X

# Intelligent Genetic Algorithms in Evolutionary Computation

## Part II. Application to Combinatorial Multimodal Optimization Problems

Liwen He and Neil Mort

Department of Automatic Control and Systems Engineering

University of Sheffield, UK

Research Report No. 690

**Abstract**: Combinatorial Multimodal Optimization Problems (CMOP) arising in the scheduling of manufacturing systems involve the determination of multiple integer solution vectors that optimize a given objective function with regard to some definite constraints. The genetic algorithm is an efficient computational paradigm to search such a large optimization space for the best solutions. But simple genetic algorithms are notorious for their 'premature convergence' to a unimodal (global or local) optimum because of genetic drift. Following the previous discussion in Part I, a new Intelligent Genetic Algorithm is developed to include spatial structured population, relative fitness vector, absolute fitness value with dynamic fitness sharing function, super conservative selection strategy, intelligent recombination through speciation, optimal outbreeding and neutral mutation. Experimental results and simple fitness landscape analysis illustrate that intelligent genetic algorithms can effectively solve a typical combinatorial multimodal optimization problem, which is a challenging problem for GA applications. These advanced intelligent genetic algorithms present a prospective arena for multi-objective optimization [Fonseca & Fleming., 1995a,b; Shaw & Fleming,1996] and multimodal optimization problems in evolutionary computation.

**Keywords**: Intelligent Genetic Algorithms, Combinatorial Multimodal Optimization, Spatially Structured Population, Intelligent Recombination, Neutral Mutation

**Contact Details (both authors):**

☎ +44 114 222 5601

FAX +44 114 222 5602

💻 n.mort@sheffield.ac.uk

April 1998

## 1. Introduction

Many problems in the scheduling of manufacturing systems and artificial intelligence can be regarded as combinatorial multimodal optimization problems, which require the determination of multiple integer solution vectors that maximize (minimize) a given objective function with regard to some definite set of constraints [He & Mort, 1997a, b; Luling, R & *et al*, 1996]. Problems of this class, which are characterized by their high discreteness, high dimension and noise, are quite different from multimodal continuous function optimization problems [Goldberg & Richardson, 1987], [Deb & Goldberg, 1989]. The combinatorial multimodal optimization problem is arguably one of the most deceptive problems for GA applications [Ohkura and Ueda, 1995]. As far as we know, very few contributions have either addressed this problem or have solved it satisfactorily.

Intuitively, this problem can be solved by searching all possible elements in the finite solution space for all multimodal optimal solutions. However, the solution space increases exponentially with input size, and intolerable computation times result. This problem is known as a NP-hard combinatorial optimization problem.

The genetic algorithm is a well-known stochastic optimization search method which is based on a global search procedure . When we use a simple genetic operator (selection , multi-point crossover and mutation) to solve a typical combinatorial multimodal optimization problem, only a unimodal (global or local) optimal solution is obtained for every search procedure [ He & Mort, 1997a]. The reason is that, as there is no selective pressure on any of the peaks for the finite population in the fitness landscape, the population converges to one alternative or another randomly. This problem is known as genetic drift -- stochastic errors in sampling caused by small population size [Goldberg, 1989]. In other words, this difficulty comes from the fundamental theorem of genetic algorithms [Holland, 1975; De Jong, 1975; Goldberg, 1989] which tells us that exponentially increasing trials will be given to the best schemata observed, thus losing some potentially useful schemata in the evolution.

Therefore , the problem is how to reduce the effect of this genetic drift and enable stable sub-populations to form around different fitness peaks, and lead populations to move from local optima to global optima.

Previous works which have attempted to address these issues are:

(1) De Jong (1975) used a *crowding method* to let individuals replace the most similar genotype strings measured by the Hamming distance in an overlapping population. This replacement tends to maintain diversity within a population and reserve room for other species in a limited-size population.

(2) Goldberg and Richardson (1987) introduced the idea of a *sharing function* to permit the formation of stable subpopulations of different strings within a genetic algorithm, thereby permitting the population to investigate many peaks in parallel.

