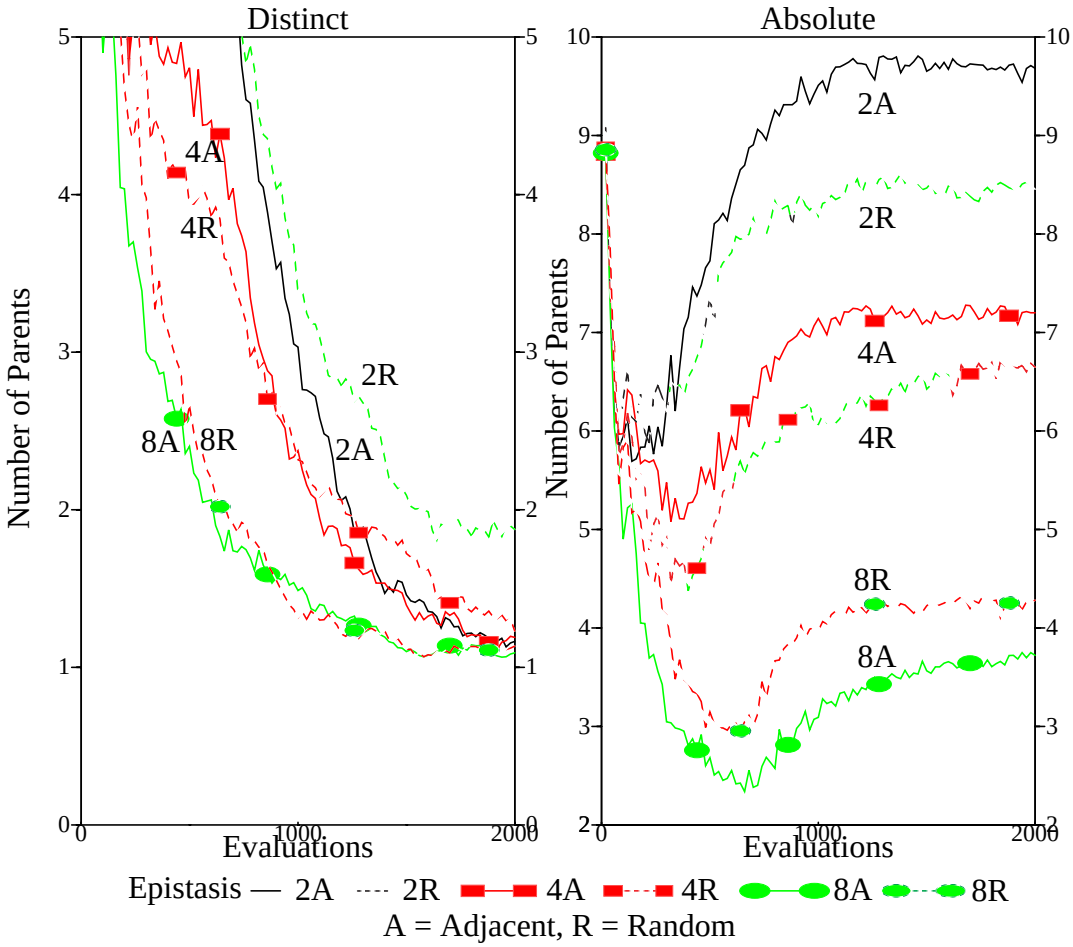


Figure 18: Mean Number of Parents per Offspring

From these plots the following observations can be made:

i) For both types of landscape the absolute amount of recombination decreases with increasing K - i.e. as K increases so longer linked blocks of co-adapted genes dominate the genepool as noted above.

ii) In all but one case (the exception being K = 2 Random Epistasis) the effect of converging (sub) populations is that the distinct number of parents is reduced to one i.e. as Hordijk & Manderick noted the second phase of the search is mutation led.

iii) For both types of landscapes the rate at which this happens increases with K.

These observations tie in with the first part of Kauffman's findings on the structure of the landscape and observation (iii) ties in with the theoretical result that the mean walk length to a local optima decreases as K increases since this implies that populations using an asexual strategy will be able to locate and explore local optima more quickly.

iv) For K= 2 the absolute number of parents involved in recombination is lower on the random-

K landscapes (i.e. there is more gene linkage as noted above) but the distinct number is higher, falling after 2000 evaluations to 2 parents as opposed to 1 on the adjacent-K landscape. Despite this the "mean score" curves (not shown) are very similar. This shows that on the random-K landscape the algorithm has maintained sub-populations on at least two optima. This ties in with Hordijk & Manderick's findings that in fact the "central massif" for the case of K = 2 contains more peaks of a similar height close to the global optima in the random case than in the adjacent case.

The higher linkage in the random case indicates that there are more larger blocks in the genepool, which increases the probability of incompatible recombination strategies, and allows the maintenance of diverse solutions for longer. This effect is a kind of "implicit niching", and is the result of the enforcement of maintaining block consistency.

v) For K = 4 the absolute number of parents involved in recombination is again consistently lower in the populations adapting on random landscapes, i.e. the amount of gene linkage is again higher. However this difference is much lessened.

vi) For K = 8 there is a different situation. The numbers of distinct parents involved are very similar in the two cases, falling rapidly to 2 and then declining slowly as the population converges. In both cases the absolute number of parents (inversely related to mean block size) falls rapidly, but then climbs once the population has converged. However for this value of K the absolute number of parents climbs to a higher value for "random" than "adjacent" landscapes. This means that although the recombination mechanism evolved in each case is effectively sampling the same number of parents, the number of blocks picked is greater for random epistasis. This means that in terms of schema bias analysis [Eshelman et al., 1989] the recombination strategies which evolve via gene linkage show less positional bias for random epistasis than for adjacent epistasis.

Although clarity prohibits the display of the mean scores of the algorithm running on the two different types of landscapes, they are very similar, and the mean fitnesses after 2000 evaluations are not significantly different (using Student's t-test at the 5% level).

These experiments have been repeated with a variety of selection mechanisms and very similar results have been obtained in each case. As noted in the previous chapter, comparisons on the model within a function optimisation setting showed it to be fairly insensitive to changes in mutation rates.

4.4. Performance Comparison with Fixed Operators

Having investigated the evolving behaviour of the Lego operator, on NK landscapes, it was decided to run some performance tests against fixed recombination operators. One reason for this is that the evolution of "appropriate" strategies is of little consequence if the search is ineffectual compared to fixed operators. By using a standard set of landscapes and seeds, comparisons can (and will) later be made to find the relative significance of, and interactions between, different operators.

For the two landscape sizes N = 16 and 32, 10 landscapes each were created for the K values 0, 4, 8 and 15. On each landscape 10 runs were made with different seeds, giving 100 runs per NK combination. Early experimentation showed a mutation rate of 1/len to provide good results for all of the operators tested, which were Lego, 1-point Crossover and Uniform Crossover. A population of 100 was used with deterministic two-way tournament selection and elitist FIFO replacement.

As the optimum value is not known for a randomly created landscape (and is in general not known for many optimisation tasks) the principal performance measure used was the best value found in a fixed time. This was set at 20,000 evaluations for N = 16 and 100,000 for N = 32, which gave most of the runs time to converge.

For each run the value of the best member in the final population was noted and the mean of these values, along with the F values obtained by a 1 way Analysis of Variance are shown in Table 2 below. The F value for 95% confidence of a significant difference between groups is 3.00 in this case, so the differences in performance between the three operators are only significant for the N 32, K 15 landscapes.

For the cases where N = 16, an exhaustive search was made of each landscape in order to discover the global optima. The number of runs in which this was discovered is shown in brackets.

Table 2: Recombination Comparison: Mean of Best Values Found

Recombination Operator	*N = 16*			*N = 32*		
	K = 4	*K = 8*	*K = 15*	*K = 4*	*K = 8*	*K = 15*
1-Point	750.48 (95)	784.34 (54)	775.80 (31)	762.87	782.46	746.25
Uniform	750.08 (89)	783.00 (55)	776.77 (33)	766.64	783.06	725.08
Lego	749.721 (88)	784.355 (56)	777.34. (31)	762.37	780.22	730.48
F-value	0.0109	0.2478	1.6098	0.4051	0.9824	51.2889

The N=32, K=15 landscapes has very high epistasis, and will be characterised by lots of local optima and low fitness-distance correlation, which were identified in the previous chapter as causing difficulties for lego and other multi-parental operators. The performance ranking between One Point and Uniform crossover suggests that the linkage evolves sufficiently to allow the preservation of some high order schemata, but that in some partitions convergence to lower order sub-optimal schema occurs before the linkage strategy has evolved.

Inspection of the number of times the optima is found on the N=16 landscapes shows little difference between the three operators, and a clear pattern of decreasing frequency as K increases. For K= 0 the optima is always found, and for K=4 it is found with about 90% reliability. In fact even for N =16, K=15, which are a set of random uncorrelated landscapes, the global optima is still found more often (~31% of runs) than would be expected by a random search (26.3%). When the Lego algorithm was allowed to run for a further 20,000 evaluations, the number of times the optima was found increased to (100, 95,64,48) for K = (0,4,8,15) respectively. The relatively low increases tie in with Hordijk's observation that the latter stages of the search are mutation led.

It should be emphasised that the GA used was not "tuned" in any way for this performance comparison, and that only one mutation rate was investigated. Many authors have suggested measures such as prevention of duplicates, and the use of higher selection pressures coupled with higher mutation rates. These might yield improved performance, especially on the highly complex landscapes where the more exploitative p.d.f.s associated with higher mutation rates will provide a more thorough search of the space. However, the results demonstrate that in the setting of a very standard GA, for all but one of the problems the choice of recombination operator makes no significant difference in terms of the quality of solution found.

4.5. Conclusions & Discussion

The analysis of the behaviour of the algorithm shows that the recombination strategies which evolve using this model correspond closely to what would be expected from theoretical and empirical investigation into the structure of the fitness landscapes concerned. In general it can be seen that as the degree of correlation of the fitness landscape decreases, so there is an increasing tendency for longer blocks of linked genes to take over the genepool. Since individuals created from large blocks will necessarily involve fewer (absolute) parents than those created from smaller blocks, they will be less likely to combine information from a number of different peaks. In terms of schemata, there is a reduced distributional bias against high-order schemata as the linkage increases. Conversely if there is low epistasis and high fitness correlation then smaller blocks will take over the genepool as individuals created from these are less likely to suffer from the deleterious effects of genetic

hitchhiking.

However analysis of the strategies evolved over the course of the search also demonstrates that the most successful recombination strategy is not simply a function of the landscape being searched but of the distribution of the population over that landscape.

For a relatively diverse population, the amount of distributional and positional bias exhibited by the recombination operator vary according to the degree of linkage. However as the population converges (or niches) such that most offspring are created by recombination of material from just a few different parents, then the distributional bias becomes independent of the degree of linkage. In this situation the linkage evolves to reflect the success of strategies showing different amounts of positional bias.

An obvious potential criticism of this model is that it is geared around links between adjacent bits, and that on problem representations which do not follow this pattern it may perform poorly, and be unable to adapt to the structure of the landscape. However the results obtained with random interactions show that this is not the case. This shows that the adaptation of recombination strategies is triggered by something more than simply identifying and keeping together co-adapted genes.

It has been suggested that this is an adaptation to match the mean Hamming distance of individuals from their parents to the correlation length of the landscape. This reflects a view of the search in terms of the p.d.f. governing the sampling of new points in the space.

An alternative perspective, in terms of schema theory, is that the degree of positional and distributional bias exhibited by Lego varies according to the linkage of the genepool. The evolution of high linkage reduces the distributional bias of the operator, and allows the algorithm to preserve and discriminate between higher order schemata.

On problems where the restriction to adjacent linkage would be expected to be a problem (i.e. high K) successful individuals are created without the extra overhead of a mechanism to allow links between any two loci. This is because with a nearly converged population the gene linkage is able to adapt according to the utility of positional bias in recombination. This illustrates the relationship between the recombination function *R* and the updating function *U*.

Comparisons with other crossover operators showed that the differences in performance (for the metric of best value found) were not significant on any but the most complex landscapes.

An advantage to using the new algorithm is that the evolved linkages can be viewed as an extra source of information about the search space. Examination of the evolution of linkage (with either type of epistasis) shows that there are distinct patterns which can be used to give a guide to the nature of the landscape on which evolution is taking place. The evolution of recombination strategies with low linkage suggests a highly correlated landscape, and high linkage implies a relatively uncorrelated landscape. This “evolutionary feedback” on the nature of the search space will be discussed further in a later chapter.

Chapter Five
Self Adaptation of Mutation Rates

5. Introduction

In this chapter an investigation is made of the use of self-adaptation of mutation rates in a Steady State Genetic Algorithm. The rationale behind the use of Self-Adaptation was discussed in the first chapter.

There are two major reasons for investigating the use of a SSGA. The first of these is that the overall plan is to incorporate the various operators developed, and it was found that the Lego operator worked better in a Steady State setting (although "standard" crossover operators are used throughout this chapter). Secondly, part of the reasoning behind the development of self-adaptive techniques is to facilitate the use of Evolutionary Algorithms on real problems. In practice the real world is a constantly changing environment, and a major class of these problems can be characterised by temporal variations in the fitness landscape. It has been shown on a number of problems that SSGAs are far better than GGAs at tracking moving optima [Vavak and Fogarty, 1996], hence the concentration on SSGAs. As will be seen, although self-adaptive mutation has been successfully incorporated into GGAs, its use in an incremental setting requires the addition of an inner GA, which forms a useful way of adding local search to the GA.

5.1. Background.

Mutation has long been regarded as a vital ingredient in evolutionary algorithms, and some paradigms e.g. both Evolutionary Strategies and Evolutionary Programming [Fogel et al. 1966] use it as their principal search mechanism. Within the field of Genetic Algorithms there has been much work, both practical e.g. [Schaffer et al., 1989] and theoretical e.g. [Spears, 1992] on the relative merits of mutation as a search mechanism. As has been discussed earlier, much of the work has been concerned with finding suitable values for the rate of mutation to apply as a global constant during the search. There have also been a number of approaches suggested for changing the mutation rate on-line.

This issue has been tackled successfully within both Evolutionary Strategies and Evolutionary Programming by encoding the mutation step applied within the representation of each solution. This approach also means that the mutation rate is now governed by a distributed rather than a global rule (see [Hoffmeister and Bäck, 1991], or [Bäck et al. 1996, Section C 7.1.] for a good overview of the issues tackled and approaches taken). These ideas have been applied to a generational GA by adding a further 20 bits to the problem representation, which were used to encode for the mutation rate [Bäck, 1992b]. The results showed that the mechanism proved competitive with a genetic algorithm using a fixed (optimal) mutation rate, provided that a high selection pressure was maintained. The most successful method used created λ ($> \mu$) offspring from the fittest μ parents (such that other λ-μ less fit members of the population have zero probability of being selected as parents) and is referred to as (μ, λ) "extinctive" selection.

In this chapter an investigation is made of the issues confronted when this paradigm is implemented within the setting of a SSGA, where the methods used in Evolutionary Strategies for updating the population are not suitable. The class of landscapes chosen to investigate the behaviour of the algorithm is the well-studied NK family of landscapes, as described in Section 4.1. For the same reasons as before, the measure chosen to study the performance of the algorithm in various flavours is the best value found, averaged over a number of runs and landscapes.

The performance of the preferred version of the operator is also compared to that of otherwise identical algorithms using a number of fixed mutation rates which are widely quoted and recommended in GA literature.

5.2. The Algorithm

The SSGA is different to the generational models used by Bäck in that there is typically a single

new member inserted to the population at any one time, which according to the standard ES nomenclature would correspond to a (μ+1) strategy. However it has been shown empirically (and theoretically for certain simple problems [Schwefel, 1981]) that to optimise the convergence velocity within a (1+1) ES mutation step sizes should be adapted so that there is an "acceptance" ratio of approximately 1:5.

This heuristic means that one individual should be incorporated into the population for every five created and is achieved by increasing/decreasing the mutation step size according to whether the proportion of successful offspring created is more/less than 1:5.Similarly Bäck found it necessary to use "extinctive selection" in his GGA implementation, whereby only the fittest fifth of each generation were used as parents.

The need for extra selection pressure can be explained by considering repeated mutation of a single individual as a form of local search. In [Bull and Fogarty, 1994] a (1+1) non-adaptive local search mechanism was added to the GA and found to improve performance on a number of problems. Since the adaptive algorithm is effectively performing two parallel searches (one in problem space and one in algorithmic space) it can be readily seen why this might be beneficial. The use of just a few parents (a result of the high selective pressure) effectively allows the updating mechanism to make direct comparisons amongst mutation strategies, by comparing many offspring generated from the same parent.

As a simple example of this, imagine a (1,λ) algorithm climbing a unimodal landscape. When the phenotype initially reaches the top of the hill, it will have a non-zero mutation probability. Of the λ offspring created at the next time step, there will (allowing for stochastic variations) be a correlation between the encoded mutation rates and the distance moved from the summit.Thus the member selected to form the next generation is likely to be the one with the smallest mutation rate, and this process will continue until a zero (or close to zero) mutation rate is reached.

However, there is a trade off between this local search and global search (either via recombination or simply by considering other members of the population) which will affect both the convergence velocity and (possibly) the quality of the final solution. The optimal value for this trade off will depend on the relative ability of the different reproductive operators to generate fit new offspring. This will be a factor of (among other things) the correlation length of the landscape.

The approach taken here is to encode the single mutation rate to be applied to each individual within its genetic material, using a binary string of length *l'*. The algorithm works by creating a single individual via recombination and then "cloning" that individual a number of times. The transition function is then applied to the mutation rate encoded in each of these offspring. The offspring then undergo the mutation process and are evaluated before one is selected (according to some policy) and inserted into the population (according to the deletion strategy used). Effectively this inner loop can be thought of as a (1, λ) generational GA, and the algorithm may be formalised as:

$GA = (P^0, \delta^0, O, \mu, \lambda, l, l', F, R', M, \Gamma, p_r, p_s)$ where (D36)

$P^0 = (a_1^0,..., a_\mu^0, m_1^0,..., m_\mu^0) \in I^\mu \times I'^\mu$ Initial Population (D37)

$I' = \{0,1\}^{l'}$ Mutation Encoding (D38)

$O^t = (a_1^t,..., a_\lambda^t, m_1^t,..., m_\lambda^t) \in I^\lambda \times I'^\lambda$ Offspring. (D39)

$R' : \quad (I \times I')^\mu \times \delta \rightarrow (I \times I')^\lambda$ (D40)

R' is the modified version of the recombination operator *R*, which creates one offspring by recombination from the population and then copies it, i.e.

$$\forall i \in \{1, \ldots, \lambda\} \bullet O_i^t = R'(P^t, \delta^t) = \begin{cases} R(P^t, \delta^t) & i = 1 \\ O_1^t & 2 \le i \le \lambda \end{cases} \quad \text{(D41)}$$

The transition function Γ works by decoding each of the mutation rates m_i to get a value in the range [0,1] and then applying bit-wise mutation to each of the elements in the encoding with that probability. A function $D: \{0,1\}^{l'} \rightarrow \Re_+$ is used to do the decoding.

$$O'_i = \Gamma(O_i) = (a_{i\,1},..., a_{i\,l}, m'_{i\,1},..., m'_{i\,l'}) \text{ where } m'_{ij} = \begin{cases} 1 - m_{ij} & X \le D(m_i) \\ m_{ij} & X > D(m_i) \end{cases} \quad \text{(D42)}$$

X is drawn from [0,1] separately for each j, using a uniform random distribution.

This transition function forms the basis of the self-adaptation, since there are now λ offspring with identical problem encodings, but with different mutation rates attached. The mutation operator M is now applied to each of the offspring individually.

$$O''_i = M(O'_i) = (a'_{i\,1},..., a'_{i\,l}, m'_{i\,1},..., m'_{i\,l'}) \text{ where } a'_{ij} = \begin{cases} 1 - a_{ij} & X \le D(m'_i) \\ a_{ij} & X > D(m'_i) \end{cases}$$ and X is defined exactly as before. (D43)

The two functions Γ and M are effectively the same, only applied to different parts of the individuals. Because they are applied sequentially the values used to decide whether to bit-flip ($D(m_i)$ and $D(m'_i)$) are potentially different.

Setting all these operators together defines an iteration of the algorithm as:

$$O''^{t}_{i} = M\Gamma R'(P^t, \delta^t) \qquad \forall i \in \{1, \ldots, \lambda\};\ P^{t+1} = U(P^t \cup O''^{t}) \quad \text{(D44)}$$

where the updating function U is given by the p.d.f. p_s, which will depend in part on the result of applying the fitness function to each of the new offspring.