(3) Deb and Goldberg (1989) introduced the natural concepts of *niche* and *species* into a population of strings to optimize a number of multimodal functions; phenotypic and genotypic sharing and a mating restriction in the recombination strategy are implemented to improve genetic search performance. Simulation results showed that a GA with a 'sharing function' is able to converge and distribute trials to all the peaks of the test functions.

(4) Collins and Jefferson(1991) and Davidor *et al* (1993) implemented the Parallel Genetic Algorithm, where spatially structured populations are developed according to some topology and distance metric, to maintain genetic diversity, allow niche differentiation and several competing subpopulations to search more effectively, thus finding multiple global optima in the same run.

(5) Srinivas and Patnaik(1994) recommended the use of adaptive probabilities of crossover and mutation to realize the twin goals of maintaining diversity in the population and sustaining the convergence capacity of the GA. Their experiments demonstrate that the adaptive genetic algorithm represents the first step in self-adaptation for locating the global optima in a multimodal landscape.

(6) Jelasity and Dombi(1995) developed a niching technique called GAS (here 'S' stands for Species) which creates a subpopulation structure using a *radius function* and a 'cooling' method similar to simulated annealing. An NP-completeness combinatorial task is examined, where 93 percent optimal solutions and many near-optimal solutions were found per run.

(7) Schneider *et al* (1996) provided a simple evolution strategy based on a mutation and selection scheme without recombination for a multimodal optimization search space, and proposed a relation between the search time and the number of optimal offspring.

These methods can be categorized into three types of solution: niche techniques, parallel GA (spatially structured populations) and adaptive GA. These niche techniques distribute sub-populations across multiple peaks of search space. The sharing method is similar to ecological competition to distribute individuals across fitness peaks in proportion to the heights of the peaks. Adaptive GA is the alternation of crossover rate and mutation rate for a specific problem. Both niche techniques and adaptive GA require a previous understanding of the number and the distribution of fitness peaks, and need careful setting of various parameters. Spatial Structuring has proven very reliable and efficient in exploring multimodal fitness peaks, and it also needs fewer assumptions than niche methods. However, when we use only these niching or spatial structuring methods for the combinatorial multimodal optimization problem discussed, the solutions are disappointing.

All the success of niche techniques , adaptive GA and parallel GA in preserving deep genetic diversity overlook the main method that nature uses to produce biodiversity: **sexual selection** based on *assortative mating*, *selective mating* and *speciation* [Miller, 1994]. In assortative mating, only similar animals pair up, implying animals prefer the mate's similarity. In selective mating, female animals will favor the mate's quality displaying attractiveness, novelty and some desired trait. In speciation, a lineage is divided into reproductively isolated sub-populations (species) that have no interbreeding.

Traditional genetic algorithms use recombination (crossover) under random mating, which results in the production of many useless offspring, and the disruption of good schemata at early stage thus eroding genetic diversity.

In this paper, based on the biological background discussed in Part I, we propose an original mechanism -- **Intelligent Genetic Algorithms** where the interactions between natural selection and sexual selection in biology are integrated with evolutionary computation perspectives as a process of search, diversification and optimization. The power of intelligent genetic algorithms is examined via a typical combinatorial multimodal optimization from a manufacturing scheduling problem.

## 2. Combinatorial multimodal optimization problems

The general mathematical model of a typical combinatorial multimodal optimization problem can be stated as follows :

Let $n$ be a positive integer, $V=\{v_1, v_2,..., v_n\}$ be the search vector, $W$ be the solution space which is the combinatorial set of all elements in the search vector, $S$ be a domain defined as the discrete set of all vectors $x$ in the solution space that satisfies a set of constraints, and $f: W \rightarrow R$ be the objective function from $W$ onto a completely ordered set $R$. We call $x \in S$ a *feasible solution* and $f(x)$ the fitness value of $x$. Without losing generality, we are maximizing problem. A feasible solution $x$ is better than $x' \in S$ if $f(x) > f(x')$. We search for an optimal solution $x^*$, which satisfies the condition: $f(x^*) \geq f(x)$ for all $x \in S$. We assume $S$ finite and non-empty, and the set of all best feasible solutions, $S^* \subset S$ finite and non-empty. If the number of entries in the optimal solution space $S^*$ is greater than or equal to two, this problem is named as a combinatorial multimodal optimization problem. Minimization problems can be stated and dealt with similarly. This problem is NP-hard.