It can be seen from the above that the following parameters will affect the algorithm:

1) Deletion Policy: Two standard policies are frequently used with steady state GA's, namely Delete-Worst and Delete-Oldest. In addition there is the issue of whether a member of the population should be replaced only if it is less fit than the offspring which would replace it (Conditional Replacement) or always (Unconditional Replacement).

2) Selection Policy: Parental selection (p_r) is via two way deterministic tournament. Selection of an individual from the offspring to enter the population can use the same mechanism or be deterministic, i.e. the best offspring is always picked.

The combination of deletion policy and offspring selection policy will define the p.d.f. p_u which governs the updating function U.

3) Recombination Policy: What type of crossover is used, and should the mutation encoding be subject to crossover?

4) Mutation Decoding Function (D): The genotypically encoded mutation rate must be decoded and then scaled onto the range 0-100%. Three types of encoding are used, namely binary, gray and exponential. In the latter (suggested in [Kaneko and Ikegami, 1992]) the mutation encoding is first binary decoded to give a value j and then the mutation rate m is given by $m_j = \text{maxposs} * 2^{(j - jmax)}$ (where $jmax$ is the largest number allowed by the binary encoding).

5) Number of offspring (λ): This will affect the balance between local search and global search in the early stages of the run before the population has converged.

5.3. Implementation Details

In order to test the effects of the above factors a set of standard values had to be adopted which could be kept constant during the testing of other factors. Since empirically derived standards (e.g. the 1/5 heuristic) existed for some of the factors, and Bäck's previous work in a generational setting had shown that the selection pressure was probably the most important factor, experiments were run in the order shown above, with the optimal (or most robust) combination of parameters from one set of experiments being carried through to the next set.

All experiments were run using a population of 100 on NK landscapes with N = 16, and K values of 0, 4, 8 and 15 to represent a spread of problems with increasing complexity from a simple unimodal hill (K = 0) to a randomly coupled landscape (K = 15). All experiments were averaged over fifty runs, each run being on a different landscape. For equivalence the same fifty seeds and

landscapes were used with each of the alternatives under comparison. The runs were continued for 20,000 evaluations.

5.4. Results

5.4.1. Selection/Deletion policies

A variety of policies for determining the insertion of new individuals into the population were tested, using one point crossover (at 70% probability), 16 bit gray encoding for the mutation rates and a λ of 5.

The results of the more successful methods are shown in Figure 19. Not shown are the curves for the combination of a delete-oldest policy with unconditional acceptance of an offspring, which showed very erratic behaviour and failed to reach optima on any but the simplest problem (K = 0). The following observations can be drawn from the plots:

1. The combination of “delete-worst” with deterministic offspring selection (WC-B / WU-Best) performs well on the simpler problems, and there seems to be little effect whether the offspring is accepted conditionally or unconditionally. However the search stagnates on the most complex landscape at a level below that found by other policies with a less intense selection pressure. This indicates that the higher selection pressure is causing premature convergence to sub-optimal values.

2. The combination of “delete-worst” with stochastic offspring selection (WC-FP) performs worse than the deterministic selection policy. The use of stochastic offspring selection reduces the selection pressure in the algorithm, which explains the relatively slower growth curves, but noticeably the runs also converge to lower optima as the complexity of the search space increases (i.e. as K increases - this is especially clear for K = 8). The reduced offspring selection pressure allows the introduction of greater diversity via mutation, but this is not enough to counteract the twin converging effects of recombination and high replacement selection pressure.

3. The use of a “delete-oldest” policy is only successful if the insertion of an individual is conditional on its being better than the member it replaces. Even with the conditional acceptance policy, the overall selection pressure in the algorithm is much less than for the “delete-worst” policy, and this is reflected in reduced growth curves. The relaxation of the selection pressure also highlights the difference between the stochastic (OC-FP) and deterministic (OC-Best) selection policies, with the former showing very slow improvements, although the searches do not stagnate on the more complex problems as the algorithms with “delete-worst” policies do.

4 Overall the best policy is replacement of the oldest of the population with the fittest of the offspring, conditional on the latter being the fitter of the two (OC-B). This is effectively a tournament of size λ+1 between the offspring and the oldest in the population.

This policy shows growth curves comparable with the other policies on the simpler problems (K = 0,4,8), but on the most complex problem it significantly outperforms all others, reaching much higher optima.

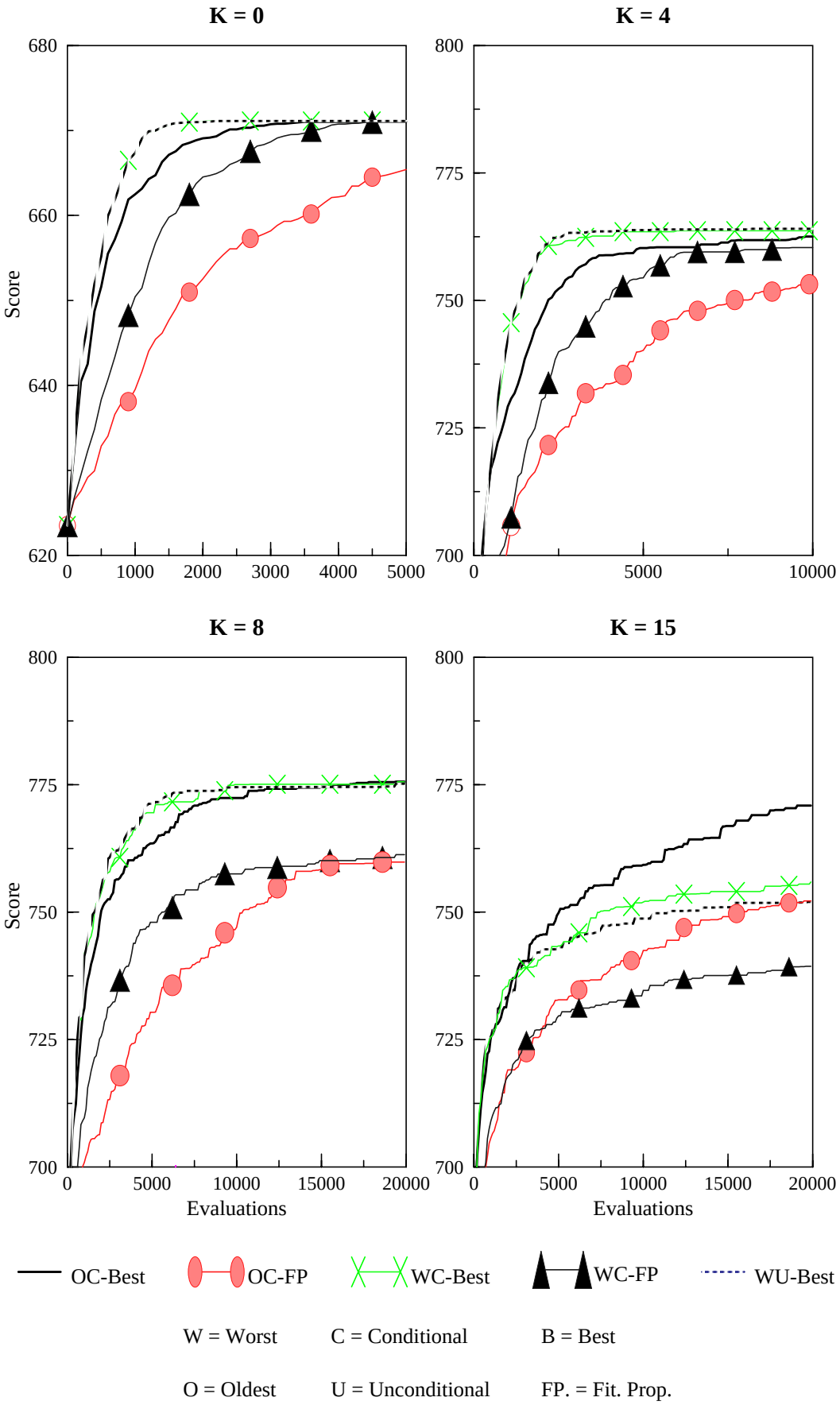


Figure 19: Replacement Policy Comparison

5.4.2. Recombination Policy

The standard algorithm above was run with every combination of 1point/uniform crossover, applied at 70% / 100% to the whole chromosome / problem representation only. The performance curves showed no clear pattern, with all of the variants showing similar performance in terms of rate of progress and the values of the fittest individual at the end of the run. The mean best fitnesses found, and the off-line performance (running mean of best individual seen so far) are given in Table 3. As can be seen, there are very slight differences in performance compared to the large differences found above. This shows that either the algorithm is highly tolerant of crossover mechanism, or that the nature of the selection pressure is such that the algorithm proceeds via mutation - based search. This fits in with the results reported in the previous chapter.

In order to test this a set of experiments was run using no crossover. These are shown as the final row in the table. For K = 4, the results are very similar, but above this the results are noticeably inferior to any of the algorithms employing recombination.This indicates that although the type of crossover used is not important, it does have a value in enabling the population to reach different optima.

Table 3: Recombination Policy Comparison

Crossover		*Applied to Mutation Encoding?*	*Best Value Found*			*Off-line performance*		
Operator	*Rate*		*K = 4*	*K = 8*	*K = 15*	*K = 4*	*K = 8*	*K = 15*
1 Point	70%	No	763.78	775.6	770.92	757.66	765.83	755.08
1 Point	70%	Yes	762.86	774.94	772.62	757.38	765.44	755.08
Uniform	70%	No	763.72	780.36	769.66	756.59	768.71	756.56
Uniform	70%	Yes	762.72	776.26	768.48	755.48	763. 99	756.27
1 Point	100%	No	763.58	774.86	767.66	756.93	762.27	755.92
1 Point	100%	Yes	763.28	777.26	771.82	757.89	765.93	757.08
Uniform	100%	No	763.18	778.82	772.98	755.21	763.91	757.24
Uniform	100%	Yes	762.58	776.5	773.78	755.26	764.09	760.55
Asexual	*n/a*	*n/a*	*762.66*	*766.54*	*744.4*	*757.76*	*757.56*	*737.05*

This finding that better results are obtained with the use of a crossover operator than without would appear at first to run counter to the findings of other researchers who have studied the relative utility of crossover and mutation on NK landscapes with high K [Hordijk & Manderick, 1995, Eiben & Schippers, 1996] where it was found that the utility of crossover decreased as the amount of epistasis was increased. The results presented in those two papers use a fixed mutation rate (0.5% and 1/len respectively), and the analysis concentrated on the ability of recombination operators to generate useful new points in the search space. However it was noted in Chapter 3 that there is a significant relationship between the crossover and mutation operators such that for many problems the “optimal” recombination function is dependant on the choice of mutation rate.

In Figure 20 the evolution of mutation rates is plotted over time on N = 16 landscapes for two values of K (8 and 15) with asexual reproduction, uniform crossover and 1 Point crossover. These curves clearly show that higher mutation rates are evolved by algorithms employing crossover, and further to this, that there is more difference between the rates evolved for different K in the presence of crossover. These results can be explained from the “Repair Mechanism” view of recombination.

Both the papers quoted above suggest that recombination will be an unreliable means of generating new individuals if there is no correlation between the positions of the optima in the search

space. This was backed up by the results in the previous chapter, where with fixed mutation rates less recombinatory strategies evolved at high K.However if mutation is considered as the primary search mechanism, then there is a different role for recombination, which is to recombine offspring created from the same peak, allowing the search to avoid Muller's ratchet. Thus in the presence of crossover the algorithm is able to sustain higher mutation rates and perform a more effective search without losing track of the good points discovered.

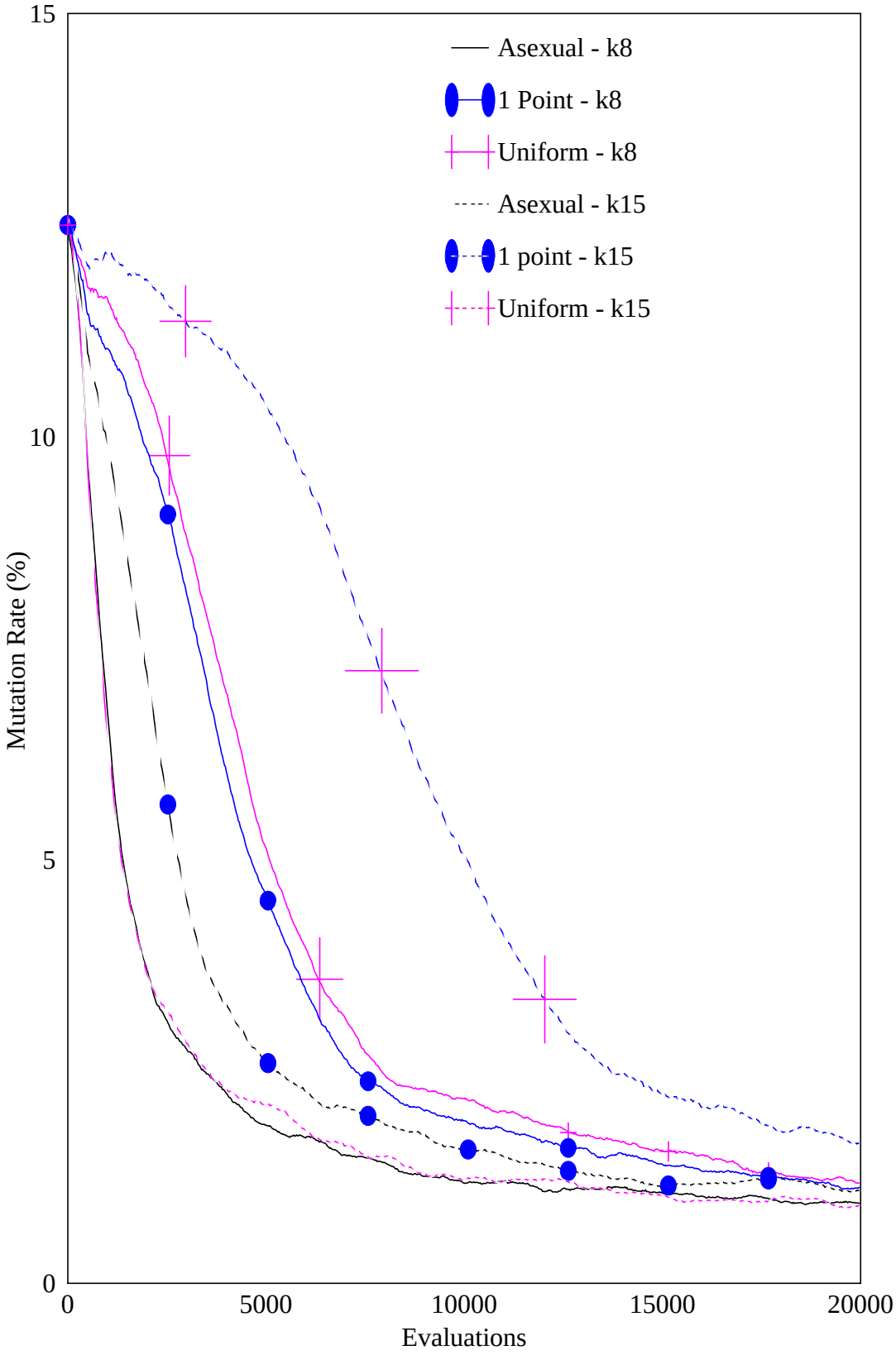


Figure 20: Evolved Mutation Rates with Different Recombination Operators

5.4.3. Mutation Encoding

Following the results of the last section, a set of experiments were run using 1 Point Crossover at a rate of 70%, with no crossover on the mutation encoding. The replacement was via (λ + oldest) tournament. The mean value of the fittest individual found, and the off-line performance are shown in Table 4.

Of the three different encodings investigated, the Gray coding and Binary encoding showed similar performance, both substantially outperforming the exponential encoding according to either measure. This was most noticeable on the more complex landscapes.

The use of 16 bits as opposed to 8 for the representation provides a slight advantage which is more noticeable on the more rugged landscapes, but the difference is small.

Also tested was the importance of the maximum value which the decoded mutation rate can take. This was done using 16-bit gray coding for the mutation rate, and simply changing the multiplication of the decoded value. The results are shown in Table 5.

Table 4: Mutation Encoding Comparisons

Mutation Encoding		*Best Value Found*				*Off-line Performance*			
Code	*Length*	*K = 0*	*K = 4*	*K = 8*	*K = 15*	*K = 0*	*K = 4*	*K = 8*	*K = 15*
Binary	8	671.12	763.4	776.36	767.82	669.43	757.12	765.63	756.22
Binary	16	671.12	763.78	779.64	770.08	669.48	757.44	769.21	756.63
Gray	8	671.12	763.86	775.46	767.74	669.42	757.67	765.46	755.45
Gray	16	671.12	763.78	775.6	770.92	669.6	757.66	765.83	755.08
Exp.	8	668.68	751.4	747.88	735.54	663.81	740.76	736.75	726.34
Exp.	16	668.76	749.24	751.04	736.26	663.2	738.14	736.6	723.09

Table 5: Effect of Changing Maximum Decoded Mutation Rate

Maximum Decoded Value (%)	*Best Value Found*				*Off-line Performance*			
	K = 0	*K = 4*	*K = 8*	*K = 15*	*K = 0*	*K = 4*	*K = 8*	*K = 15*
0	*668.76*	*749.24*	*751.04*	*736.26*	*663.2*	*738.14*	*736.60*	*723.09*
10	671.12	763.58	771.90	762.08	669.21	757.46	760.35	748.48
25	671.12	763.78	775.60	770.92	669.60	757.66	765.83	755.08
50	671.12	763.32	774.78	773.90	669.05	756.48	762.86	759.64
100	671.12	762.48	775.06	776.38	668.66	754.85	761.79	761.97

Clearly there is a trade-off between retaining the ability to learn large mutation rates which enable escape from sub-optima on less correlated landscapes, and the time taken to “learn” to low mutation rates on smooth landscapes. However the maximum value of 25% leads to the best performance on all but the random landscapes

Whether the encoding be Binary or Gray, there is always a single bit which when mutated

causes a large change in the decoded value, and it is this effect which can allow escape from sub-optima even when the mutation rate has evolved to a low value. The potential scale of this change, and hence the size of the basin of attraction from which the algorithm can escape via mutation, will depend on the range of values the decoded mutation rate can take. This effect is most obvious in the figures for the best value reached on complex landscapes (K = 15).

The disadvantage of having a wide range of rates available to the algorithm is that it can take longer to learn a suitable rate, especially in the final stages of the run when the rate is being minimised. This shows up especially in the off-line results on the low epistasis problems.

However it must be emphasised that these effects are tiny compared to those of selection and the size of the inner GA. This is good since the aim of self-adaptation is to build robust algorithms, and there would be little point if the choice of decoding range etc. became critical to success. It appears from these results that if nothing is known about the landscape to which the mechanism is to be applied, initialising the population at random over the range of 0-25% represents a good compromise.

5.4.4. Size of the Inner GA

The next set of runs used the values above (with a mutation range of 0-25%) and varied the value of λ. These results are shown in Figure 21. and offer strong empirical support for the 1:5 acceptance heuristic referred to above.

For low epistasis, (Figures 21a and b) there is no difference in the levels of fitness achieved, but there is a clear pattern that the speed of reaching the peaks is inversely related to the number of offspring created. In the previous chapter it was shown that recombination is a good strategy on this type of landscape, i.e. that many of the offspring created will be at least as fit as their parents. However, for every iteration only a single new member is added to the population, so the other λ - 1 evaluations are wasted from the algorithm's point of view.

This can be demonstrated by considering the time taken to achieve the optimum for the simple problem, K = 0. This was a mean of 5800 evaluations with 5 offspring and 9900 with 10. This near doubling of time to converge is also reflected in the time taken to reach lesser values such as 99% of the maximum (1300 evaluations vs. 2000) and 95% of the optimum (200 evaluations vs. 400)

As the amount of epistasis increases, and the landscape becomes less correlated, the situation changes for three reasons:

Firstly, the original offspring pre-mutation has a decreasing chance of being fitter than its parents. This was reflected in the previous chapter by the decreasing evolutionary success of highly recombinatory search strategies. The decrease in the safety ratio of recombination puts an increased emphasis on the mutation strategy.

Secondly, as the fitness-distance correlation decreases, so the region around any point generated by recombination contains a greater range of fitnesses. The local search mechanism is generating points which sample this region, and the more samples taken, the higher the chance of finding a fitter solution.

The third factor is the fact that as well as searching the problem space, the algorithm is also learning mutation rates, and having a larger number of offspring means that more strategies can be tried in parallel. Equally, as the mutation rate in the transition function Γ decreases, so it is more likely that several of the offspring will have the same mutation rate. Since mutation is applied separately to each locus, there will on average be $m_i \times l$ alleles changed. However this is a stochastic process, and it is quite possible for an individual with a large mutation rate to only change in a very few positions, giving a false impression of the utility of the strategy. Thus increasing λ, and taking a larger sample should aid the learning process.

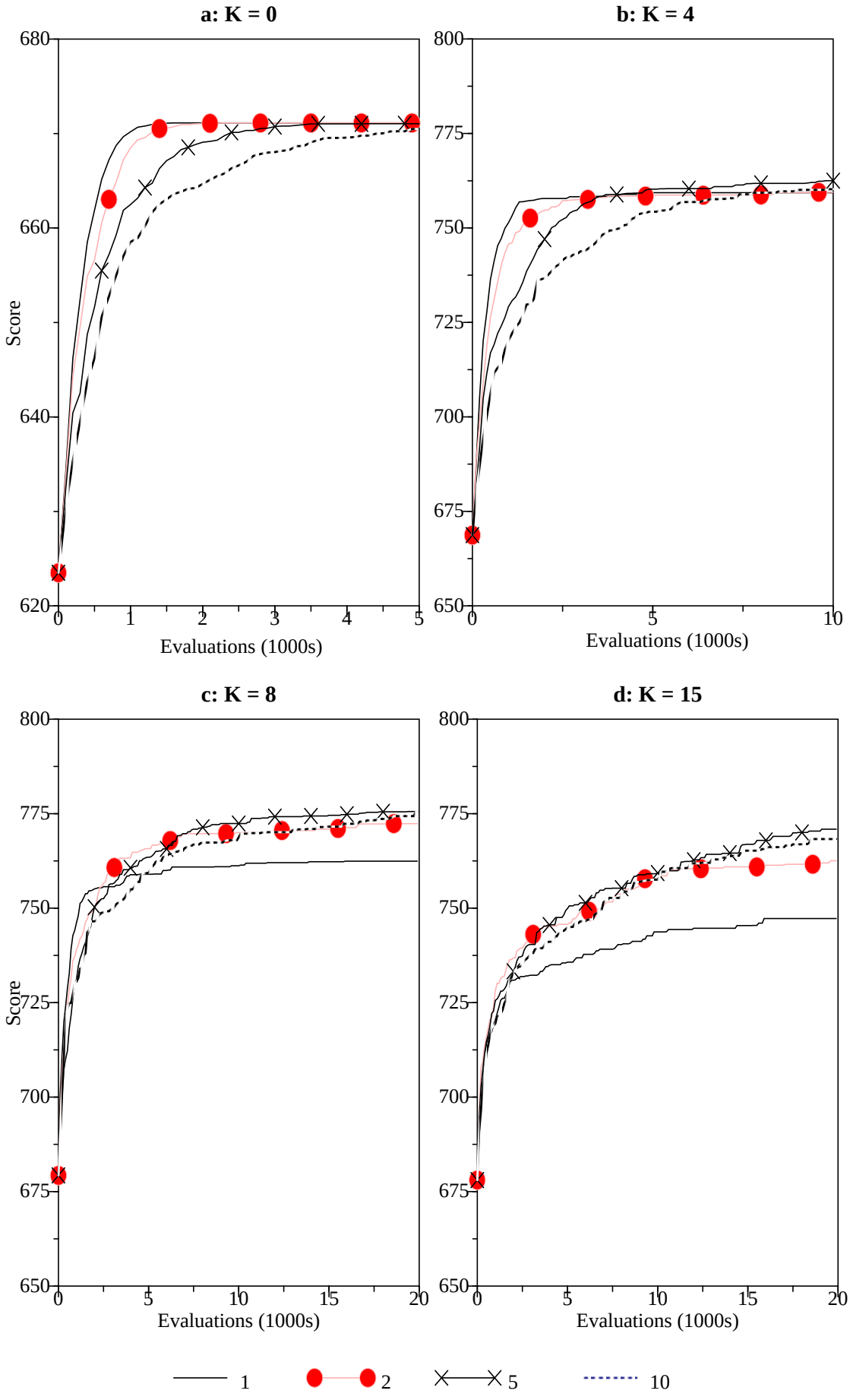


Figure 21: Comparison of Different Values of λ

For these reasons, as K increases so the curves for λ = 1 show stagnation and reach lower values than those incorporating more local search. However the improvements in maximum fitness obtained do not scale linearly with λ and from the plots above, the mean fitnesses reached with λ = 5 are higher than those for λ = 10.

In order to test this a single landscape and fifty populations were created for each value of K. Experiments were run using λ values of 1,2 5 and 10, and the fitness of the highest individual recorded for each run of 20,000 evaluations.These results are summarised in Table 6 below, where starred values are significantly different using Student's t-test at the 5% confidence level. As can be seen there is a significant advantage to using the local search mechanism, and the optimal value for λ (in terms of speed and values) reached is 5, in agreement with the 1:5 heuristic.

Table 6: Final Fitness Comparison: Varying λ

λ	*Mean Maximum Fitness*		
	K = 4	*K = 8*	*K = 15*
1	791.48*	755.28*	751.18*
2	796.12*	754.84*	763.14
5	799.26	767.2	766.68
10	798.42	763.6	764.8

5.5. Comparison with Standard Fixed Mutation Rates

5.5.1. Conditional Replacement

The optimal set of parameters and policies identified above were tested against a SSGA using a normal mutation mechanism with the same recombination parameters, population size and parent selection mechanism as the adaptive algorithm. The replacement policy was also to delete the oldest member of the population if its fitness was worse than that of the new individual.

The algorithms were compared using several metrics. These were the performance of the best of the current population, tested every 200 evaluations, the mean highest fitness reached, and (for the N 16 landscapes) the number of runs in which the global optimum was discovered. The experiments were run 10 times each on 10 different landscapes for each combination of K= {4,8,15} with N = and 32. Each run was allowed to continue for 20,000 (N=16) or 100,000 (N=32) evaluations.

A variety of authors have attempted to determine fixed values for the mutation rate which will yield good results across a range of problems, and a number of these common settings were tested including;

p_m = 0.001 [DeJong, 1975],

p_m = 0.01 [Grefenstette, 1986],

p_m = 1/l = 0.0625 / 0.03125, *(*where l is the length of the problem representation)

p_m = 1.75 / ($\sqrt{l}$ * μ) = 0.0044 / 0.0031- this comes from [Bäck, 1992b] as an empirical formulation of a result from [Schaffer et al.,1989].

Following initial experiments, which showed that higher mutation rates gave the best performance, a rate of p_m = 2/l was also tested.

Figure 22 shows plots of the current best in the population against time. Table 7 shows the fittest value found for each rate, along with the number of runs in which the optima was found for the N = 16 problems. For each problem class the 1-way Anova F value is given, (a value of 3.06 is significant with 99% confidence).

Table 7: Best Value Found: Conditional Replacement, Standard GAs

Mutation Rate	*N = 16*			*N = 32*		
	K = 4	*K = 8*	*K =15*	*K =4*	*K =8*	*K =15*
Adap-tive	749.101 (93)	778.337 (43)	762.106 (22)	757.306	774.088	731.648
DeJong	739.966 (49)	756.317 (16)	710.244 (1)	741.353	740.101	694.556
Schaffer	741.819 (53)	756.863 (15)	713.791 (2)	747.840	745.456	709.947
Gref.	741.799 (53)	760.008 (21)	718.616 (2)	750.999	751.375	715.639
1/len	748.751 (91)	773.014 (40)	744.284 (7)	758.730	764.333	729.257
2/len	750.161 (98)	780.632 (55)	762.519 (15)	760.101	775.467	738.318
F Value	5.517	29.414	110.468	12.762	58.491	76.710

As the amount of epistasis in the problem increases the adaptive policy shows a marked ability to continue to improve the fitness of the population compared with the fixed rate versions. There is a clear pattern of increasing performance with increasing fixed mutation rate, with the best overall performance coming from the value of 2/*l*, and most of the recommended rates demonstrating poor performance.

In Figure 22 the fixed mutation rate algorithms show curves which either stagnate or show very slow improvements, indicating that on most if not all of the runs the search had become trapped in a local optimum. As can be seen, the number and frequency of jumps in the performance curves increases with the mutation rate, suggesting that the discovery of new optima is mutation led

In contrast the adaptive mutation algorithm always shows continuing improvement. This is because there is always a chance that changing a single bit in a low mutation rate encoding will create an individual with a high mutation rate, providing a means of escaping local optima. The costs of this improvement are the overheads of the learning process. Even though the adaptive algorithm reaches similar values (except N = 32, K = 15) to the algorithm using a fixed rate of 2/*l*, it usually takes longer to do so, and the curves for the adaptive algorithm are often still climbing after the others have stagnated. Possibly allowing more evaluations would have favoured the adaptive algorithm.

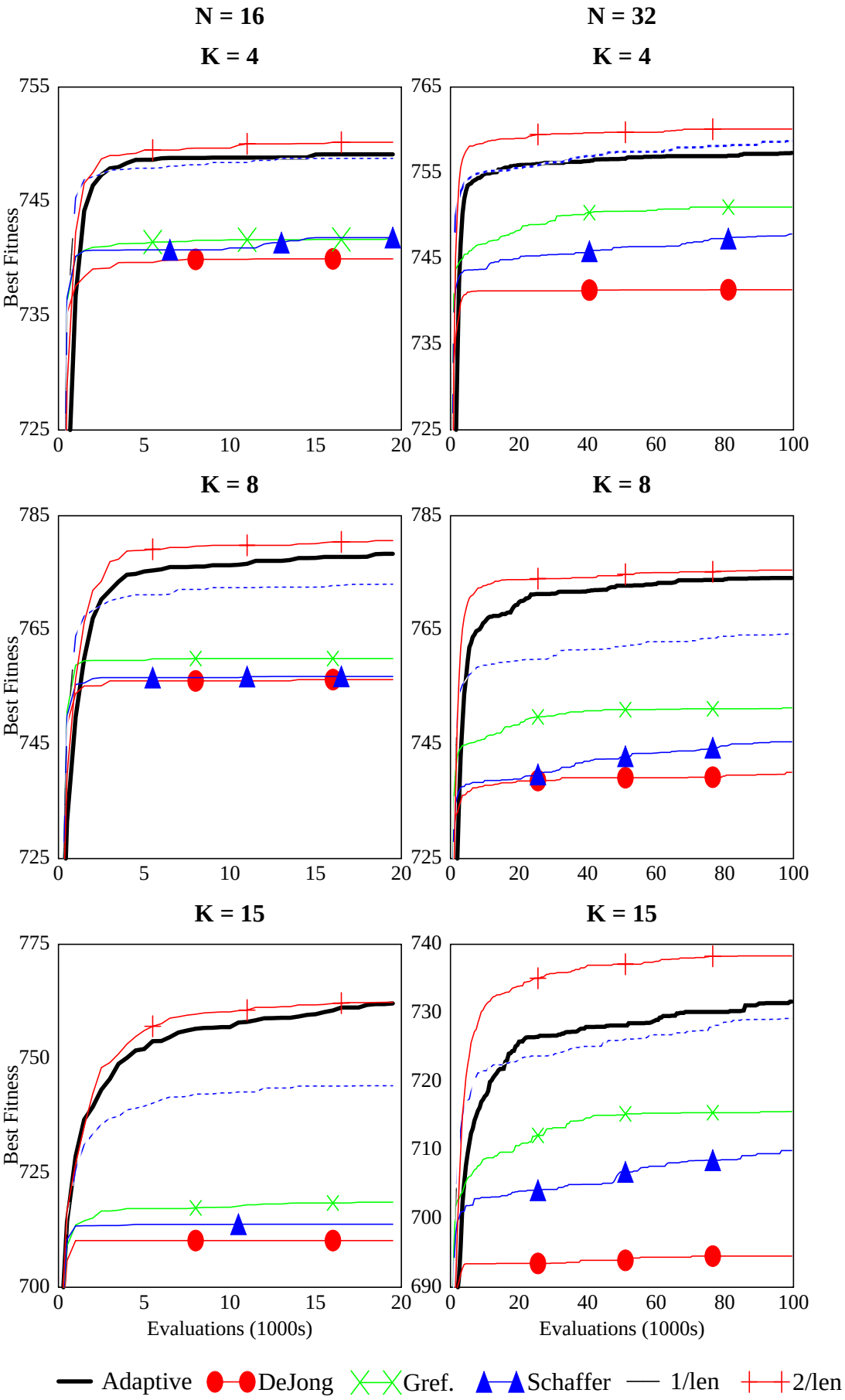


Figure 22: Adaptive vs. Standard GAs: Conditional Replacement

5.5.2. Elitist Replacement

In the previous section it was noted that many of the runs with fixed mutation rates got stuck in local optima, and that the higher the mutation rate the greater the chance of escaping. In order to test this, it was decided to run a further set of tests using a lower selection pressure. The selection pressure for all the algorithms was reduced by changing the replacement strategy so that the oldest member was always replaced by the new offspring unless it was the sole copy of the fittest in the population, and was fitter than the new offspring The results of these experiments are given in Table 8 and Figure 23. Again a F-value of 3.06 or above indicates a significant differences between groups.

Table 8: Best Value Found: Elitist Replacement, Standard GAs

Mutation Rate	*N = 16*			*N = 32*		
	K = 4	*K = 8*	*K =15*	*K =4*	*K =8*	*K =15*
Adap-tive	749.96 (97)	786.41 (73)	779.13 (41)	762.94	788.80	746.80
DeJong	737.34 (43)	745.40 (10)	716.73 (1)	739.56	732.37	700.29
Schaffer	738.11 (44)	750.12 (11)	716.20 (3)	746.57	742.44	705.93
Gref.	740.24 (54)	755.05 (17)	724.84 (2)	749.71	756.23	720.16
1/len	750.48 (99)	784.34 (66)	775.80 (37)	762.87	782.46	746.25
2/len	750.61 (100)	784.25 (63)	777.62 (38)	761.2	778.14	729.99
F Value	10.15	98.97	219.77	26.80	174.97	131.54

Comparing the results in Tables 7 and 8 shows that the relaxation of selection pressure makes little difference to the lowest rates, which still give poor performance, but improves the performance of the higher rates significantly both in terms of values found and the reliability of finding the optima. The relative ranking of 1/*l* and 2/*l* is reversed (although both benefit from the relaxation in pressure), and the adaptive algorithm gives the best performance. Figure 23 shows that the fixed rate algorithms now display a slower search but do not stagnate as they did before.

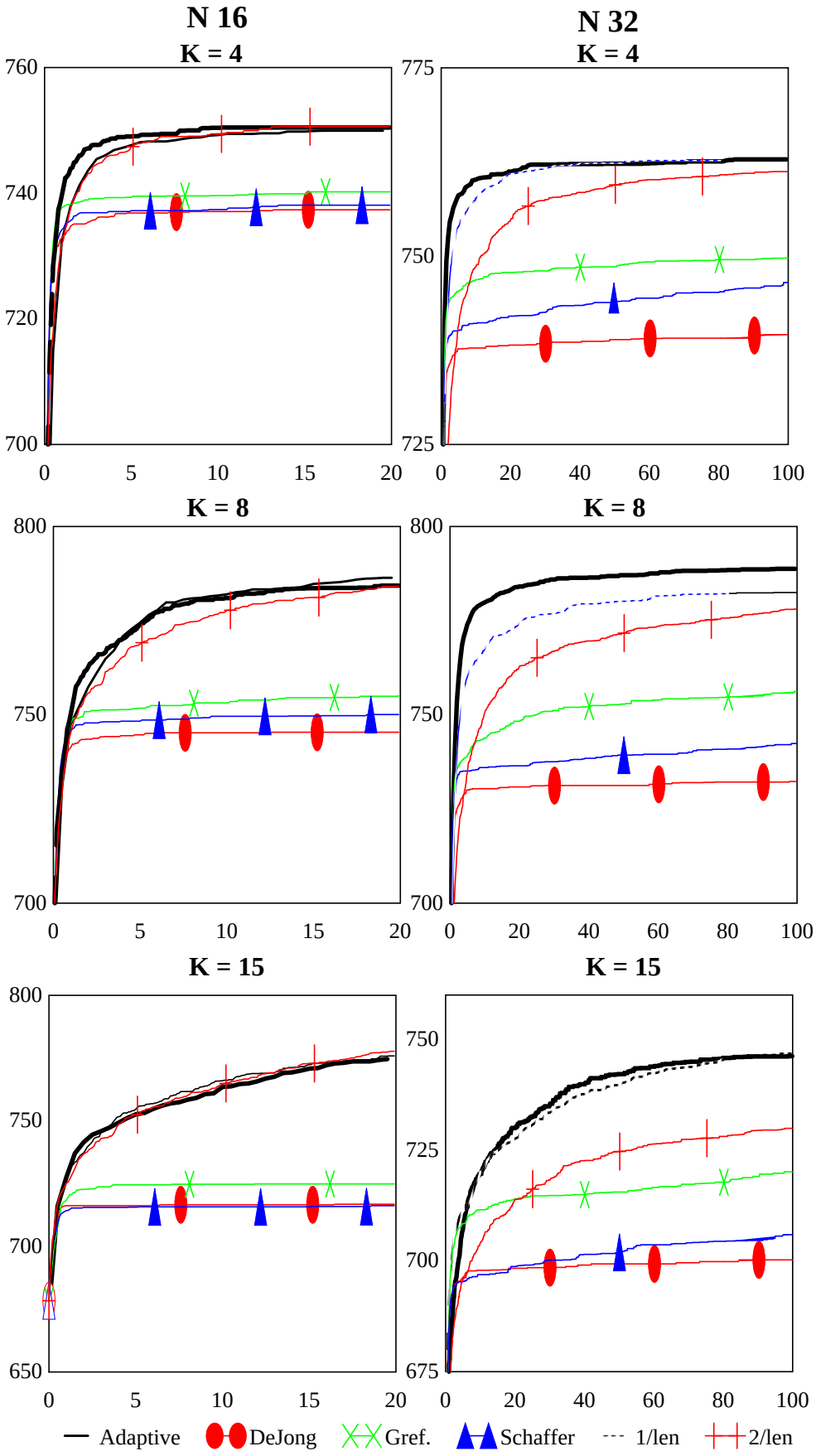


Figure 23: Adaptive vs. Standard GAs: Elitist Replacement

5.5.3. Conditional Replacement and Local Search

After noting that the version of the algorithm with 5 offspring is noticably better on the more complex problems than the single offspring version, further experiments were run in order to determine whether the improvements noted above were the result of adaptive mutation rates or simply the result of adding a form of local search to the GA.

These experiments used the same suite of fixed mutation rates as above, but this time in exactly the same algorithm as the adaptive mechanism.The results of running the Hybrid GA's with conditional replacement are shown in Figure 24 and Table 9. The fixed rate algorithms are all

Table 9: Best Value Found: Conditional Replacement, Hybrid GAs

Mutation Rate	*N 16*			*N 32*		
	K = 4	*K =8*	*K =15*	*K =4*	*K =8*	*K = 15*
Adaptive	749.101 (93)	778.337 (43)	762.106 (22)	757.306	774.088	731.648
DeJong	742.171 (58)	751.412 (7)	719.122 (3)	743.216	741.085	700.360
Schaffer	743.77 (63)	752.779 (8)	720.886 (2)	750.166	751.474	702.129
Gref.	746.601 (80)	761.631 (17)	728.948 (5)	754.611	759.032	717.579
1/*l*	749.696 (94)	774.103 (40)	746.24 (6)	759.032	768.044	732.547
2/*l*	750.042 (97)	781.652 (58)	760.229 (17)	760.801	778.348	741.557
F-value	2.648	49.819	71.294	10.444	49.347	76.411

improved by the addition of local search in that the problem of premature convergence to a local optimum is ameliorated. However, all of the commonly recommended values still display stagnation: there is still a pattern of increased performance with higher rates, and the adaptive algorithm outperforms all but the highest rate (which it beats for N = 16, K = 15). Again there is a time penalty associated with the adaptation.