Now we consider a practical scheduling optimization problem [Fanti, et al, 1996, He & Mort, 1997a, b] as follows:

A batch processing machine (BPM) which consists of a number of identical servers is regarded as a processor that can operate on a batch of jobs simultaneously with common start and end times.

There are $n$ types of jobs available to be processed on $m$ identical servers, where each job type requires an appropriate tool to be fixed on the server. Furthermore there exists no priority order within job types. Once the process begins, no job can be interrupted until the entire batch is completed. The processing time of a batch of jobs is constant and independent of job type and on the number of jobs in the batch.

Since the servers operate synchronously, the BPM alternates between running intervals (operation steps) and breaks for unloading the old tool and choose a new tool to process the next job. Moreover, the duration of each operation step is fixed and coincides with the time necessary for any server to complete a job. It is assumed that the time to load a new tool on a server and the time for the corresponding unloading operation are constant, and that the duration of the breaks for the BPM depends on the tool change time. Obviously, one tool changing (for loading and unloading tool ) time is necessary for each job type (i.e. n job types need at least n tool change times). If some jobs cannot be completed in one server, they have to be divided into two segments of varying size and another tool change operation is required.

Since the operation for replacing a tool is time-consuming, the scheduling objective is to determine the batches of different type jobs and their loading sequence to minimize the tool replacing times and to maximize the BPM's utilization.

In a particular case study describing a shoe factory (Fanti, et al, 1996), there are $n$ (=21) job types and tools and $m$ (=7) servers available. A production mix of $n$ types of jobs is described as a search vector:

$V=\{v_1, v_2,..., v_n\}=\{176\ 380\ 216\ 688\ 144\ 497\ 153\ 12\ 714\ 231\ 310\ 170\ 6\ 660\ 50\ 114\ 282\ 12\ 454\ 128\ 266\}$ ( i.e. the quantity of job type #2 is "380" ).

Since for each operation step the BPM can process *m* jobs at most, the length **P** of operation steps necessary for the BPM to complete the whole production mix must satisfy the relation: $\mathbf{P} \geq \text{sum}(V) / m = 5663/7 = 809$.

In order to balance the load among the servers and minimize the tool change times, different jobs are combined into different batches, where each batch can be completed in one server within the entire length of **P** operation steps. The sum of the number of this batch of jobs should be close to the **P** and no more than **P,** and we define this batch of jobs as a "*combination*". For example, when P is chosen as 809, if job '497' and '310' is processed within one server, and 497 + 310 (= 807) < 809 is valid, then we can name job '497' and '310' as a *combination*.

To represent the solution with binary strings, we give the following correspondence relation:

| machine j | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| string | 001 | 010 | 011 | 100 | 101 | 110 | 000, 111 |

The bitlength of the chromosome is : *bitlength* = $r \times n = 3 \times 21 = 63$.

A binary string whose length is 63 is randomly generated, for example:

b=000 101 111 010 110 001 101 100 010 100 001 011 110 110 010 100 111 101 100 000 101

*V*={176 380 216 688 144 497 153 12 714 231 310 170 6 660 50 114 282 12 454 128 266}

We map the 63-bit binary string into the loading matrix $\mathbf{M}_1$ . It is easy to compute the binary string's fitness value.

The fitness function f(*x*) of a feasible solution *x* is the number of combinations searched in the solution *x*. Obviously because of the limitation of BPM's processing capability, the constraint condition for a feasible solution is:

$$\sum_{j=1}^{Nc} (\mathbf{P} - \text{sum}(C_j)) \leq b$$

and $b = P \times m - \text{sum}(V)$

where $N_c$ is the number of combinations in a feasible solution; $C_j$ is the j-th combination.

When the length P of operation steps for this production mix is chosen as '811', the maximization of processor utilization is achieved: the minimum tool changing times is "22" . Four different but equally valued optimal results have been obtained (For detail see [He & Mort, 1997a, b].), two of them are:

$$\begin{array}{lccccccc}
\text{machine} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\mathbf{M}_1 & \left[\begin{matrix} 497 \\ 310 \\ 0 \\ 0 \end{matrix}\right. & \begin{matrix} 688 \\ 714 \\ 50 \\ 0 \end{matrix} & \begin{matrix} 170 \\ 0 \\ 0 \\ 454 \end{matrix} & \begin{matrix} 12 \\ 231 \\ 114 \\ 266 \end{matrix} & \begin{matrix} 380 \\ 153 \\ 12 \\ 0 \end{matrix} & \begin{matrix} 144 \\ 660 \\ 6 \\ 0 \end{matrix} & \left.\begin{matrix} 176 \\ 216 \\ 282 \\ 128 \end{matrix}\right] \\
\text{sum} & \mathbf{\mathit{807}} & 1452 & 170 & \mathbf{\mathit{811}} & \mathbf{\mathit{811}} & \mathbf{\mathit{810}} & \mathbf{\mathit{802}} \\
\text{Combination} & C_1 & & & C_2 & C_3 & C_4 & C_5 \\
\text{phenotype schema} & A & & & B & C & D & E
\end{array}$$

$$\begin{array}{l|ccccccc|}
\text{machine} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
 & 216 & 688 & 497 & 12 & 380 & 144 & 176 \\
 & 0 & 714 & 310 & 231 & 153 & 660 & 454 \\
\mathbf{M}_2 & 0 & 0 & 0 & 114 & 12 & 6 & 128 \\
 & 0 & 0 & 0 & 170 & 266 & 0 & 50 \\
 & 0 & 0 & 0 & 282 & 0 & 0 & 0 \\
\text{sum} & 216 & 1402 & \mathbf{\mathit{807}} & \mathbf{\mathit{809}} & \mathbf{\mathit{811}} & \mathbf{\mathit{810}} & \mathbf{\mathit{808}}
\end{array}$$

(Note: We notice that job '714' should be divided into two segments because it cannot be combined with any other job to make a batch close to **P** and no more than **P**. That is to say, only one another tool change time is necessary for this production mix. The minimum number of tool change times for the **P**(=811) operation steps is n + 1 = 22).

According to the loading matrix $\mathbf{M}_1$, it is clear that $C_1$=(497 310), $C_2$=(12 231 114 454), $C_3$=((380 153 12 266), $C_4$=((144 660 6) and $C_5$=(176 216 282 128) are different combinations, and the condition:

$$\sum_{j=1}^{Nc} (\mathbf{P} - \text{sum}(C_j)) = (811-\text{sum}(C_1))+(811-\text{sum}(C_2)+(811-\text{sum}(C_3))+ (811-$$

sum($C_4$))+(811-sum($C_5$)) = 14 $\leq$ b (where = P m -sum($V$) = 811 $\times$ 7 - 5663 = 14) holds, therefore the loading matrix $\mathbf{M}_1$ is a feasible solution. Its fitness function f($\mathbf{M}_1$) is the number of combination , equal to 5.

Furthermore, the job '714' should be divided into two segments of size '73' and '641'. This adds another tool change operation for job type '714'. The schedule for server 2 is to process job '688', '50' and '73' ( 688 + 50+73 = 811,), and the schedule for machine 3 is process job '641' and '170' ( 641+170 = 811). Therefore all jobs are almost evenly distributed into different machines, all the servers complete their jobs at nearly the same time and the minimum number of tool changes is 22.

Following a similar analysis procedure, we find that the loading matrix $\mathbf{M}_2$ is a feasible solution. Its fitness function f($\mathbf{M}_2$) is also equal to 5

Assume that $x$ is a feasible solution, $S$ is the solution space, since the condition f($\mathbf{M}_1$)$\geq$f($x$) and f($\mathbf{M}_2$)$\geq$ f($x$) for all $x \in S$ holds, the set of all optimal feasible solutions $S^* \subset S$ is finite and non empty, and the number of entries in optimal solution space $S^*$ is at least equal to two, so this problem is a NP-hard combinatorial multimodal optimization problem.