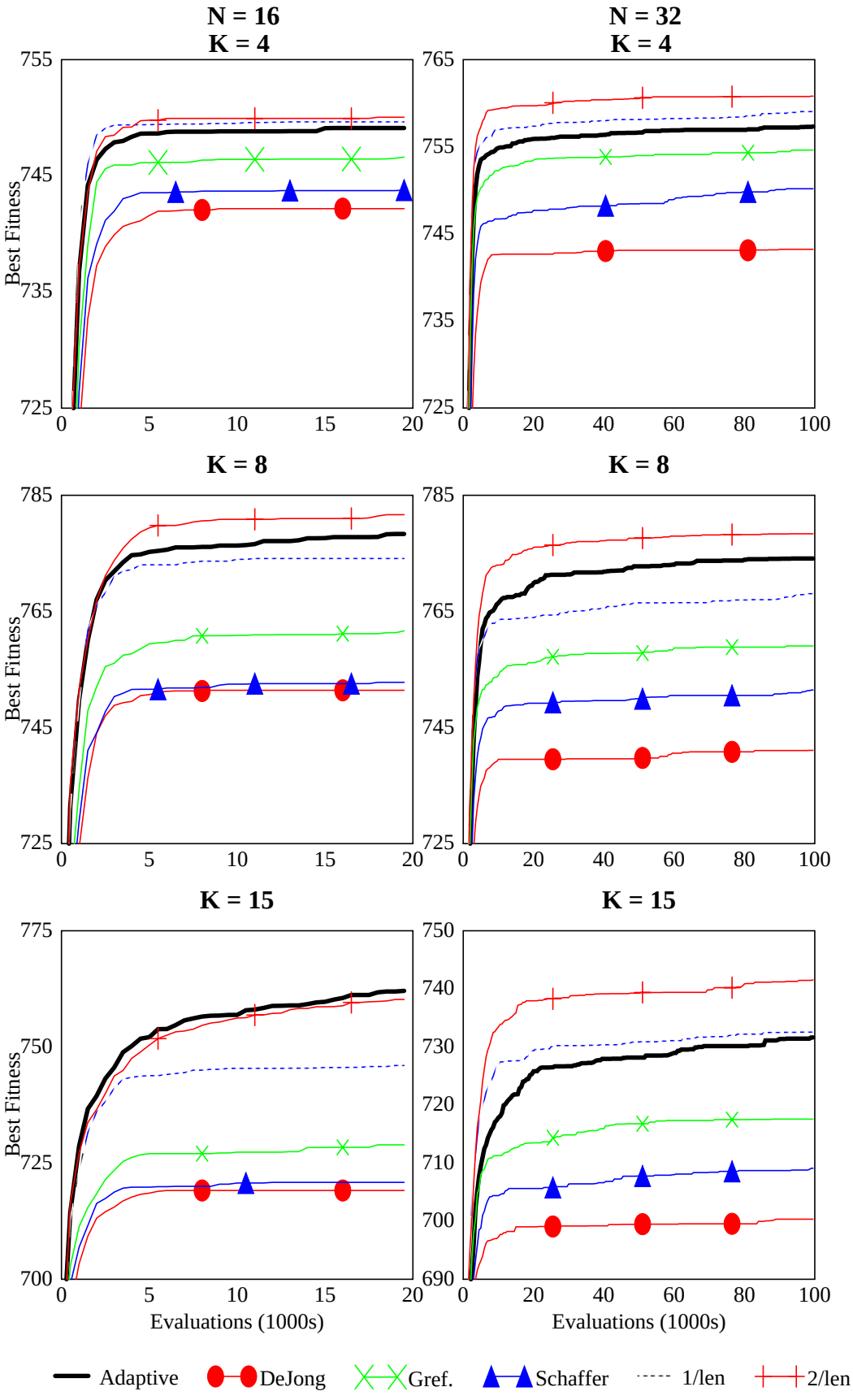


Figure 24: Adaptive vs. Hybrid GA's: Conditional Replacement

5.5.4. Elitist Replacement and Local Search

The results for the Hybrid GAs with elitist replacement are shown in Figure 25 and Table 10.

Table 10: Best Value Found: Elitist Replacement, Hybrid GAs

Mutation Rate	*N 16*			*N 32*		
	K = 4	*K =8*	*K =15*	*K =4*	*K =8*	*K = 15*
Adaptive	749.96 (97)	786.41 (73)	779.13 (41)	762.94	788.80	746.80
DeJong	737.83 (45)	749.43 (12)	719.66 (2)	737.82	741.23	698.97
Gref.	742.12 (64)	762.17 (18)	735.56 (7)	753.76	760.07	720.72
Schaffer.	739.64 (58)	755.82 (15)	730.6 (5)	744.89	748.87	712.94
1/len.	749.05 (93)	776.43 (44)	761.70 (17)	758.38	773.20	735.59
2/len	750.57 (99)	784.99 (65)	775.36 (31)	762.00	784.74	755.03
F-value	7.85	58.92	120.69	25.11	108.63	169.68

Comparing these with the previous tables shows that for most of the fixed rates these are the best results, suggesting that both if the changes helped the search to avoid becoming trapped in local optima. The same broad patterns are visible in these results as the others, but this time a fixed rate of $2/l$ provides better performance than $1/l$.

The best performance comes from the adaptive mutation mechanism, and the learning overheads are less noticeable on all but the most complex problems.

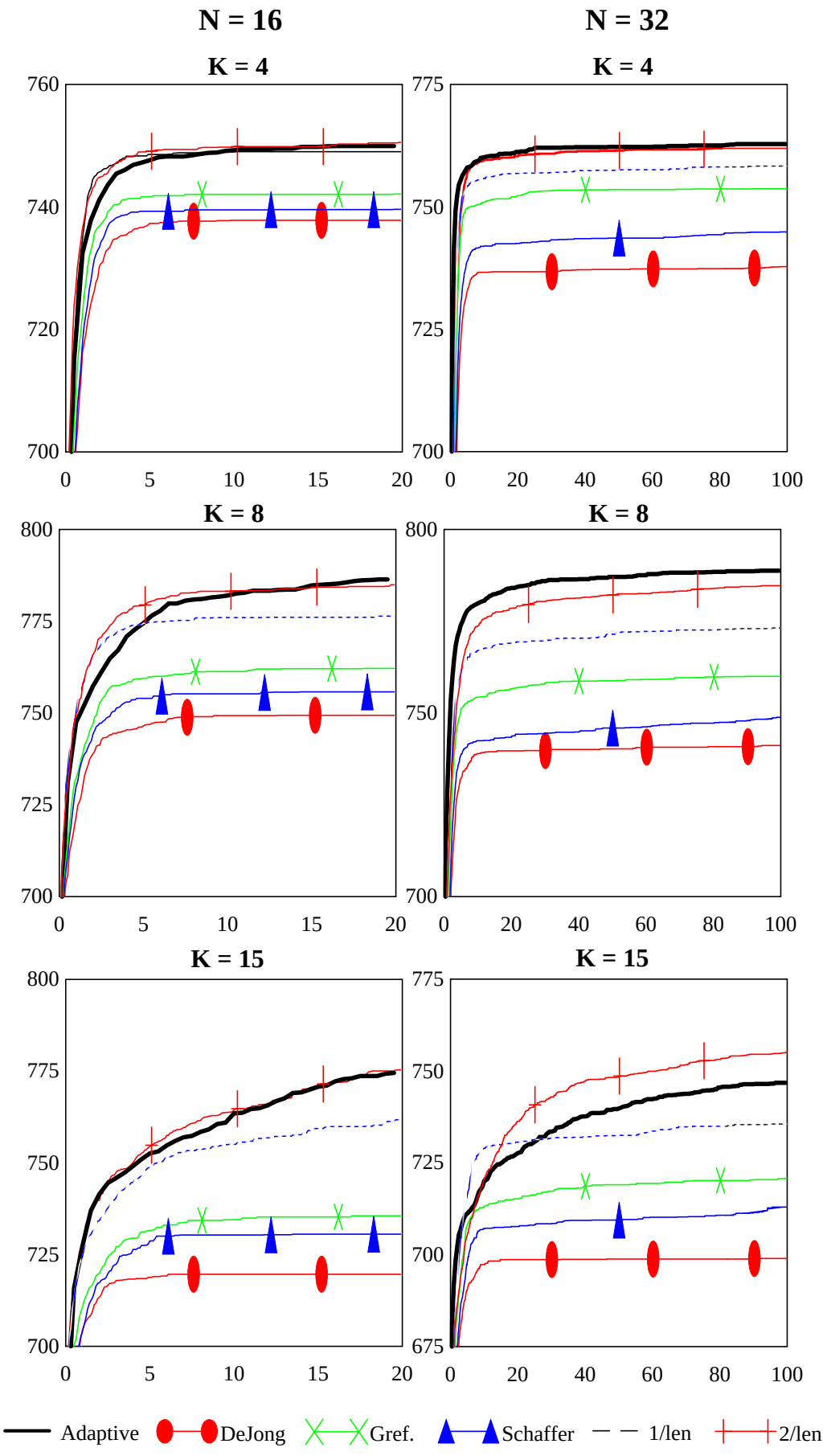


Figure 25: Comparison of Adaptive GA with Hybrid GAs: Elitist

5.5.5. Summary of Comparison Results

Table summarises the results above, showing the mutation rate which provided the highest mean value obtained for each landscape class.

Table 11: Best Algorithm for each Problem Class

	N = 16			*N = 32*		
	K= 4	*K=8*	*K = 15*	*K = 4*	*K = 8*	*K = 15*
Rate	*2/l*	*Adap-tive*	*Adap-tive*	*Adap-tive*	*Adap-tive*	*2/l*
Elitism?	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Local Search?	*N*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*

Although the fixed rate of 2/l yielded the best performance on the N = 16, K = 4 landscapes, the adaptive algorithm and the fixed rate of 1/*l* both yielded results which were not statistically significantly different in any of the four experiments.

As the problem complexity is increased, some clear patterns emerge. Firstly, there is a tendency for the fixed rate algorithms to get stuck in local optima, so that the searches stagnate for periods of time. In the previous chapter it was shown that mutation was the more effective search mechanism as K increased, and this is confirmed by these results. As Figures 22 -25 show, there is a clear relationship between the ability to escape from local optima (as indicated by the number and frequency of “jumps” in the value of the best in the current population, ending periods of stagnation) and the size of the mutation rate.

Of the four “standard” rates tested, the value of 1/*l* gave the best results, and when the extra value 2/*l* was tested it did better still (although not in all configurations). It is possible that an even higher rate would have performed even better on these landscapes. although the change in ranking when the selection pressure was reduced, suggests that the value is close to optimal (recalling from Chapter Three that the performance curves vs. mutation had a parabolic form for most of the algorithms on most of the problems tested).

The problem of getting stuck in sub-optima was shown to be ameliorated by both the addition of local search, and a reduction in the selection pressure. Both of these also helped the performance of the adaptive algorithm (the local search was shown in section 5.4.4 to be crucial), and on four of the six landscape classes, the best results came from the adaptive algorithm. There was a noticeable slowing in the rate of progress, which can be attributed partially to the overheads of learning extra information, and partially to the cost of evaluating λ individuals for every 1 incorporated into the population. The latter factor is most important on the less epistatic landscapes, when recombination is more useful as a means of producing fit individuals (and of course is not relevant when compared to the Hybrid GAs).

5.6. Analysis of Evolved Mutation Rates

In order to observe the behaviour of the algorithm, 10 landscapes were created for each of the standard K values used above, with N = 16. For each landscape 10 runs of 100,000 evaluations were made, with the mean mutation rate in the population observed every 50 evaluations. The results are shown in Figure 26 for λ =5.

From these plots the following observations can be made:

1. Since the genepool is randomly initialised the initial mean rate encoded will be half the maximum value allowed, but there is a high initial variance. During the first phase of the search there

is a period of rapid decline in the mean rate as individuals with higher mutation rate are selected out of the population. Given that all new individuals created using One Point Crossover, the speed at which individuals with higher mutation rates are removed by selection will reflect the relative value of recombination vs. mutation on the landscapes at a given stage of the search.The observation that high mutation rates are removed less quickly with increasing epistasis fits in with the previously noted results on the efficacy of recombination on epistatic landscapes.

2. In the second phase of the search, the mutation rate enters a stage of a far more gradual decline as the population converges. At this stage there is still a noticeable relationship between the K-value of the landscapes and the evolved mutation rate, with populations searching on more complex landscapes maintaining higher mutation rates.

3. After between 10,000 and 20,000 evaluations (depending on K) the populations begin to converge and the encoded mutation rates drop towards zero. The rate of decline in this final stage is slight. The later stages of the plots are not shown (for reasons of clarity), but there is little difference between the different values of K after around 20,000 evaluations. The mutation rates evolved drop from about 1% at 20,000 evaluations to around 0.65% after 100,000 evaluations. (The mean rates over the last 10,000 evaluations measured varied between 0.56% for K = 4 to 0.75% for K = 15).

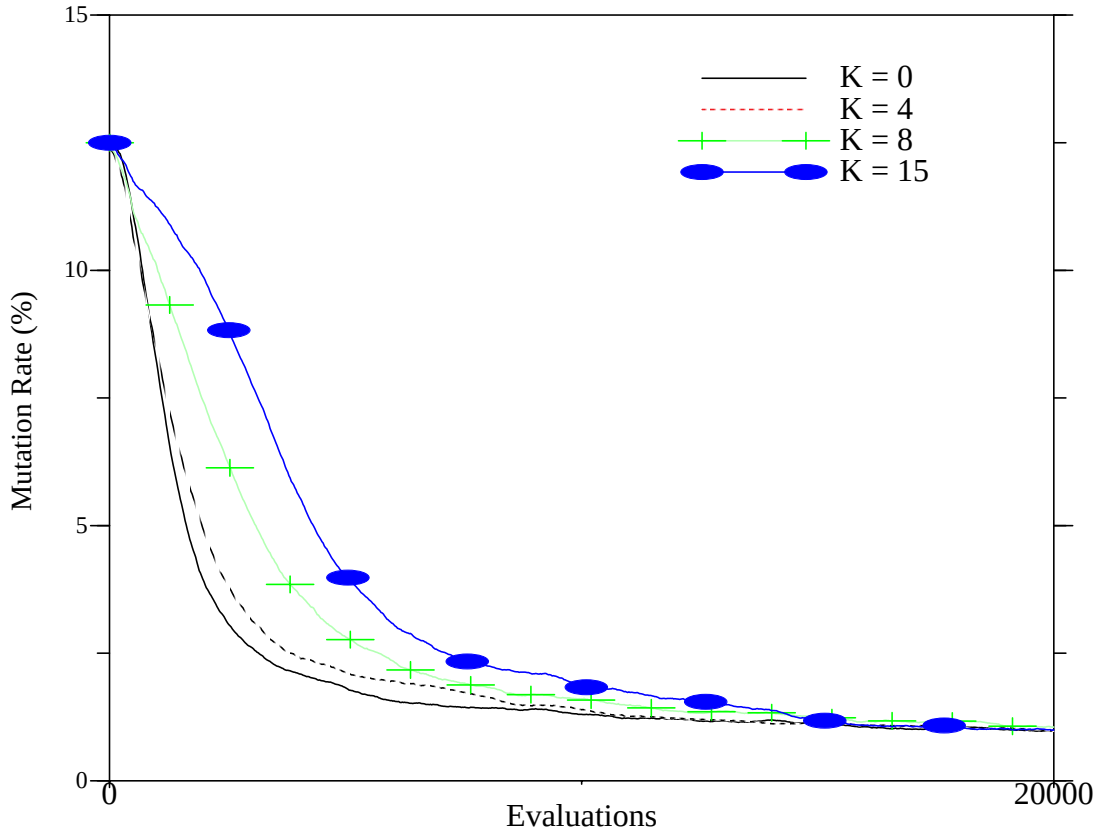


Figure 26: Evolved Mutation Rates vs. Number of Evaluations, λ = 5

This last feature of a very slow decline in the mutation rate towards zero is despite the effect of selection, which is to reduce the number of mutations likely in the production of a copy of the converged solution.

It can be shown that the mutation rates will eventually converge towards zero using a Markov Chain analysis. This was done in [Bäck, 1992b], but a simple version is sketched here for completeness. The reader is referred to e.g [Hoel et al.,1972] for a full discussion of the convergence of Markov Chains.

A converged population is assumed, so that the effects of selection and recombination can be ignored, allowing the proof to concentrate on the encoding of the mutation rates in a single individual. Since a finite sized binary representation is used, there are a finite number of values over the interval [0,25%] that the mutation rate can take. The value that the rate takes at the next time step is given by applying Γ to the encoding at the current step as defined above (D42). The mutation rate at any time

step is thus a state in a Markov chain, with the corresponding transition matrix Γ´ being defined by the probability of mutation from one state to another.

Some properties of this matrix can immediately be stated, based on a physical interpretation. The first of these is that for the state 0 (i.e no mutation encoded for) there is a probability of 1.0 of remaining in that state and 0.0 of moving to any other, i.e this is an absorbing state:

$$\Gamma'(0, a) = \begin{cases} 1.0 & a = 0 \\ 0.0 & a \neq 0 \end{cases} \quad \text{(D45)}$$

The second of these is that all other elements in the matrix are non zero. Even for a pair of states (a, b) represented by binary complementary patterns there is a finite chance of mutating all *l'* bits i.e. $\forall\, a \neq 0, \Gamma'(a, b) = p_a^{\,l'} > 0.0$ (where p_a is the decoded mutation probability in state *a* and *l'* is the length of the representation). Specifically, this means that $\Gamma'(a,0) > 0.0\ \forall\ a$.

Combining the fact that state 0 can be reached from all states, with the fact that once entered it cannot be left, tells us that all other states must be transient, and 0 is the only recurrent state of the system.

If X_n denotes the state of the system at time step *n*, then as a property of Markov chains:

$$P(X_{n+1} = 0) = P(X_n = 0) + \sum_1^{2^l} P(X_n = a) \times \Gamma'(a, 0) \quad (3)$$

Since all items in the summation on the right hand side will always be positive, $P(X_n = 0)$ must be increasing with *n*, hence the system will converge to a state of zero mutation. ■

As was noted, this sketched proof ignores the effects of selection and recombination on the population. It was reported in Section 5.4.4 that a major factor affecting the performance of the algorithm was the size of the inner GA, given by the value λ. It was found that there was a significantly improved performance on more epistatic problems when a degree of local search was allowed. This ties in with Bäck's findings that a significantly higher degree of selection pressure was desirable with self adaptation, which follows naturally from the fact that the algorithm is searching the problem space and the space of different mutation strategies simultaneously.

In order to test the effect of the extra selection pressure, evolved mutation rates were noted for different values of λ over the same set of problems as above. These results are plotted in Figure 27 for K = 4. The results for all other values of K were very similar. In this figure the x-axis denotes the number of generations rather than the number of evaluations, and the scale is logarithmic.

The plots with λ = 5 and 10 are virtually identical until the final phase, and during the first phase the mutation rates fall slightly faster than for λ = 1. This is because the larger sample size gives a better estimate of the value of a particular strategy. As the population converges to a solution there is a dramatic switch: the curve for λ= 1 continues to fall steadily towards zero as would be expected, but the other two flatten out and show the marked decrease in rate of change noted above, leaving a "residual" mutation.

It is the differences in the mutation rates evolved at the end of the runs which set the plots apart, and which hold the key to the improved performance of the algorithm with increased λ. As the size of the inner GA is increased, so there is a corresponding increase in the value of the mutation rate at which the population settles. The value of this residual mutation, which does not get selected out, is that it provides a means of avoiding becoming trapped in local optima. This explains the fact that the adaptive GA does not stagnate.