Before we use Intelligent Genetic Algorithms to solve this typical combinatorial multimodal optimization problem, we must identify the solution space $S$ as a phenotype space; the combination is named as a *phenotype schema*. For example, the phenotype schema A can be expressed as { * (497+310) * } , '*' means 'do not care', we just need a combination (497+310), we do not care about where it is and the circumstances of other combinations. The phenotype space is mapped into a binary genotype space with $l$ - dimension according to some kind of topology (see [He & Mort, 1997a), $l$ is the length of the binary representation.

## 3. Intelligent Genetic Algorithms

Natural selection and sexual selection are the twin engines that drive evolution in the natural world. 'Natural selection' was dominant and sexual selection was overlooked in biology for over a hundred years following Darwin (1871). In the evolutionary computation community, natural selection is mature in genetic algorithms simulation, while GA simulations of sexual selection have emerged only recently [Todd, 1991]. Sexual selection through evolvable mate choice can provide a robust and effective solution for problems that natural selection finds difficult or impossible, such as preserving genetic diversity, discovering evolutionary innovations, and guiding a search from local optima to global optima.

Intelligent genetic algorithms are proposed to combine both natural selection and sexual selection with evolutionary computation, such as spatially structured population, super conservative selection strategy, dynamic sharing fitness function, individual's relative fitness vector, intelligent recombination and neutral mutation.

Note that, in sexual selection, assortative mating searches for diversification and optimal outbreeding through automatic niching and speciation to maintain genetic diversity, whereas selective mating guides the stabilizing selection that reduces stochastic sampling errors and noise arising in the mapping from genotype to fitness [Miller, 1994].

Through the interaction of both natural selection and sexual selection, intelligent genetic algorithms create speciation automatically based on the individual's relative fitness vector and the degree of similarity along certain phenotypic dimensions make sub-populations explore more 'intelligently' around each peak in the fitness landscape by directing future evolution according to previous information .

### 3.1. Spatially Structured Population through Phenotypic Analysis

The concept of 'spatially structured population' comes from Parallel Genetic Algorithms (Muhlenbein, 1991). The spatial population structure can introduce diversification naturally, and facilitate the diversity of the whole population more effectively in evolutionary computation, typically through local competition and local recombination.

In intelligent genetic algorithms, we construct a kind of 'spatially structured population' based on similar *phenotypic schemata* through phenotypic analysis: around any schemata, those individuals which contain these common schemata will naturally construct a subpopulation. So, the number of sub-populations is equal to the number of phenotypic schemata searched so far. This kind of sub-population is semi-isolated, that is to say: any individual with more than one phenotypic schemata will be a member of more than one sub-population, the more phenotypic schemata it has, the higher fitness value it has , and the more subpopulations it belongs to.

For example, {* (497 310) *} is a phenotype schema A, then all individuals who have this phenotype schema will automatically form a subpopulation A.

It will be shown that this spatially structured population proves both reliable and efficient for intelligent genetic algorithms to explore multimodal optima in the fitness landscape.

## 3.2. Individual's Relative Fitness Vector

Since an individual is characterized by the phenotype schema, its raw fitness value is decided by the number of phenotypic schemata in the individual. It is seen that when a *phenotype schemata vector* is built up as a database of all searched phenotype schemata in the entire population, then the individual's phenotype can be mapped into a binary form in relation to the schemata vector according to the following principle :

the schemata vector $\boldsymbol{M} = [\ M_1\ \ M_2 \ldots\ \ M_q\ ]$ ; for $M_i$ ( $i = 1, \ldots, q,\ q \leq m$ ) is the distinct schema searched in the entire population;

the phenotype of individual *j* is converted into its relative fitness binary vector:

$fv\,(j) = [b_1\ \ b_2\ \ \ldots\ \ b_q]$ ; and $b_i$ ( $i = 1, \ldots, q$) is a binary bit ,

$$b_i = \begin{cases} 1 & \text{if the phenotype of individual j includes the schema Mi ;} \\ 0 & \text{if the phenotype of individual j does not include the schema Mi} \end{cases} \qquad (1)$$

Then the individual's relative fitness value is represented by a binary vector.