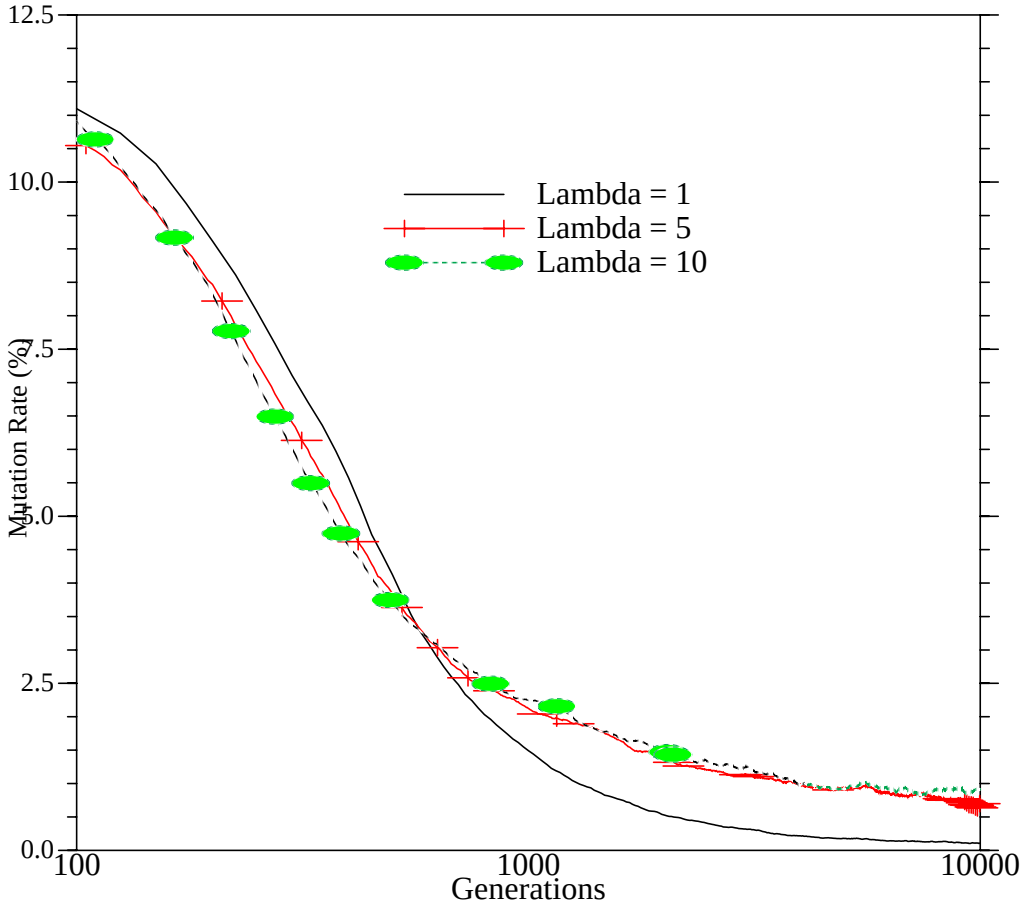


Figure 27: Evolution of Mutation Rate: Varying Lambda, K = 4

The residual mutation can be explained as a function of selection within the inner GA as follows:

For any copy of the current best string, with a mutation rate p_m, the probability that it survives both the transition function and mutation of its problem encoding intact is:

$$P(survival) = (1 - p_m)^{l'} \times (1 - p_m)^{l} = (1 - p_m)^{l+l'} \quad (4)$$

The probability that an individual is altered by mutation is simply the complement of this, and so for a set of offspring, the probability that at least one of them survives the mutation process intact is:

$$P(At\ least\ One\ Intact) = 1 - P(AllMutated) = 1 - P(OneMutated)^{\lambda} \quad (5)$$

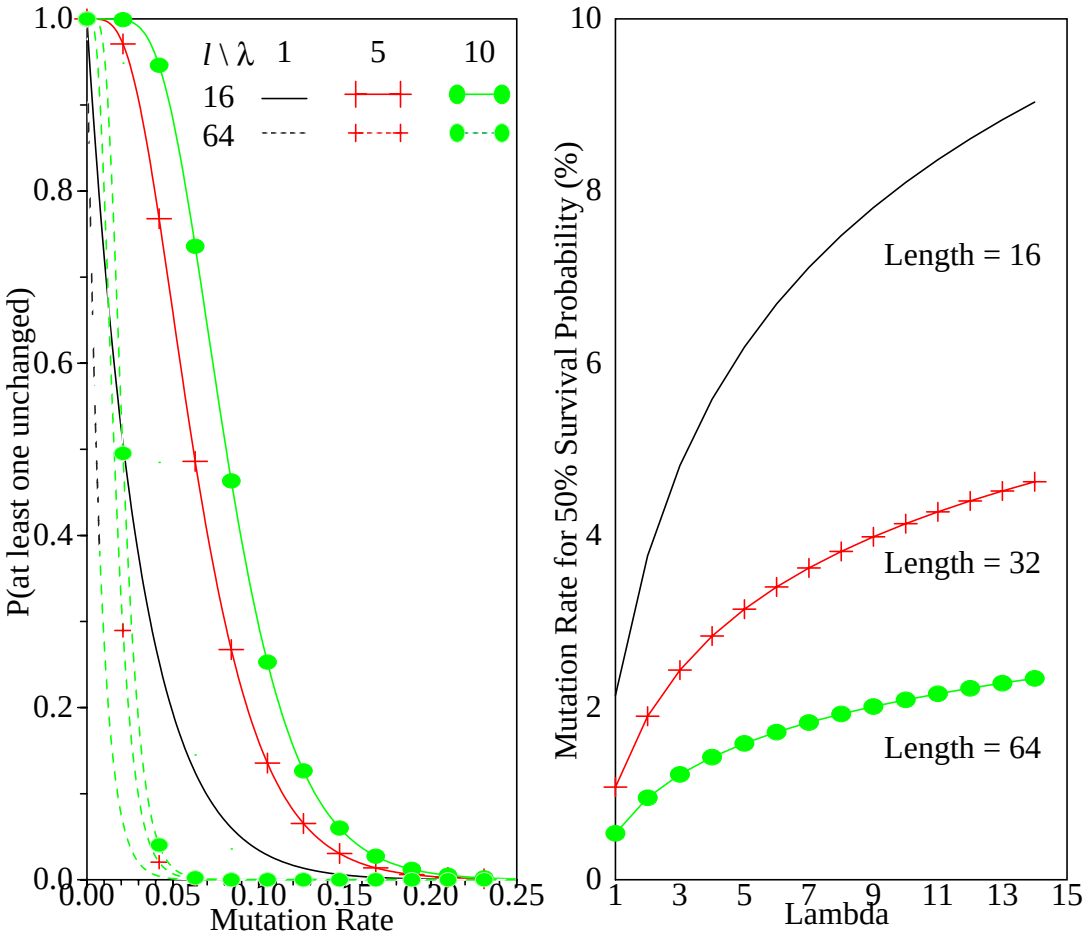
$$= 1 - (1 - P(survival))^{\lambda} \quad (6)$$

$$\therefore P(AtLeastOneIntact) = 1 - (1 - (1 - p_m)^{l + l'})^{\lambda} \quad (7)$$

In Figure 28a curves are plotted showing the probability of at least one individual surviving against mutation rate, for different values of λ and for two different length strings (16 and 64- in both cases $l = l'$ for clarity). These clearly demonstrate the fact that since the negative term on the right hand side of Equation 7 goes with the power λ, so increasing this value has a dramatic effect on the probability that mutation at any given rate will have no effect.

Equation 7 was used to calculate the value of the mutation rate at which there is a fifty percent chance of maintaining an individual, and the values obtained are plotted in Figure 28b.

These values demonstrate that although increasing λ from 1 to 5 increases the sustainable mutation rate threefold, there is only a slight further increase as the size of the inner GA is increased to 10, which explains the empirical confirmation of the 1:5 rule noted above.



a: Survival Probability as a function of Mutation vs. Rate

b: Mutation Rate for 50% Survival Probability

Figure 28: Survival Rates under Mutation

5.7. Conclusions

A mechanism has been presented which allows the incorporation within the SSGA of a Self-Adaptive mutation rate. This mechanism also incorporates some local search as a by-product of the means whereby it learns good mutation rates. The nature of the mechanism is such that it provides a form of distributed control for the GA whereby individuals in different parts of the search space may have different mutation rates.

Experiments with a variety of parameters have shown that the mechanism is robust in the face of major changes such as choice of crossover, mutation rate encoding, etc.

On the most complex, uncorrelated landscapes, Gray coding worked significantly better than binary coding. This is not surprising as Gray coding provides a much smoother landscape for the system to learn mutation rates, and as the landscapes become less correlated, so mutation becomes more important to the search process. This is why differences in performance generally become more noticable at higher values of K.

The most sensitive choice is which member of the population to replace, and with which offspring. The combination of a Delete-Worst policy with selection of the fittest offspring works well on all but the most complex problems, but overall the best performance came from using a Delete-Oldest policy. This has the advantage of being computationally simpler as it does not require the re-ordering of the population.

When the results from the performance comparisons (where the replacement pressure was

relaxed still further to a simple elitist FIFO policy), are taken in conjunction with the analytical and experimental results on the dependence of the evolved mutation rates on lambda, a more complex picture of selection pressure materialises. In previous work using generational models [Bäck, 1992b] it was found that a very high selection pressure was needed. By creating approximately 5 times as many offspring as parents are used, multiple strategies can be evaluated at different points in the search space.

However the results here demonstrate that by separating the two phases of selection, so that the inner GA selects for strategies, a more relaxed selection pressure can be used in the main GA, with the benefit from a more diverse population that the search is less likely to become trapped in local optima.

Analysis of the effects of changing the size of the inner GA showed that values greater than one allow the maintenance of an amount of “residual” mutation, with the population at a low rate. This, coupled with the chance of the transition function occasionally producing an offspring with a high mutation rate, is enough to allow escape from local optima and prevent stagnation of the search. A Markov Chain analysis was sketched to illustrate why, with a single offspring, the population will tend to converge to a state of no mutation, and probabalistic reasons were given why the rate of convergence to zero is much slower when more than one offspring is cloned. These also provided some support for the empirical 1:5 heuristic.

The comparisons with GAs using fixed rate operators (with or without local learning) showed two results. Firstly, for all non-trivial problems the inclusion of a form of local search can improve performance. Secondly the addition of the adaptive mutation mechanism significantly improves the performance of the Genetic Algorithm, as well as removing a parameter from the set of decisions faced by the operator.

It is possible that an extended search through fixed mutation rates might have yielded further improvements but this was not thought to be productive, as any such results would be entirely problem dependant, and the aim was to compare the performance of the adaptive algorithm with that of the most commonly recommended set of values. As discussed above, it is expected that for any given problem there will be a set of fixed operator values and parameters which outperforms an adaptive algorithm, (especially when the performance metric is time dependant) since the latter suffers from the learning overhead of adapting suitable strategies

Chapter Six:
Combining Self-Adaptation of Recombination and Mutation

6. Introduction

In previous chapters representations and operators have been proposed and investigated for achieving the Self Adaptation of recombination strategies and mutation rates within a SSGA. In this chapter two possible methods of combining the two algorithms developed previously are investigated. The resultant algorithms are compared to a suite of "standard" GAs which use common operators and settings, and are found to exhibit superior performance as measured via a range of metrics on the NK landscapes.

6.1. Description

In Chapters 2-4 a recombination algorithm was developed which was able to self adapt so as to preserve partitions within the genepool, thus achieving a degree of implicit problem decomposition. It was noted that there was a degree of sensitivity of the algorithm to the probability of mutation, although less so than the other recombination operators tested. In the work on adaptive mutation rates, only a single rate was evolved. However in [Bäck, 1992b] it was found that for certain types of problems there was an advantage in having separate rates encoded for each subcomponent, the problem being suitable identification of the sub-components. In this chapter two possible methods of combining the adaptive mutation and recombination algorithms will be investigated, one with individual level adaption of mutating rates and one with component level.

6.1.1. Single Mutation Rate: Individual Level Adaption

Initially a straightforward approach is taken and the two approaches are mixed with a single adaptive mutation rate added to the Lego representation for each individual. This is done as a gray coded binary string of the same length as the problem representation.

One offspring is created by Lego recombination, followed by an inner GA - mutation loop as per the last chapter. During the second phase, when mutation is applied to the problem representation, it is also applied at the same rate to the encoded linkage. This enables self adaption of both the mutation rate and the linkage.

Although conceptually very simple, there are a number of possible drawbacks to this approach. Firstly, as was noted above, for some problems there may be an advantage in having individual mutation rates for different subcomponents. Secondly, the relative contribution of a block of genes to the mutation rate will depend on its position, as well as the number of loci spanned, since the rate is gray encoded. In practice this means that the most important mutation genes are those in the last few loci. This would be far more serious if simple binary coding was used.

6.1.2. Multiple Mutation Rates: Component Level Adaption

Another solution to these problems is to combine the ideas of adaptive mutation with the Lego algorithm by learning useful blocks of linked genes along with a suitable mutation rate for each block. This provides a means of automatically decomposing the representation into a suitable number of sub-strings and mutation rates, since there are as many mutation rates in a given individual as there are discrete blocks of genes in it.

The fact that block definitions evolve over time immediately raises several questions, such as what rate should be given to the two blocks formed by the mutation of a larger one, or to the single block resulting from the linking over two smaller ones.

The simplest solution, which is used here, is to encode a distinct mutation value $m_{i,j}$ for each gene j in each individual i in the population. The probability of mutation within a block is then some function of the values attached to its constituent genes. Possible functions include using the lowest or the mean of the rates in the block. Here the mean is used, for the reason that this will be more stable

as the length of the blocks increases.

For reasons of size and scalability the mutation rate for each gene is encoded as a single value rather than as a binary string, and this necessitates a small change to the transition function for mutation Γ_m. The approach used here is the one from Evolutionary Strategies, namely to perturb the values by a random factor drawn from a Gaussian distribution with mean 0. This ensures that most of the search will be concentrated around previously successful mutation values whilst not precluding occasional large jumps (which were seen earlier to be of value in enabling escape from local optima).

6.2. Representation and Definition

Referring to the two algorithms discussed above as Adapt1 and AdaptM respectively, they are formalised as:

$$GA = (P^0, \delta^0, O, \mu, \lambda, l, F, R, M, \Gamma_R, \Gamma_{M,} p_r, p_s) \quad \text{(D46)}$$

where each member of P^t, O^t is of type:

$$a_i = (\alpha_i, L_i, R_i, m_i) \in I \times I \times I \times \Re^l \text{ for the multiple rate case (AdaptM)} \quad \text{(D47)}$$

$$a_i = (\alpha_i, L_i, R_i, m_i) \in I \times I \times I \times I \text{ for the single rate case (Adapt1)} \quad \text{(D48)}$$

A single gene at locus j in the ith member of the population is thus defined by the 4-tuple $a_{i,j} = \{\alpha_{i,j}, L_{i,j}, R_{i,j}, m_{i,j}\}$

The recombination operator, R, is the Lego operator defined earlier in (D34) with the corresponding transition function Γ_R, and this defines the linkage sets in the offspring. Again the necessary selection pressure for self-adaptation of the mutation rates is provided by using an inner GA, with λ members cloned from a single recombination event as per (D41).

The next step is to apply the mutation transition function Γ_m to the mutation rate of each gene in the offspring:

$$O'_i = \Gamma_m(O_i) = (a_{i,1},..., a_{i,l}, L_{i,1},..., L_{i,l}, R_{i,1},...,R_{i,l}, m'_{i,1},..., m'_{i,l}) \quad \text{(D49)}$$

For Adapt1 the transition function Γ_m works by applying the decoding function, D to each of the mutation rates m_i to get a value in the range [0,1] and then applying bit-wise mutation to each of the elements in the encoding with that probability:

$$m'_{i,j} = \begin{cases} 1 - m_{i,j} & X \le D(m_i) \\ m_{i,j} & X > D(m_i) \end{cases} \quad \text{(D50)}$$

where X is drawn uniformly from [0,1] for each position j.

For AdaptM, with real-valued m_i, the transition function Γ_m works by generating a random factor for each locus which is used to perturb the encoded rate:

$$m'_{i,j} = \Gamma_m(m_{i,j}) = m_{i,j} \times e^{N(0,1)} \quad \text{(D51)}$$

where $N(0,1)$ represents a random variable drawn separately for each locus from a distribution with mean 0 and std. deviation 1.

Finally value flipping mutation is applied to each element of the linkage and problem encodings $L_{i,j}$ $R_{i,j}$ and $\alpha_{i,j}$ at a rate $p'_m(i,j)$, which is given by

$$p_m'(i,j) = \begin{cases} \frac{1}{|S(i,j)|} \sum_{S(i,j)} m'_{i,j} & AdaptM \\ D(m_i) & Adapt1 \end{cases} \quad \textit{(D52)}$$

where $S(i,j)$ is the linkage set of locus j in offspring i for the AdaptM algorithm.

Following on from the results in the previous Chapter, the population updating p.d.f. is defined via a tournament between the λ members of the inner GA, the winner replacing the oldest in the current population, provided that the latter is not the sole copy of the current fittest population member.

6.3. Experimental Details

In order to test the effectiveness of the algorithms a series of experiments were run using the same sets of NK landscapes as before (10 runs were done on each landscape, giving 100 runs per NK combination). The same seeds were used so as to allow direct comparison of results with those obtained previously.

Again the approach taken was to compare the fittest member of the population after a fixed number of evaluations. Runs were continued for 20,000 evaluations on the 16 bit landscapes and 100,000 evaluations on the 32 bit landscapes, using a population of 100.

In the light of the results from previous sections, the two Adapt algorithms were compared with a range of combinations of operators which were found to be successful, namely 1-point, Uniform and Lego recombination together with the five fixed mutation rates specified in the previous chapter, and the hybrid adaptive mutation algorithm from Chapter 5 with 1-point crossover. All other details of the SSGA were kept the same.

6.4. Comparison Results

In Table 12 the mean value of the best value found is given for all fifteen algorithms, along with the number of times the optima was found (in braces) for the N = 16 landscapes. The results for K = 0 are not shown as all algorithms managed to reach the global optima on each run. The final column in the table shows the performance of each algorithm averaged over all combinations of N and K. in the table. The bottom row of values is the 1-way Anova F value, which always exceeds the value of 2.34 required for 99% confidence in their significance.

The term for the interaction between algorithm and landscape is also significant (F value of 23.43). In fact calculating this interaction term for the fixed algorithms only shows it to be still significant (F = 19.39), which is further empirical evidence for the arguments in favour of adaptive algorithms.

On all the NK combinations the mean best performance is shown by one of the adaptive algorithms. For the N = 16 landscapes this is always the Adapt1 algorithm, whereas for N = 32 it is always the 1pt-Adaptive-Mutation algorithm.

The performance of algorithms using Lego with fixed mutation rates are also better on the N = 16 landscapes than with N = 32, especially as the epistasis increases. This suggests that on the larger landscapes the adaptive recombination mechanism is unable to adapt sufficiently quickly to a strategy with high positional bias. One possible way of avoiding this would be to alter the way in which the population is initialised so that the mean initial block size was related to the length of the representation.

As shown by the final column, the two most successful algorithms are Adapt1, and 1-Point-Adaptive-Mutation. Of the static algorithms, the ones with $p_m = 1/l$ provide notably better performance than any of the other commonly used values, and than the extra value of $2/l$ which was also tested. This is mainly due to the differences in performance on the N=32, K = 15 landscape.

At the outset it was suggested that the self-adaptive mechanism was designed to provide a robust algorithm which would perform reasonably well on a range of problem types, rather than fine tuning it to do well on a particular sub-set of problems and suffering from poor performance on other subsets (as would be suggested by the “No Free Lunch” theorem [Wolpert and Macready, 1995]). However given the metric under consideration the adaptive algorithms, especially Adapt1, are superior across the suite of problems tested.

It must be stressed that these results relate to a particular performance metric, namely the best value found in a given (reasonably long) time. Comparisons using other metrics such as on-line performance, or shortening the time allowed for each run, would probably show a very different picture, as the adaptive algorithms are all quite exploratory in nature.

Table 12: Performance Comparison: Best Values Found

Operators	*N = 16*			*N = 32*			*Overall*
	K 4	*K 8*	*K 15*	*K 4*	*K 8*	*K 15*	
Adapt1	**750.69 (100)**	**789.59 (91)**	**791.67 (80)**	762.19	786.19	744.63	**770.83**
AdaptM	748.20 (85)	783.04 (62)	787.56 (63)	758.70	778.45	737.59	765.59
1pt-adaptive	749.96 (97)	786.41 (73)	779.13 (41)	**762.94**	**788.80**	**746.80**	769.00
Lego - 2/l	749.62 (96)	780.45 (53)	779.68 (42)	760.22	764.79	718.77	758.92
Lego-1/ *l*	750.26 (98)	784.30 (63)	772.38 (31)	762.37	780.22	730.47	763.33
1pt -2/l	750.61 (**100**)	784.25 (63)	777.62 (38)	761.20	778.13	729.99	763.63
1pt-1/ *l*	750.48 (99)	784.34 (66)	775.80 (37)	762.87	782.46	746.25	767.03
1pt- 0.001	737.34 (43)	745.40 (10)	716.73 (1)	739.56	732.36	700.29	728.62
1pt-0.01	740.24 (54)	755.05 (17)	724.84 (2)	749.71	756.23	720.16	741.04
1pt-1.75/(μ√*l*)	738.11 (44)	750.12 (11)	716.20 (3)	746.57	742.44	705.94	733.23
Uniform 2/l	749.70 (93)	779.68 (46)	777.09 (36)	757.88	761.58	720.90	757.81
Uniform-1/ *l*	750.08 (95)	783.00 (67)	776.77 (43)	760.64	783.06	725.08	763.10
Uni.-0.001	737.78 (48)	750.22 (11)	728.74 (2)	738.42	737.00	696.21	731.40
Uni-0.01	741.44 (56)	755.44 (15)	737.67 (4)	750.05	758.28	723.05	744.32
Uni-1.75/(μ√l)	739.91 (54)	752.03 (16)	732.14 (5)	743.29	747.39	710.03	737.46
F-value	7.98	79.74	186.49	21.67	108.51	94.65	211.00

Using a second common metric, the number of evaluations to reach the optimum, the situation with K = 0 (a simple unimodal problem) might be expected to show up the deficiencies of the algorithms for the following reasons:

1. Only one individual is incorporated into the population for every five evaluated, so on a

simple problem there is a delay in taking advantage of good schema.