According to the Definition 7 in Part I, the individual *j* is characterized by its relative fitness vector. If it is non-dominated by any other individuals in the entire population, it is defined as an *non-dominated individual.*

For example, the fitness vectors of individual *a* , *b* and *c* are *fv*(a), *fv*(b) and *fv*(c), and

$$fv(a) = [\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1];$$

$$fv(b) = [\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1];$$

$$fv(c) = [\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1];$$

It is easy to see that individual *a* dominates individual *b* , and individual *a* , individual *c* are non-dominated with each other. If both *a* and *c* are non-dominated by any other individuals in the current population, they are *non-dominated individuals.*

This individual's *relative fitness vector* by combining Pareto-based ranking with individual's phenotype information produces a suitable high-dimensional (relative) fitness landscape, facilitating the 'super-conservative selection' strategy, and 'hybrid drift', thus helping population explore new adaptive zones. This will be discussed in the following section.

## 3.3. Initial Population, Subpopulation, Entire Population and Constraint Population

We do not know the exact number of multiple optima and the actual fitness landscape in this problem, so the initial entire population size is chosen as Nchrom=100.

Since all subpopulations construct an entire population, it is obvious that the size of entire population is very great. So a new concept of '*constraint population*' ***chrom*** is introduced to limit the 'entire population' size to a relatively small scale: all superior non-dominated individuals (***chromA***) are part of the 'constraint population' ***chrom***, the remainder are selected from the current entire population using a stochastic sampling method [Baker, 1987]. The subpopulation in the next generation is selected from the constraint population according to their similar phenotype schemata; the dynamic fitness sharing function operates on the 'constraint population'. 'Entire population' includes all the offspring generated from every subpopulation.

So the constraint population is the central population. At first, the constraint population size equals to the initial population size. As the computation procedes, if the size of all superior non-dominated individuals is greater than initial population size, the constraint population size is increased by 100 so as to keep some 'room' for natural selection through stochastic universal sampling--allowing individuals with higher absolute fitness value with dynamic sharing to obtain more copies in the constraint population of the next generation.

## 3.4 Absolute Fitness Value with Dynamic Sharing Function

The objective function is the number of phenotypic schemata in the individual, so we name *absolute fitness value* as the value of the objective function. The best individuals have the maximum absolute fitness value. Clearly, the set of all best individuals in optimal solution should be the subset of non-dominated individuals in relative fitness landscape.

It is convenient to calculate the absolute fitness value fa(j) of an individual *j* by summing all the elements in the relevant individual's relative fitness value:

$$\mathrm{fa}(j) = \mathrm{sum}\,(\mathit{fv}(j)) \qquad (3)$$

The difficulty is how to employ this absolute fitness value in the multimodal optimization problem. Goldberg and Richardson (1987) invented a sharing function to impose niche and speciation on the string based on some measure of their distance from each other. This sharing function is a method of determining the degradation of an individual's payoff due to its similarity with all other individuals in the entire population. Deb and Goldberg (1989) considered two forms of sharing functions, so called phenotypic and genotypic sharing, and implemented them with a mating restriction scheme to improve the on-line performance in continuous multimodal functions.

Phenotypic sharing seems to work better than genotypic sharing (Deb and Goldberg, 1989; Goldberg, Deb, & Horn,1992) because more or less error may be induced while an individual is mapped from phenotypic space to genotype space, such as binary encoding of phenotype; phenotypic sharing is more directly utilized also. In intelligent genetic algorithms, the *dynamic phenotypic sharing function* of individual *j* has been created as follows:

$$\mathrm{fis}(j) = \mathrm{fa}(j)\,\gamma^{\,\mathrm{sh}(j)} \qquad (4)$$

where, $0 < \gamma < 1$; *sh*( j ) is the similarity coefficient of individual *j* with all other individuals in the constraint population, and is obtained by calculating the phenotypic similarity with all other individuals (excluding itself):

$$\mathrm{sh}(j) = \sum_{i=1}^{Ncps} [\mathit{fv}(j)\,\mathit{fv}(i)'] \qquad \text{but} \quad i \neq j \qquad (5)$$

where $N_{cps}$ is the constraint population size. ( ' ′ ' means ' the transpose of ').

This dynamic phenotypic sharing function requires few assumptions about the number and distribution of fitness peaks than the previous 'sharing function' [Goldberg and Richardson 1987; Deb and Goldberg, 1989] where many parameters such as the distance metric $\sigma_{share}$, and the power of the sharing function $\alpha$ should be established because it is not possible to know much a priori knowledge about the fitness landscape. The only parameter $\gamma$ is normally selected as 0.85 to determine the payoff of individual's absolute fitness value due to its similarity to all other individuals in the constraint population.

Absolute fitness value with dynamic sharing function works on the basis of relative fitness vector, fights against the 'genetic drift', prevents 'premature convergence', and permits a stable subpopulation to investigate different fitness peaks in parallel.

### 3.5 Super Conservative Selection Strategy

Eshelman (1991) described a 'conservative selection' strategy as one that always preserves the best individuals found so far.

In intelligent genetic algorithms, a 'super conservative selection strategy' is developed, where different non-dominated individuals should be kept in the *constraint population* of the next generation after selection until they are dominated by other new-born individuals with higher performance. The remainder of the constraint population are selected from the entire population using a Stochastic Universal Sampling method [Baker, 1987] in order to reduce the sampling bias and increase efficiency.

In practice, if all non-dominated individuals are included in the next generation, their population size increases greatly, and becomes larger than the initial population size. Actually, most of these non-dominated individuals contain few schemata; they are inferior individuals with relatively smaller absolute fitness values, and might not continue to evolve. So these individuals are ranked according to their absolute fitness value. The following 'ranking' method is used here: non-dominated individuals with greatest absolute fitness value *Ma* so far are ranked as 'first class' individuals, and non-dominated individuals with absolute fitness value (*Ma-k*) (Ma>k, for positive integer k) are ranked as *k* class individuals. In order to speed up optimization, only distinct non-dominated individuals with top *k* absolute fitness values will be kept in the next generation. The choice of *k* is determined by the initial population size and the number of non-dominated individuals; the greater the value of *k*, the more genetic diversity will be preserved, and the more computation complexity will be required. For this combinatorial multimodal optimization problem , we choose k=2; that is to say: only first-class and second-class non-dominated individuals searched so far are guaranteed to be preserved in the next generation in order to maintain suitable genetic diversity and accelerate convergence speed.

Super Conservative Selection Strategy lets intelligent genetic algorithms exhibit *explicit parallelism* by always keeping suitable distinct non-dominated individuals, helping preserve appropriate genetic diversity, and enhancing speciation and parallel search in evolutionary computation.

### 3.6 Intelligent Recombination with Neutral Mutation

In simple genetic algorithms, recombination under random mating results in interbreeding across different species, thereby producing many useless offspring, disrupting good schemata at early stage and eroding genetic diversity.

In nature , animals pair up, breeding occurs within the same species. Animals prefer the similarity in mates. Female animals will favor quality in mates that display attractiveness, novelty and some desired trait [Miller, 1992].

In intelligent genetic algorithms, recombination is limited within subpopulations whose individuals share some kind of phenotypic similarity. 'Male tournament' strategy lets several 'male' individuals compete with each other in order to obtain a mating opportunity with a 'female' individual, the winning male with the optimal mate preference will recombine with the female. The *Intelligent Recombination* operator can be stated as follows:

1) randomly designate an individual as 'female'; and several other individuals as 'male';

2) determine the mutual 'Mate Preference' of every two 'female-male' mate pairs

3) choose the pairs with best 'mate preference' value using the 'male tournament' strategy;

4) mutate them with the 'neutral mutation' method;

5) perform crossover using the highly disruptive universal crossover operator (HUX)

Eshelman & Schaffer (1991) developed an 'incest avoidance' method of preventing premature convergence by allowing two individuals to crossover only if their Hamming distance is above a certain threshold. This threshold drops according to some schedule as evolutionary computation converges. The difficulty of this method is how to choose the appropriate threshold reduction schedule.