2 There is an additional overhead of learning suitable block sizes and mutation rates.

However this is not borne out by the results from the experiments, where the time at which the optima was reached was noted. Since every run was successful, this is effectively the longest time taken to solve the problem. The number of evaluations taken is shown in Table 13 below.

Table 13: Longest time to Solve Unimodal (K = 0) Problem

Recombination	*Mutation*	*N = 16*	*N = 32*
Adapt1		2000	5600
AdaptM		700	8200
1pt	Adaptive	2500	6000
Lego	0.01	1000	1900
Lego	1 / *l*	1700	4000
1 Point	2/l	5300	45200
1 Point	1 / *l*	2200	7300
1 Point	0.001	3300	6200
1 Point	0.01	1200	2300
1 Point	1.75/ ($\mu * \sqrt{l}$)	1600	3000
Uniform	2/l	5400	48600
Uniform	1 / l	1300	3600
Uniform	0.001	2500	6600
Uniform	0.01	700	1800
Uniform	1.75/ ($\mu * \sqrt{l}$)	1300	2100

The best performance comes with low mutation rates. This is to some extent dependant on selection - a higher selection pressure allows the use of more exploratory algorithms (i.e. higher mutation rates) as was noted earlier in the experiments using a Delete-Worst strategy e.g on the MaxOne problem (Figure 10 on p. 47) and with adaptive mutation (Figure 19 on p. 68). There is an obvious trade-off here - the rates which solve the zero epistasis problems (K = 0) quickly are not sufficiently explorative to escape from local optima on the more complex problems.

More surprising is the result of the adaptive algorithms, which did not appear to suffer too badly from the expected time-penalties of learning extra genes. The Lego operator is among the fastest on both problems. The dependency of finding suitable parameters for fixed operators is especially obvious on the larger problems.

Comparing the solution times for the four adaptive algorithms, there is clearly an advantage to adaptation of the recombination strategies, and an overhead to learning mutation rates. On the N = 16 landscapes AdaptM is faster than Adapt1, and linkage analysis shows that this is because it is faster at adapting to a lower linkage.This order reverses for N = 32. For low linkage the number of blocks will approximately equal the length of the problem, so the size of the AdaptM algorithm’s search space is increasing rapidly compared to that of Adapt1 which only learns one mutation rate.

6.5. Analysis of Evolved Behaviour

During the performance comparison runs for Adapt1 and AdaptM, the mean linkage in the population was noted, as was the mean mutation rate and the mutation rate of the best individual in the population. These are plotted in Figures 29 and 30.

6.5.1. Adapt1 Algorithm

The Adapt1 algorithm shows patterns for the evolution of both linkage and mutation rates which are very similar to those observed when the two were considered in isolation.

In summary, both the mutation rate and the degree of linkage increase with the amount of epistasis in the problem, largely independent of problem size. There is a peak at around K=8 beyond which this smooth progression of evolved strategies stops, which ties in with Kauffman's observed "Complexity Catastrophe". Above this point the predominant factor in the rate of discovery of "good" new points is simply the rate of sampling new points in the search space as evidenced by the high values for the evolved mutation rates.

The two facets of the evolving search strategies complement each other well: for low epistasis the mutation rate decreases and the amount of linkage dwindles to zero, allowing a very BSC-like strategy. Similarly at high epistasis, where high linkage means that recombination is having little, if any effect, population diversity and the generation of new points in the search space is maintained by higher rates of mutation.

It must be emphasized that this is not a case of recombination and mutation strategies evolving separately and "happening" to work well together. The nature of the Self Adaptive mechanism is that the strategies are co-evolving. The mutation rates which emerge as "successful" by being associated with relatively fitter offspring, have done so *in the context* of the reproductive strategy encoded within that offspring, and vice-versa. Although not shown (for reasons of clarity) there is initially a wide range of both mutation and linkage rates, and it is the selection mechanism which rapidly discriminates between the multitude of reproduction strategies on the basis of quality of solution obtained.

6.5.2. AdaptM Algorithm

The evolved behaviour for the AdaptM algorithm (Figure 30) is very different. Although the pattern of increased linkage with increasing epistasis is still present, the size of the variation is far less, and even for the K=8 case the mean linkage only reaches around 40%.

The reason for this is the extremely high mutation rates which evolve, which will have a tendency to destroy large blocks. The mutation rates are much higher than for the Adapt1 algorithm, both for the mean and the best in the population. The same pattern is present as observed before that the evolved rates increase with the degree of epistasis in the landscapes.

These high mutation rates, coupled with a vigorous evolved recombination strategy, creates algorithms that will be highly destructive of higher order schemata.This helps to explain the (relatively) poorer performance of AdaptM on the more complex (K = 8) landscapes.

Despite this bias against longer schemata, the algorithm fares well in the performance comparison, which is evidence of the power of mutation as a search algorithm, when coupled with a suitable selection mechanism.

The mutation rates applied to the linkage bits were artificially reduced, and few trial runs showed patterns of strategy evolution far more like the Adapt1 algorithm, but there was a reduction in the performance. Inspection showed this to be the result of the formation of "greedy" blocks, with very low mutation rates which took over certain portions of the representations. Although it is probably possible to discover a separate mutation rate for the links which creates a happy medium between the two extremes, it was not felt fruitful to investigate this since such a value would be contrary to the spirit of the algorithm, and more importantly would almost certainly be problem dependent.

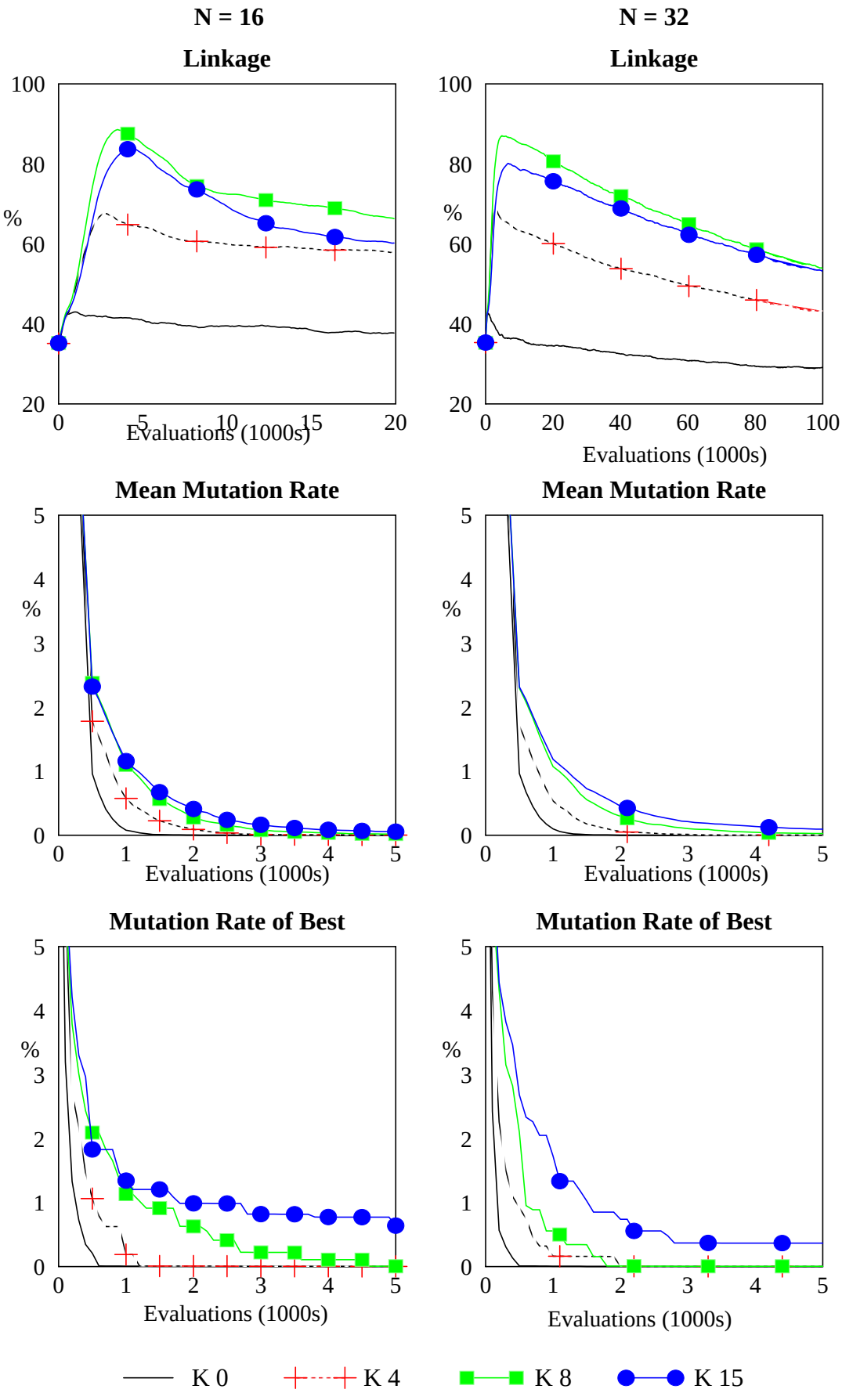


Figure 29: Evolved Behaviour: Adapt1 Algorithm

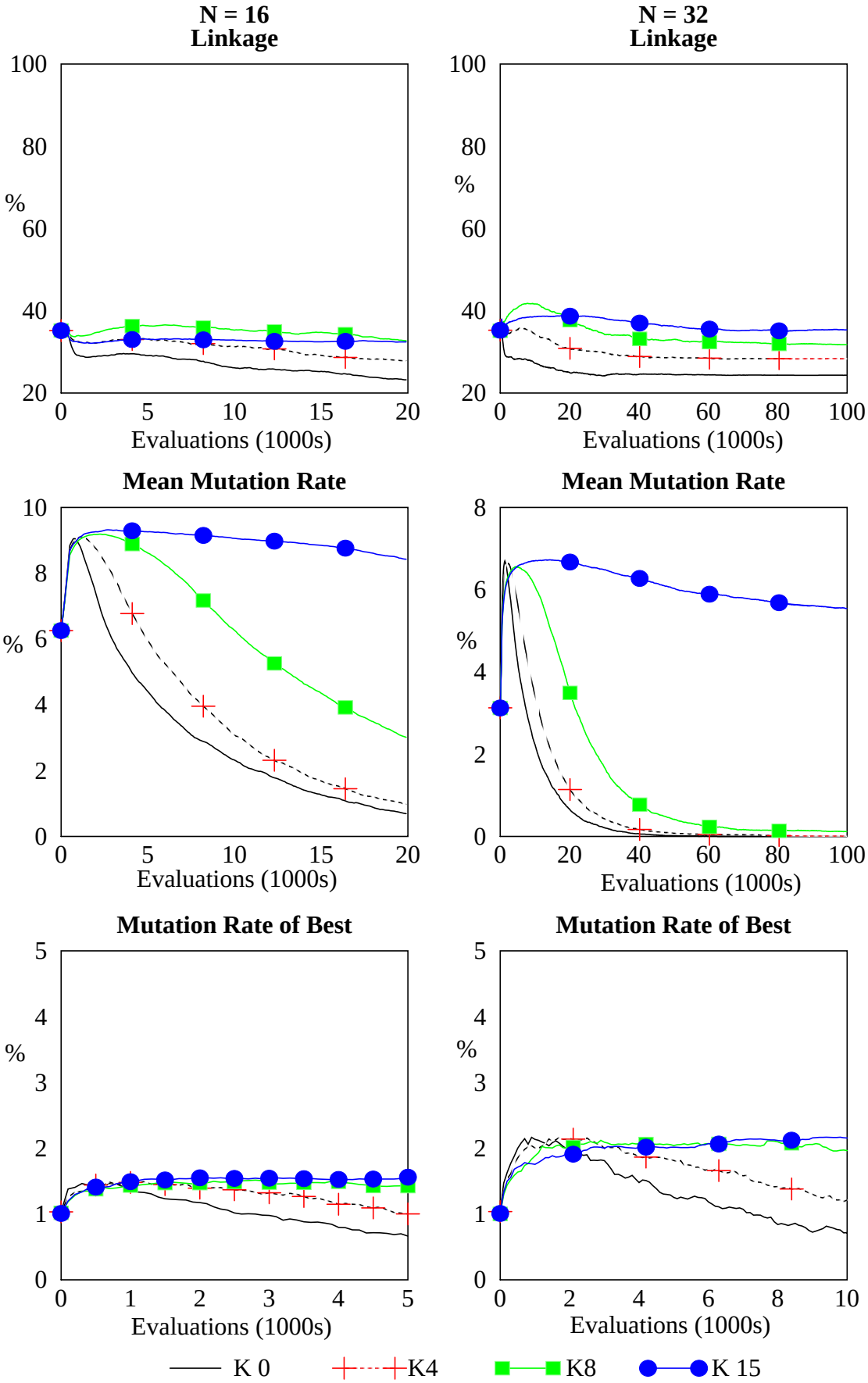


Figure 30: Evolved Behaviour AdaptM Algorithm

6.6. Conclusions

Two methods of combining the self adaptive algorithms developed previously have been investigated in the setting of a SSGA. The outcome of the performance comparisons showed these both to produce results which were significantly better than algorithms using a wide range of static operators.

The more successful of the two was the Adapt1 algorithm, which combined the Lego recombination mechanism with the Self-Adaptation of a single mutation rate. Plotting the evolution of the recombination and mutation strategies for different levels of problem complexity showed that the behaviours were very similar to what has been described before. Both the linkage (increasing with the amount of epistasis) and the mutation (ditto) strategies which emerge can be explained individually using the same arguments as were given in earlier chapters. What is interesting is that these emerge via co-evolution and yield significantly superior performance. This can be viewed as a synergy: the mutation operator evolves a more explorative p.d.f. as that of the recombination operator becomes more exploitative and vice versa.

Altering the algorithm to evolve a separate mutation rate for each block (AdaptM) produced results which were not quite as good as Adapt1, although still better than all but one the static algorithms. Investigation of the evolved behaviour showed extremely high mutation rates, with consequently low mean linkages, although there was still a pattern of increased linkage and mutation with increasing problem complexity. The overhead of learning a greater number of mutation rates also caused a slower search on the unimodal problems

Overall it would appear that the Adapt1 algorithm represents the more robust of the two approaches.These results demonstrate the effectiveness of the algorithm as a function optimiser, and its ability to adapt to display different behaviours on different types of search space. Importantly it does not seem to suffer from a great “learning overhead” on simple problems as demonstrated by the competitive times to reach maxima on the unimodal problems, and on more complex functions it discovers significantly higher optima than the majority of other algorithms tested. This can attributed to the synergistic effects of simultaneously adapting both recombination strategy and mutation rates.

The original intention of combining the two Self Adaptive mechanisms was to find an algorithm which would perform “reasonably” well on all landscapes, and it was expected that on any given landscape one of the variants of the standard SGA used would outperform it. Surprisingly this has not proved to be the case, the performance being better than expected as a result of the synergy between the two types of self-adaptation.

Conclusions and Discussion

The use of Genetic Algorithms as search algorithms, especially for function optimisation, is now well established in many different application areas. However, it has been shown by a number of authors that the quality of fitness values obtained can depend severely on the particular operator and parameter settings chosen.

In Chapter One the Genetic Algorithm was introduced and a formalisation presented. Some of the theoretical and practical results motivating research into adaptive GAs were discussed, and the concept of an "algorithmic space" was introduced within which all variants of the GA can be placed. In this context adaptation becomes a traversal of this algorithmic space under the influence of a Transition Function. Building on the formalisation a framework was developed to categorise adaptive algorithms, based on several complementary criteria.

The first criterion is what features of the algorithm are susceptible to adaptation. These range from the probabilities of applying specific operators (e.g. variable mutation rates) to the parameters they use (e.g. the "population size" parameter in selection algorithms) through to the form that the operators take. Into the latter category fall algorithms such as Schaffer and Morishima's "Punctuated Crossover" mechanism, where the form of recombination can adapt between the extremes of asexual reproduction (i.e. no crossover) and $l/2$ - point crossover (i.e. crossing over at every locus).

The second criteria is the granularity at which changes are effected. The terminology of [Angeline, 1995] was adopted, which divides algorithms into population, individual and component level adaptation.

The most important distinction was drawn according to the nature of the Transition Function governing the adaptation. This can be divided into two separate parts, the first being the evidence considered by the function, and the second being the form of the function itself.

A major distinction can be drawn between those algorithms which monitor the relative performance of several search strategies concurrently, and change probabilities of application according to the relative fitness of the offspring produced, and those that make changes on the basis of any other evidence.

Algorithms falling into the latter category necessarily use externally provided rules to make changes, even though these may be based on feedback from the search (such as degree of convergence etc.). In many cases these strategies are based on theoretical reasoning, but given the incomplete state of GA theory, this must be at best an inexact science.

The class of algorithms based on relative performance can be further sub-divided according to the form of the transition function. In [Spears, 1995] a useful distinction is made between "tightly coupled" and "uncoupled" algorithms according to whether the genetic operators themselves form the transition function or whether other externally defined functions guide the search. A more common usage than "tightly coupled" algorithms is the term Self Adaptation.

The three classes of algorithms distinguished on the basis of the transition function differ significantly from those of [Hinterding et al., 1997], who agree on the category of Self Adaptation, but split all other adaptive algorithms into two classes according to whether they utilise any feedback (of any kind) from the GA or not. Into the latter category ("Deterministic Adaption") fall only algorithms which change strategies according to evolutionary time, and all other algorithms are lumped together into an "Adaptive" category.

The framework proposed in this thesis differs in discriminating between those algorithms which test several strategies concurrently and use as feedback their relative performance of strategies, and those which base the transitions on other feedback, often only employing a single strategy at any given time.

As noted, Self-Adaption was defined as a means of accomplishing adaptation by embedding the parameters, or even definition of their operators within the representation and using the process of evolution itself to define their adaptation. All the arguments used in favour of using GAs to solve real problems apply equally to the search of the space of possible algorithms, which motivated this study of Self Adaptation in GAs

In chapter two a description was given of previous work on recombination in genetic algorithms, and the concept of Gene Linkage was introduced. Under the definition given, a set of genes are said to be linked if they are not separated by a recombination operator. Preserving linkage between genes allows the propagation of blocks of co-adapted genes over evolutionary time. Also, if the problem encoding is mutated faster than the block definitions change, then the preservation of blocks of loci from the destructive effects of crossover allows the testing of competing schemata within the partitions defined by those blocks.

A formalisation of this concept was developed to describe various operators in terms of a "linkage array". Based on this a specification was derived for a new multi-parental adaptive operator based on adjacent linkage arrays. This Linkage Evolving Genetic Operator (Lego) combines the desirable properties of self-adaptation with taking advantage of learned knowledge about linkage between genes which many operators discard.

In chapter three the implementation and testing of the Lego operator was described using a set of widely known test functions. For these problems the performance metric used was the time taken to reach the optimum. Despite the learning overheads associated with adaption, its behaviour was shown to be comparable to the optimal operator on a set of test problems, although the optimal operator for each problem was not the same. Some decrease in performance was noted on the multi-modal problems, particularly the deceptive Trap function, when compared to One Point Crossover. Experiments showed the operator to be relatively insensitive to changes in other details of the GA such as mutation rate and population size. It was found that the Lego operator worked better in a steady state setting, provided that a delete-oldest rather than delete-worst replacement strategy was used. On three of the four problems it provided the fastest solution times in the steady state setting.

An analysis of the behaviour of the algorithm was made by classifying individuals according to their mean linkage, and observing the numbers falling into various categories over time. Comparisons with the theoretical model of the linkage in the absence of any selective pressure shows that some kind of adaptation is indeed taking place. A clear pattern was noted that the degree of linkage evolving increased with problem complexity. This was explained in terms of the safety ratios of recombination and mutation on different types of landscape.

In order to investigate the phenomenon more fully, and to provide a more systematic basis for change between problems, Kauffman's NK landscapes were implemented. These are a well studied family of landscapes with tunable epistasis which allow control over the ruggedness of the search landscape.

In chapter four the linkage evolution was investigated over a range of these problems.The analysis of the behaviour of the algorithm shows that the recombination strategies which evolve can be explained in two complementary ways, based on different views of the search. The first viewpoint is based on fitness-distance correlation and the distribution of optima for the fitness landscapes concerned. In general it can be seen that as the degree of correlation of the fitness landscape decreases, and the optima diverge, so there is an increasing tendency for longer blocks of linked genes to take over the genepool. Since individuals created from large blocks will necessarily involve fewer parents than those created from smaller blocks, they will be less likely to combine information from a number of different peaks. Conversely if there is low epistasis and high fitness correlation, then smaller blocks will take over the genepool as individuals created from these are less likely to suffer from the deleterious effects of genetic hitchhiking.

The second viewpoint is that of schema disruption, and the changes in distributional and positional bias exhibited by the operator as the linkage and convergence of the population evolve. From this perspective, increasing linkage in a diverse population causes a reduction in the distributional bias towards low order schemata. This enables the algorithm to preserve and discriminate between schemata occupying higher order partitions of the search space.

An obvious potential criticism of this model is that is geared around links between adjacent bits, and that on problem representations which do not follow this pattern it will perform poorly, and be unable to adapt to the structure of the landscape. However the results obtained with random interactions show that this is not the case. This confirms that the adaptation of recombination

strategies is triggered by something more than identifying and simple keeping together genes with co-adapted allele values. Two possible reasons for the differential evolutionary success of different linkage strategies according to the problem type have been identified above.

Performance comparisons with other crossover operators were carried out on a range of landscapes with varying size and epistasis. These showed that the differences in performance (for the metric of best value found) were not significant on any but the N = 32, K = 15 landscapes.

In chapter five a mechanism was presented which allows the incorporation within the SSGA of a self-adaptive mutation rate. This incorporates some local search as a by-product of the means whereby it learns good mutation rates. The nature of the mechanism is such that it provides a form of distributed control for the genetic algorithm whereby individuals in different parts of the search space may have different mutation rates.

Experiments with a variety of parameters showed that only the size of the inner GA and the replacement policy appeared to be important. As before a delete oldest policy was found to work far better than delete-worst, as it allows escape from local optima. This is believed to be a fundamental rule for self adaptation in GAs.

Analysis of the effects of changing the size of the inner GA showed that values greater than one allow the maintenance of an amount of "residual" mutation with the population at a low rate. This allows escape from local optima and prevents stagnation of the search. A Markov Chain analysis was sketched to illustrate why, with a single offspring, the population will tend to converge to a state of no mutation, and probabalistic reasons were given why the speed of convergence (of mutation rates) is much less when more than one offspring is cloned. These also provided support for the empirical 1:5 heuristic. Although it was considered beyond the scope of this thesis, it is believed that a similar Markovian proof for the convergence of mutation rates to zero holds in this general case, even when the effects of selection and recombination are allowed for.

Performance comparisons with GAs using fixed rate operators, with or without local learning, showed two results. Firstly, for all non-trivial problems the inclusion of a form of local search can improve performance. Secondly the addition of the adaptive mutation mechanism significantly improves the performance of the Genetic Algorithm, as well as removing a parameter from the set of decisions faced by the operator.

In chapter six, two methods of combining the self adaptive algorithms developed previously were investigated. The outcome of the performance comparisons showed these both to produce results which were significantly better than algorithms using a wide range of static operators.

The more successful of the two was the Adapt1 algorithm, which combined Lego with the self-adaptation of a single mutation rate. Plotting the evolution of the recombination and mutation strategies for different levels of problem complexity showed that the behaviour of both was very similar to what has been described above. The same arguments as before can be used to explain this behaviour, and why it gives superior search performance. It was noted that there was a strong synergy between the two forms of adapted behaviour, and it must be emphasised that this arises purely as a result of the co-evolution of recombination and mutation strategies by Self Adaption.

Altering the algorithm to evolve a separate mutation rate for each block (AdaptM) produced results which were not quite as good as Adapt1, although still better than the static algorithms. Investigation of the evolved behaviour showed extremely high mutation rates, with consequently low mean linkages, although there was still a pattern of increased linkage and mutation with increasing problem complexity.

In conclusion therefore the Adapt1 algorithm, is the more robust way of combining the two reproductive operators developed earlier. It demonstrates the ability to adapt search strategies to very different forms according to the nature of the landscape, and does so with remarkable success. Certainly if quality of solution found is the most important criteria, then it outperformed any of the combinations of widely used operators tested. Algorithms implementing these techniques have successfully been applied to problems such as Software Test Generation [Smith and Fogarty 1996d] and Microprocessor Design Verification [Smith, et al. 1997].

Initial experiments have also been carried out on the use of this algorithm in the time-varying

environments used in [Cobb and Grefenstette, 1993] with various rates and magnitudes of environmental change. These indicate that the Adapt1 algorithm is able to track changes in the position of the global optima without the use of trigger mechanisms. Analysis of the behaviour shows an increase in the mutation rates following a change in the position of the optima, as more exploratory strategies are successful. If the rate of environmental change is slow enough, then the mutation rates can be seen to decay back to a low rate as the population establishes itself on the new optimum position.

Unfortunately, although the off-line performance appears to compare very favourably with that of an algorithm using Triggered Hypermutation [Cobb and Grefenstette, 1993], the on-line performance is inferior. This is a result of the cost of learning suitable mutation rates.Further experimentation would be required to see if this problem could be overcome, so as to produce algorithms suitable for on-line control.

In the first chapter it was suggested that the Genetic Algorithm can be viewed as the iterated application of two processes, namely Generating new sample points in the search space, and Updating the working memory (population) of the algorithm. The operators developed here all address the first process, and do so with notable success, via Self Adaptation.

What this thesis does not tackle directly is the issue of achieving adaptation of the Updating processes, namely the selection pressure and population size. In practice these (especially the selection pressure) are influenced by degree to which the reproductive operators' p.d.f.s are exploitative (e.g. low mutation rates, high linkage) rather than explorative. However there is another reason for not tackling them here. Self Adaption relies on selection rewarding the success of strategies relative to one another in the same population, i.e. population-level Self Adaptation is a contradiction in terms. Evidently it is not possible to compare the relative success of two different population sizes within the same population. Equally, even when a distributed mechanism such as tournament selection is used, encoding and comparing different selection pressures would involve the collection of more global statistics and could not truly be called Self Adaptation.

However it is believed that there is considerable scope for developing heuristic adaptive algorithms based on observations of the evolving self-adaptive behaviour.

In previous chapters, the evolution of linkage was observed on a range of NK problems of differing complexities, with epistatic interactions between both adjacent and random loci. Despite the fact that the Lego operator utilises component level adaptation, and mean linkage is a population level statistic, there is a clear pattern that during the principal search phase, the mean population linkage stabilises at a level which increases in a manner roughly proportional to the complexity of the problem. This is a direct result of an increasing selectional advantage in keeping together blocks of genes as non-linearity in the problem increases.

As the linkage increases, so does the size of the blocks of linked loci in the genepool. Each of these blocks defines a partition over the search space, which in turn defines a class of relations which exist in that partition. In order to solve a problem it is necessary to make informed decisions between relations and their instances (schemata). It has been shown that the size of the population needed to solve a problem increases with the order of the highest relations that must be chosen e.g. [Goldberg et al., 1992], [Kargupta and Goldberg, 1994]

Hence if the mean population linkage increases, implying a selective advantage towards preserving longer blocks, so will the order and number of possible schemata that
can inhabit them. In order to obtain a better estimate of their relative fitness, a larger population is preferable.

Equally, if the degree of linkage in a population is decreasing, this shows that a strategy of more disruptive recombination is proving successful, which corresponds to more mixing of building blocks. This process will be facilitated by a smaller population as there is a higher chance of the juxtaposition of two building blocks due to the faster turnover of members

On the basis of these observations there is believed to be considerable scope for future work in developing algorithms which adapt their population size in response to observed linkages.

In summary a new framework for classifying adaptive algorithms has been proposed, two mechanisms have been investigated for achieving self adaption of the reproductive operators within a genetic algorithm, and a means of combining them was investigated. These reduce the need for the designer of an algorithm to select appropriate
choices of operators and parameters, and were shown to produce significantly better results than any of the combinations of fixed operators tested, across a range of problem types. It was suggested that observation of the nature of the evolving search strategy on an unknown landscape could yield information about its structure.

It is hoped that this work will aid designers of evolutionary algorithms for the “real world” and provide a contribution towards the topic of adaptation and self-adaption.

Appendices

A. Model of Linkage Evolution

1.1. Notation

In order to model the evolution of linkage within the population we will initially restrict ourselves to considering the proportions of linkage bits in different states between a single pair of adjacent genes *i* and *i+1* within an infinite population. There are four possible states that the linkage can be in, according to whether the bits R_i and L_{i+1} have the value *true* (1) or *false* (0). We will refer to these as the ordered pairs [0,0], [0,1], [1,0] and [1,1] and will use the following notation:

a	$\in$	$\Re$	Proportion of the population with links [0,0] at time t.	(D53)
b	$\in$	$\Re$	Proportion of the population with links [0,1] at time t.	(D54)
c	$\in$	$\Re$	Proportion of the population with links [1,0] at time t.	(D55)
d	$\in$	$\Re$	Proportion of the population with links [1,1] at time t	(D56)
$a', b', c', d' \in \Re$			Proportions of the population after recombination.	(D57)
$a'', b'', c'', d'' \in \Re$			Proportions of the population after mutation.	(D58)

1.2. Proportions after Recombination.

Given that we insist that recombination respects linkage, and noting that the juxtaposition of two genes may result in their becoming linked (see page 32) we can construct the proportions after recombination as below, where # is used as a wildcard "don't care" symbol.

a' = P(0, #) * P(#,0 | recombination occurs) (D59)

b' = P(0, #) * P(#,1) | recombination occurs) (D60)

c' = P(1, 0) * P(#,0 | recombination occurs) (D61)

d' = P(1,1) + (P(1, 0) * P(#,1 | recombination occurs)) (D62)

Evaluating these gives

$a' = (a+b) * (a+c) / (a+b+c)$ (8)

$b' = (a+b) * (b) / (a+b+c)$ (9)

$c' = (c) * (a+c) / (a+b+c)$ (10)

$d' = d + (c) * (b) / (a+b+c)$ (11)

1.3. Proportions after Mutation

For the purposes of analysis we assume value-flipping mutation with probability *p*. This gives us the following proportions after mutation:

$a'' = (1-p)^2 a'' + p(1-p)(b' + c') + p^2 d'$ (12)

$b'' = (1-p)^2 b'' + p(1-p)(a' + d') + p^2 c'$ (13)

$c'' = (1-p)^2 c'' + p(1-p)(a' + d') + p^2 b'$ (14)

$d'' = (1-p)^2 d'' + p(1-p)(b' + c') + p^2 a'$ (15)

Concentrating on the proportion of genes that are linked after recombination and mutation have taken place, and substitute equations (8) to (11)into (15) gives:

$$d'' = (1-p)^2\left(d - \frac{bc}{a+b+c}\right) + p(1-p)\left(\frac{b(a+b)+c(a+c)}{a+b+c}\right) + \frac{p^2(a+b)(a+c)}{a+b+c} \quad \text{(D63)}$$

(D64)

recognising that *(a+b+c) = (1-d),* and re-arranging yields

$$d'' = d(1-p)^2 + \frac{(ap+b)(ap+c) + p(1-p)(b-c)^2}{1-d} \quad (16)$$

If we now do the same for the other proportions we get the following results:

$$a'' = (1-p)^2\frac{(a+b)(a+c)}{(1-d)} + p(1-p)\frac{(b(a+b)+c(a+c))}{(1-d)} + p^2\left(d+\frac{bc}{(1-d)}\right) \quad \text{(D65)}$$

This can be re-arranged and factorised to yield

$$a''(1-d) = a(1-d)(1-p) - a^2p(1-p) + bc + p^2d(1-d) + p(1-p)(b-c)^2 \quad \text{(D66)}$$

Similarly

$$b'' = (1-p)^2\frac{b(a+b)}{(1-d)} + p(1-p)\left(\frac{(a+b)(a+c)+d(1-d)+bc}{1-d}\right) + \frac{p^2c(a+c)}{(1-d)} \quad \text{(D67)}$$

which yields

$$b''(1-d) = b(a+b) + p(1-p)(a^2 + d(1-d)) + p^2(b-c)^2 - p(a+2b)(b-c) \quad \text{(D68)}$$

and

$$c'' = \frac{(1-p)^2c(a+c)}{(1-d)} + p(1-p)\left(\frac{(a+b)(a+c)+d(1-d)+bc}{1-d}\right) + \frac{(1-p)^2b(a+b)}{(1-d)} \quad \text{(D69)}$$

which gives

$$c''(1-d) = c(a+c) + p(1-p)(d(1-d) + a^2) + p^2(b-c)^2 + (a+2c)p(b-c) \quad \text{(D70)}$$

These recurrence relationships mean that given the starting proportions of the various combinations of link bit values we can plot the evolution of linkage in the absence of selection pressure. The algorithm is implemented by initialising the population so that each link bit is set *true* with a biassed probability *B*. In Figure 31 the evolution of the proportion of linked genes is shown for a variety of values of *p* and *B*. The curves in this graph use the equations derived above and each was run for 10,000 iterations. Each iteration corresponds to one generation of a GGA with an infinite population.

As can be seen the linkage evolves to a steady state at which the rate of destruction of linked genes by mutation is equalled by the rate of production by recombination. Interestingly the steady state towards which the system evolves is determined solely by the mutation rate, independent of the starting conditions. The steady state concentration appears to approach 0.5 from below in the limit as *p* -> 0. These values reached are given in Table 14.

Table 14: Steady State Proportions of Linked Genes

Mutation Probability	*Steady State Proportion*
0.001	0.47815
0.005	0.45255
0.01	0.43442
0.05	0.36782
0.1	0.32815

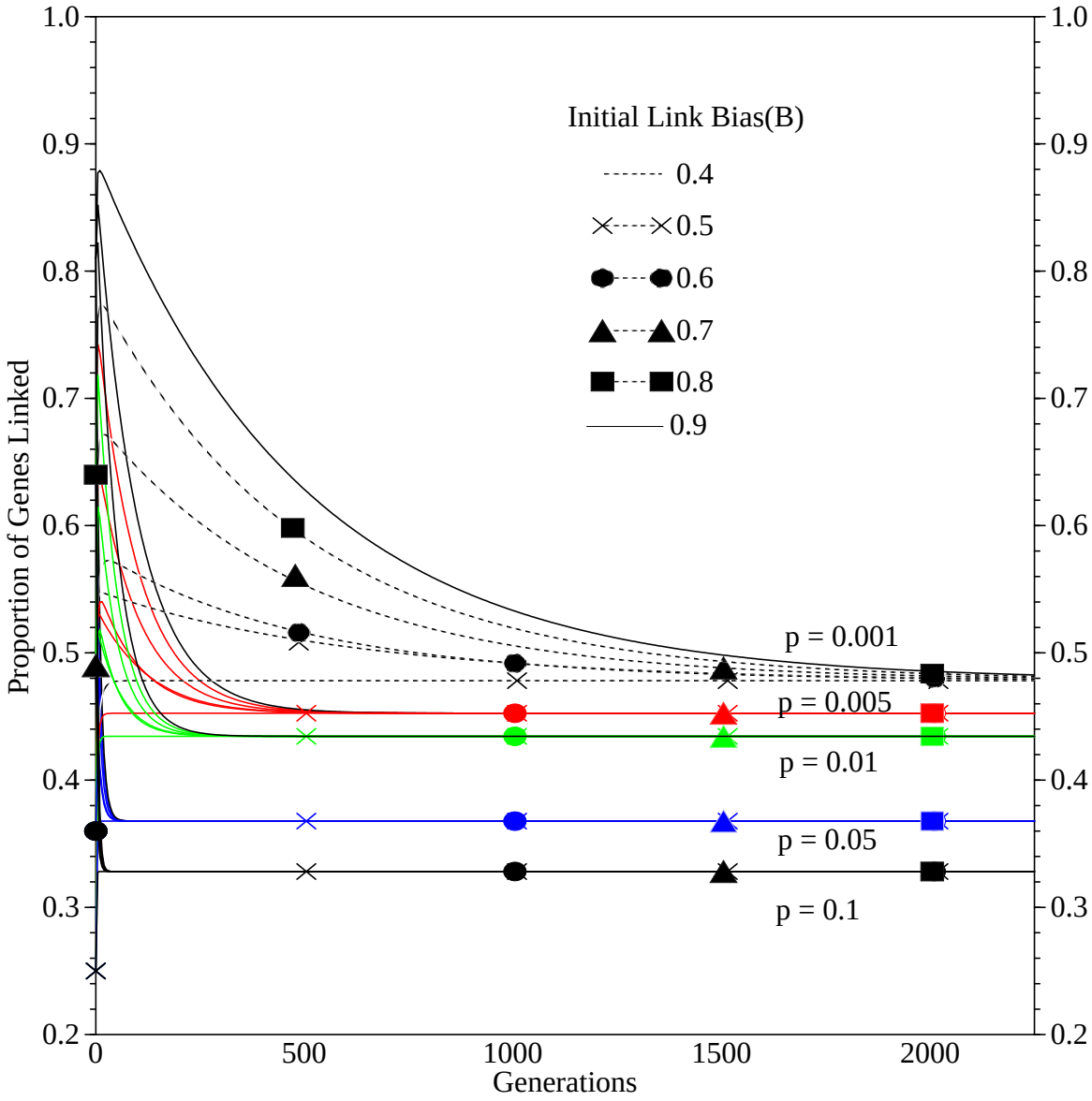


Figure 31: Evolution of Linkage in Absence of Selective Pressure

In the light of this apparent insensitivity to initial conditions, it was decided to do another run with more random initial conditions. The first evaluations represented the case where there was no selection pressure for or against linkage at any stage after initialisation, and therefore we knew the relative proportions of *a*, *b* and *c* . This second experiment mirrors the situation where, there is an initial selection pressure for or against linkage, which disappears as the population in these two loci converges to the values for the problem representation α_t and.α_{t+1}. In this case we do not know the values of *a*, *b* and *c* so we generate them, at random such that *(a+b+c) = (1-d).*

Figure 32 shows the results of performing 20 runs of the recurrence relations. As before each run continued for 10,00 evaluations, but a steady state was quickly reached. The values reached were the same as those in Table 14. As can be seen, after the initial variation the standard deviation of the linkage rapidly disappears, again more rapidly for increasing mutation.

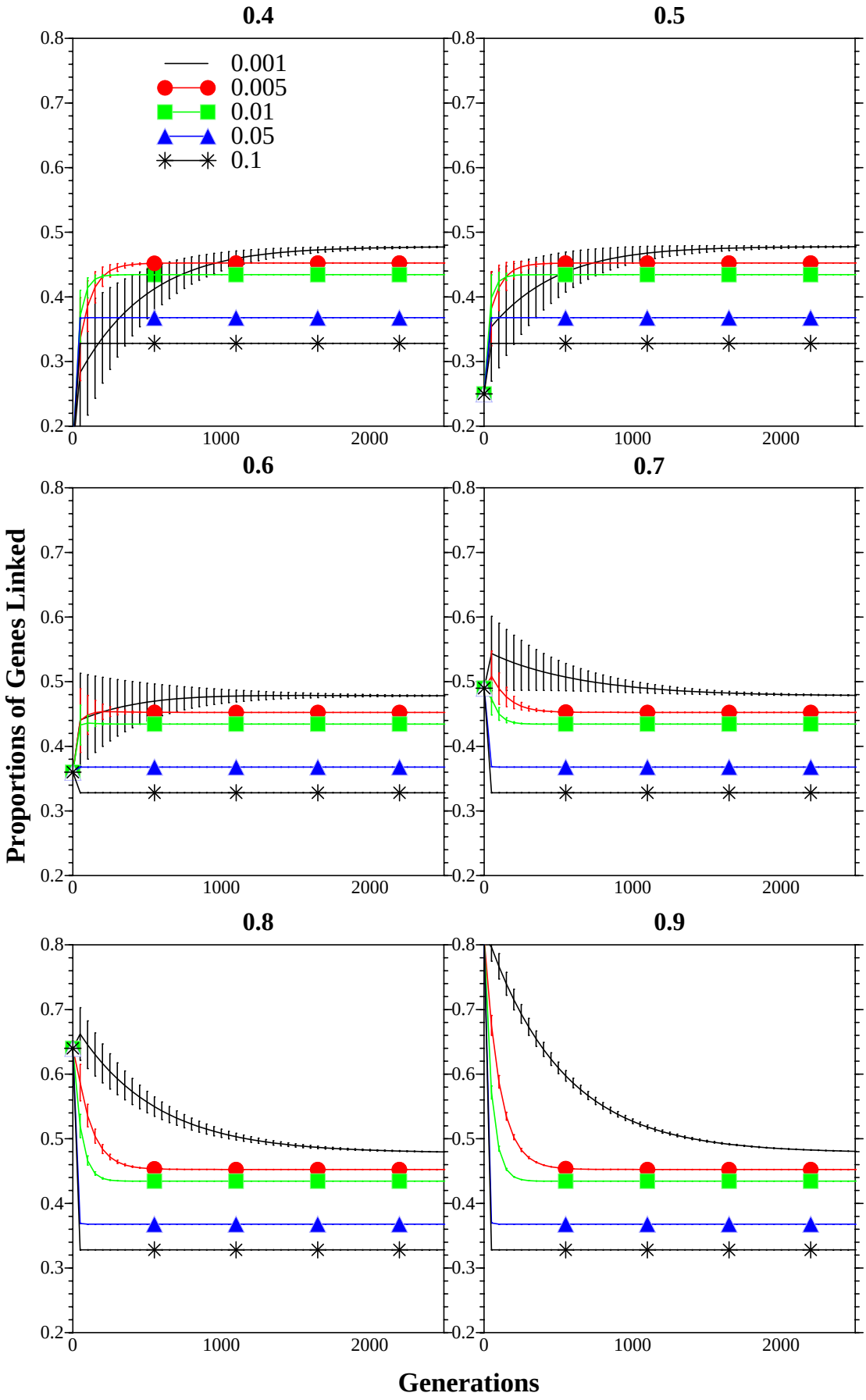


Figure 32: Evolution of linkage - random starting proportions

1.4. Derivation of Steady State Proportions

From the analytical results in the previous section we can see that a steady state is achieved at which these proportions will remain constant. If we set b'' = b and c'' = c, and then substitute (D70) from (D68) we get the following result.

$$(b\text{-}c)(1\text{-}d) = ab + b^2 - ac - c^2 - p(a + 2b + a + 2c) \quad (D71)$$

$$= a(b - c) + (b+c)(b-c) - 2p(a + b + c) \quad (D72)$$

$$= (1\text{-}d)(b\text{-}c)(1\text{-}2p) \quad (D73)$$

Since we have not made any assumptions about the mutation rate, we can say that in general $p \neq 0.5$. Since we also know that $a+b+c+d = 1$, setting either of the two remaining terms to zero reduces to:

$$b = c \quad (17)$$

This result makes sense since it preserves a natural symmetry, and holds regardless of starting conditions, providing a steady state is reached.

Substituting this equality into (1368), and noting that $a = (1\text{-}2b\text{-}d)$ gives us:

$$b(1\text{-}d) = b(1 - d - b) + p(1-p)(a^2 + d(1-d)) \quad (D74)$$

which gives

$$b^2 = c^2 = p(1-p)(a^2 + d(1-d)) \quad (18)$$

From (1266) , substituting $bc = b^2$, we get

$$a(1\text{-}d) = a(1\text{-}d)(1\text{-}p) - a^2p(1\text{-}p) + p(1\text{-}p)(a^2 + d(1\text{-}d)) + p^2d(1\text{-}d) \quad (D75)$$

$$\therefore \quad 0 = -pa(1\text{-}d) + pd(1\text{-}d) \quad (D76)$$

As noted above, we have made no assumptions about the mutation rate, and we know that for a non-zero mutation rate there will be a constant stochastic production of link bits set to *false*, so d cannot take the value unity. Therefore the equation above demonstrates the symmetrical property that

$$a = d \quad (19)$$

This result means that the value of both *a* and *d* must lie in the range 0.0-0.5 since all of the proportions derived are non-zero.

Finally substituting all of these expressions into the sum of proportions to unity yields an exact expression for the steady state proportion of linked genes in the absence of selection pressure for or against linkage

$$1 = a+b+c+d \quad (D77)$$

$$\therefore \quad 0 = d + \sqrt{dp(1\text{-}p)} - 0.5 \quad (D78)$$

Solving for the roots of a quadratic equation, and noting that we are looking for real values of d between 0.0 and 0.5, so we must take the positive second term, we get

$$d = \left(\frac{\sqrt{2 + p(1-p)} - \sqrt{p(1-p)}}{2} \right)^2 \quad (20)$$

Substituting values of p into this expression gives (to within 6 significant figures - the precision of the experiments) exactly the same results are were found empirically in the two runs.

Bibliography

Angeline, P. J. (1995). Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence*, pages 152–161. IEEE Press.

Arabas, J., Michalewicz, Z., and Mulawka, J. (1994). Gavaps - a genetic algorithm with varying population size. In *Proceedings of the First IEEE International Conference on Evolutionary Computing*, pages 73–78. IEEE Press.

Bäck, T. (1992a). The interaction of mutation rate, selection and self-adaptation within a genetic algorithm. In Manner, R. and Manderick, B., editors, *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, pages 85–94. Elsevier Science.

Bäck, T. (1992b). Self adaptation in genetic algorithms. In Varela, F. and Bourgine, P., editors, *Towards a Practice on Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 263–271. MIT Press.

Bäck, T. (1993). Optimal Mutation Rates in Genetic Search. In Forrest, S. editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 2-8. Morgan Kaufmann.

Bäck, T., Hofmeister, F., and Schwefel, H. (1991). A survey of evolution strategies. In Booker, L. B. and Belew, R., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann.

Bäck, T, Fogel, D.B,. and Michalewicz, Z. (1997) Handbook of Evolutionary Computation. Oxford University Press.

Baker, J.E. (1987). Reducing bias and inefficiency in the selection algorithm. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21. Lawrence Erlbaum.

Booker, L. (1992). Recombination distributions for genetic algorithms. In Whitley, L. D., editor, *Foundations of Genetic Algorithms 2*, pages 29–44. Morgan Kaufmann.

Bull, L. (1997). Evolutionary computing in multi agent environments: Partners. In Bäck, T., editor, *Proceedings of the Seventh International Conference on Genetic Algo rithms*, pages 370–377. Morgan Kaufmann.

Bull, L. and Fogarty, T. (1994). An evolutionary strategy and genetic algorithm hybrid: An initial implementation & first results. In Fogarty, T., editor, *Evolutionary Computation: Proceedings of the 1994 AISB Workshop on Evolutionary Computing*, pages 95–102. Springer Verlag.

Cobb, H. G. and Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 523–530. Morgan Kaufmann.

Corne, D., Ross, P., and H.-L.Fang (1994). Fast practical evolutionary timetabling. Technical Report GA-research Note 7, University of Edinburgh Dept. of Artificial Intelligence.

Cramer, N. J. (1985). A representation for the adaptive generation of simple sequential programs. In Grefenstette, J. J., editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 183–187. Lawrence Erlbaum.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In J.J.Grefenstette, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann.

Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press.

DeJong, K. (1975). *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan.

DeJong, K. and Sarma, J. (1992). Generation gaps revisited. In Whitley, L. D., editor, *Foundations of Genetic Algorithms 2*, pages 19–28. Morgan Kaufmann.

DeJong, K. A. and Spears, W. M. (1990). An analysis of the interacting roles of population size and crossover in genetic algorithms. In Schwefel, H.-P. and Manner, R., editors, *Proceedings of the First Conference on Parallel Problem Solving from Nature*, pages 38–47. Springer Verlag.

DeJong, K. A. and Spears, W. M. (1992). A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):1–26.

Eiben, A. E., Raue, P., and Ruttkay, Z. (1994). Genetic algorithms with multi-parent recombination. In Davidor, Y., editor, *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pages 78–87. Springer Verlag.

Eiben, A.E. and Schippers, C.A. (1996). Multiparents Niche: n-ary Crossovers on NK Landscapes. In Voigt, H., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature*, pages 319-328. Springer Verlag.

Eiben, A.E. and van der Hauw, J. (1997). Solving 3-SAT with Adaptive Genetic Algorithms. In Proceedings of the 4th IEEE Conference on Evolutionary Computation, pages 81-86. IEEE Service Center

Eiben, A. E., van der Hauw, J. and van Hemert, J.I. (1998). Graph Colouring with Adaptive Evolutionary Algorithms. Journal of Heuristics, 4:1. (in press)

Eiben, A. E., van Kemenade, C., and Kok, J. (1995). Orgy in the computer: Multi-parent reproduction in genetic algorithms. In Moran, F., Moreno, A., Morelo, J., and Chacon, P., editors, *Advances in Artifical Life: Proceedings of the Third European Conference on Artifical Life*, pages 934–945. Springer Verlag.

Eshelman, L. J., Caruana, R. A., and Schaffer, J. D. (1989). Biases in the crossover landscape. In Schaffer, J.D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 10–19. Morgan Kaufmann.

Eshelman, L. J. and Schaffer, J. (1991). Preventing premature convergence in genetic algorithms by preventing incest. In Booker, L. B. and Belew, R., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 115–122. Morgan Kaufmann.

Eshelman, L. J. and Schaffer, J. D. (1993). Crossover's niche. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 9–14. Morgan Kaufmann.

Eshelman, L.J. and Schaffer, J.D. (1994). Productive recombination and propogating and preserving schemata. In Whitley, L. D. and Vose, M., editors, *Foundations of Genetic Algorithms 3*, pages 299–313. Morgan Kaufmann.

Fisher, R.A. (1930) *The Genetical Theory of Natural Selection.* Oxford University Press

Fogarty, T. C. (1989). Varying the probablity of mutation in the genetic algorithm. In Schaffer, J., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 104–109. Morgan Kaufmann.

Fogel, L.J., Owens, A.J. & Walsh, M.J. (1966). *Artificial Intelligence Through Simulated Evolution*. John Wiley, NY

Forrest, S. and Mitchell, M. (1992). Relative building block fitness and the building block hypothesis. In Whitley, L. D., editor, *Foundations of Genetic Algorithms 2*, pages 109–126. Morgan Kaufmann.

Friesleben, B. and Hartfelder, M. (1993). Optimisation of genetic algorithms by genetic algorithms. In Albrecht, R., Reeves, C., and Steele, N., editors, *Artifical Neural Networks and Genetic Algorithms*, pages 392–399. Springer Verlag.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison Wesley.

Goldberg, D., Korb, B., Kargupta, H., and Harik, G. (1993). Rapid, accurate optimisation of difficult problems using fast messy genetic agorithms. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 56–64. Morgan Kaufmann.

Goldberg, D. E. (1985). Optimal initial population size for binary-coded genetic algorithms. Tcga report no. 85001, University of Alabama, Tuscaloosa, US.

Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93.

Goldberg, D. E., Deb, K., and Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362.

Goldberg, D. E., Korb, B., and Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530.

Grefenstette, J. J. (1986). Optimisation of control parameters for genetic algorithms. *Transaction on Systems, Man and Cybernetics*, 16(1):122–128.

Harik, G. and Goldberg, D. (1996). Learning linkage. In "Foundations of Genetic Algorithms 4",pp. 247-272, Morgan Kauffman.

Hesser, J. and Manner, R. (1991). Towards an optimal mutation probablity in genetic algorithms. In Schwefel, H.-P. and Manner, R., editors, *Proceedings of the First Conference on Parallel Problem Solving from Nature*, pages 23–32. Springer Verlag.

Hinterding, R., Michalewicz, Z., and Peachey, T. (1996). Self adaptive genetic algorithm for numeric functions. In Voigt, H., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature*, pages 420–429. Springer Verlag.

Hinterding, R., Michalewicz, Z, and Eiben, A.E. (1997). Adaptation in Evolutionary Computation: A Survey. In *Proceedings of the !997 IEEE International Conference on Evolutionary Computation*, pages 65-69. IEEE Press

Hoffmeister, F. and Bäck, T. (1991). Genetic self learning. In Varela, F. and Bourgine, P., editors, *Towards a Practice on Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 227–235. MIT Press.

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

Hordijk, W. and Manderick, B. (1995). The usefulness of recombination. In Moran, F., Moreno, A., Morelo, J., and Chacon, P., editors, *Advances in Artifical Life: Proceedings of the Third European Conference on Artifical Life*, pages 908–919. Springer Verlag.

Jones, T. and Forrest, S. (1995). Fitness distance correlations as a measure of problem difficulty for genetic algorithms. In Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann.

Julstrom, B. A. (1995). What have you done for me lately?: Adapting operator probabilities in a steady-state genetic algorithm. In Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 81–87. Morgan Kaufmann.

Kakuza, Y., Sakanashi, H., and Suzuki, K. (1992). Adaptive search strategy for genetic algorithms with additional genetic algorithms. In Männer, R. and Manderick, B., editors, *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, pages 311–320. Elsevier Science.

Kaneko, K. and Ikegami, T. (1992). Homeochaos: Dynamic stability of a symbiotic network with population dynamics and evolving mutation rates. *Physica-D*, (56):406–429.

Kargupta, H. (1996). The gene expression messy genetic algorithm. In *Proceedings of the Third IEEE International Conference on Evolutionary Computing*, pages 814–819. IEEE Press.

Kargupta, H. and Goldberg, D. E. (1994). Decision making in genetic algorithms: A signal-to-noise perspective. IlliGAL Report 94004, University of Illinois at Urbana-Champaign.

Kauffman, S. A. (1993). *The Origins of Order*. Oxford University Press.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by simulated anealing. *Science*, (220):671–680.

Koza, J. R. (1989). Hierarchical genetical algorithms operating on populations of computer programs. In *Proceedings of the 11th Joint International Conference on Artificial Intelligence*, volume 1, pages 768–774. Morgan Kaufmann.

Lee, M. and Takagi, H. (1993). Dynamic control of genetic algorithms using fuzzy logic techniques. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 76–83. Morgan Kaufmann.

Levenick, J. R. (1995). Megabits: Generic endogenous crossover control. In Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 88–95. Morgan Kaufmann.

Lis, J. (1996). Parallel genetic algorithm with dynamic control parameter. In *Proceedings of the Third IEEE International Conference on Evolutionary Computing*, pages 324–329. IEEE Press.

Manderick, B., de Weger, M., and Spiessens, P. (1991). The genetic algorithm and the structure of the fitness landscape. In Booker, L. B. and Belew, R., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150. Morgan Kaufmann.

Mattfeld, D., Kopfer, H., and Bierwirth, C. (1994). Control of parallel population dynamics by social-like behaviour of ga-individuals. In Davidor, Y., editor, *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pages 16–25. Springer Verlag.

Maynard-Smith, J. (1978). *The Evolution of Sex*. Cambridge University Press.

Maynard-Smith, J., and Szathmary, E. (1995). *The Major Transitions in Evolution.* W.H. Freeman.

Mitchell, M., Forrest, S., and Holland, J. (1992). The royal road for genetic algorithms: Fitness landscapes and GA performance. In Varela, F. and Bourgine, P., editors, *Towards a Practice on Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 245–254. MIT Press.

Muhlenbein, H. (1992). How genetic algorithms really work : 1. mutation and hillclimbing. In Manner, R. and Manderick, B., editors, *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, pages 15–25. Elsevier Science.

Muller, H. (1964). The relation of recombination to mutational advance. *Mut. Res.*, (1):2–9.

Paredis, J. (1995). The symbiotic evolution of solutions and their representations. In Eshelman, L. J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 359–365. Morgan Kaufmann.

Potter, M. A. and DeJong, K. (1994). A cooperative coevolutionary approach to function optimisation. In Davidor, Y., editor, *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pages 248–257. Springer Verlag.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart.

Rudolph, G. and Sprave, J. (1995). A cellular genetic algorithm with self adjusting acceptance threshold. In *Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering: Innovations and Applications*, pages 365–372. IEE.

Schaefer, C. (1987). The argot system: Adaptive representation genetic optimising technique. In J.J.Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 50–58. Lawrence Erlbaum.

Schaffer, J. and Morishima, A. (1987). An adaptive crossover distribution mechanism for genetic algorithms. In J.J.Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 36–40. Lawrence Erlbaum.

Schaffer, J. D., Caruana, R. A., Eshelman, L. J., and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimisation. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 51–60. Morhan Kaufmann.

Schaffer, J. D. and Eshelman, L. J. (1991). On crossover as an evolutionarily viable strategy. In Belew, R. and Booker, L., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann.

Schlierkamp-Voosen, D. and Muhlenbein, H. (1994). Strategy adaptation by competing subpopulations. In Davidor, Y., editor, *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pages 199–209. Springer Verlag.

Schwefel, H. P. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *ISR*. Birkhaeuser, Basel/Stuttgart.

Schwefel, H. P. (1981). *Numerical Optimisation of Computer Models*. John Wiley and Sons, New York.

Sebag, M. and Schoenauer, M. (1994). Controlling crossover through inductive learning. In Davidor, Y., editor, *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pages 209–218. Springer Verlag.

Smith, J. and Fogarty, T. (1995). An adaptive poly-parental recombination strategy. In Fogarty, T. C., editor, *Evolutionary Computing* ,LNCS 993,pages 48–61. Springer Verlag.

Smith, J. and Fogarty, T. (1996a). Adaptively parameterised evolutionary systems: Self adaptive recombination and mutation in a genetic algorithm. In Voigt, Ebeling, Rechenberg, and Schwefel, editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature*, pages 441–450. Springer Verlag.

Smith, J. and Fogarty, T. (1996b). Recombination strategy adaptation via evolution of gene linkage. In *Proceedings of the Third IEEE International Conference on Evolutionary Computing*, pages 826–831. IEEE Press.

Smith, J. and Fogarty, T. (1996c). Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of the Third IEEE International Conference on Evolutionary Computing*, pages 318–323. IEEE Press.

Smith, J. and Fogarty, T. (1996d). Evolving Software Test Data- GA's learn Self Expression. In Fogarty T. ,editor, *EvolutionaryComputing,* LNCS 1143,pages 137-146, Springer Verlag

Smith, J. and Fogarty, T. (1997a). Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2) : 81-87

Smith, J., Bartley, M. and Fogarty, T. (1997). Microprocessor Design Verification by Two-Phase Evolution of Variable Length Tests. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 453-458. IEEE Press

Smith, R. E. (1993). Adaptively resizing populations: An algorithm and analysis. Technical Report TCGA Report # 93001, University of Alabama, Box 870278, Tuscaloosa, Alabama 35487.

Smith, R. E. and Smuda, E. (1995). Adaptively resizing populations: Algorithm, analysis and first results. *Complex Systems*, 9(1):47–72.

Spears, W. M. (1992). Crossover or mutation. In Whitley, L.D., editor, *Foundations of Genetic Algorithms 2*, pages 220–237. Morgan Kaufmann.

Spears, W. M. (1995). Adapting crossover in evolutionary algorithms. In McDonnel, J., Reynolds, R., and Fogel, D., editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press.

Spears, W. M. and DeJong, K. (1990). An analysis of multi point crossover. In Rawlins, G., editor, *Foundations of Genetic Algorithms*, pages 301–315. Morgan Kaufmann.

Spears, W. M. and DeJong, K. A. (1991). On the virtues of parameterized uniform crossover. In Belew, R. and Booker, L., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–237. Morgan Kaufmann.

Srinivas, M. and Patnaik, L. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4):656–667.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann.

Syswerda, G. (1991). A study of reproduction in generational and steady state genetic algorithms. In Rawlins, G. J., editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann.

Syswerda, G. (1993). Simulated crossover in genetic algorithms. In Whitley, L.D. editor, *Foundations of Genetic Algorithms 2*, pages 239–255. Morgan Kaufmann.

Thierens, D. and Goldberg, D. (1993). Mixing in genetic algorithms. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 38–45. Morgan Kaufmann.

van Kemenade, C. (1996). Explicit filtering of building blocks for genetic algorithms. In Voigt, H.-M., Ebeling, W., I.Rechenberg, and Schwefel, H.-P., editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature*, pages 494–503. Springer Verlag.

Vavak, F. and Fogarty, T.C. (1996). Comparison of steady state and generational genetic algorithmsfor use in nonstationary environments. In *Proceedings of the Third IEEE International Conference on Evolutionary Computing*, pages 192–195. IEEE Press.

Weinberger, E. (1990). *Correlated and Uncorrelated Fitness Landscapes, and How to Tell the Difference.*Biological Cybernetics (63):325-336

White, A. and Oppacher, F. (1994). Adaptive crossover using automata. In Davidor, Y., editor, *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pages 229–238. Springer Verlag.

Whitley, D. and Kauth, J. (1988). Genitor: A different genetic algorithm. In *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, pages 118–130.

Wolpert, D.H, Wolpert. and Macready, W.G. (1995). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.