In intelligent genetic algorithms, the 'mate preference' function MP is defined as

$$\mathrm{MP} = \frac{\mathrm{Nsc}}{\mathrm{m}} \; \frac{\mathrm{H}}{r(n-y)} \qquad (6)$$

where,

Nsc : the number of similar phenotypic schemata between two individuals

H : Hamming distance between two individuals

r : length of binary representation of all *m* elements, (when m=7, r =3, we need 3 binary bits to represent different 7 elements; see He & Mort, 1997a)

y : number of entries in the search vector included in all similar phenotypic schemata between two individuals

Because the m and r is the same for all individuals, for simplicity, we calculate the 'mate preference' function MP as:

$$\mathrm{MP} = \frac{NscH}{n-4Nsc} \qquad (7)$$

Phenotypic Analysis is utilized to calculate the number of similar phenotypic schemata (Nsc) between two individuals, thereby differentiating the speciation; Hamming distance is calculated for optimal outbreeding in the similar species. The term ( n - 4 * Nsc ) is regarded as a *Bias Rate* which favors the two individuals with higher similar phenotypic schemata (Nsc) and enhances speciation by differentiating separable subpopulations and allowing recombination to occur within the subpopulation.

Optimal Outbreeding is a way of improving the safety ratio of crossover. It allows individuals to avoid breeding with others who are so similar as to produce useless offspring. Unlike the 'incest avoidance' method (Eshelman & Schaffer, 1991), where individuals mate only if their Hamming distance is above a certain threshold, *Genotypic Analysis* is fulfilled in intelligent genetic algorithms by calculating the Hamming distance between two individuals for optimal outbreeding and needs fewer assumptions than the alternative method.

Ohkura and Ueda (1995) presented an extended genetic algorithm with 'neutral mutation' , which is motivated by the fact that most parts of DNA sequences in biological creatures are inactive for their entire lives. Since these inactive parts are invisible to an objective function, a genetic transition occurring to a subset of this gene has no effect on an individual's fitness value. The extended GA succeeds in finding all the global optima of a massively multimodal test function.

Though mutation in simple GAs can bring about new evolutionary innovations for the entire population, it is easy to disrupt useful genetic schemata in the process of evolution; so it leads populations away from local optima in a passive, random and undirected way.

In intelligent genetic algorithms, 'neutral mutation' flips a bit in an individual's genotype string at a very small rate, and the effect on phenotypic schemata and fitness value can be neglected. In the model of 'neutral drift', the individual's fitness value remains unchanged, but actually its genotype has been changed, allowing individuals to explore new adaptive areas of the search space around current local optima without losing any fitness cost.

The above Intelligent Recombination method, combining both phenotype and genotype information together, can produce more viable, fertile and productive offspring than general genetic operators in simple genetic algorithms, and thus increases the convergence speed and robustness in finding global optima.

## 3.7. Pseudocode of Intelligent Genetic Algorithms

We outline *Intelligent Genetic Algorithms* in the following pseudocode: (see table 1).

## 4. Experimental Results

10 replications of applying intelligent genetic algorithms to a practical scheduling problem (Fanti, *et al*, 1996) are described, each run starting with a different random-number seed. In each run the algorithm is allowed to continue until the maximum number of generations is reached. The parameters used in the simulation are as follows:

| initial population size | string bitlength | n | m | mutation rate | crossover rate | decreasing rate $\gamma$ | maximum generation |
|---|---|---|---|---|---|---|---|
| 100 | 63 | 21 | 7 | 0.02 | 0.06 | 0.85 | 1000 |

```
Procedure IGA
Initialization of Entire Population
generation = 0
while termination condition not satisfied do
    Construct phenotypic schemata vector M, relative fitness value fv, absolute fitness
      value with dynamic sharing function fis through phenotypic cluster analysis
    Select suitable superior non-dominated individuals to form chromA, part of new
      'constraint population' in the next generation using 'super-conservative
      selection' strategy
    Aside from chromA, the remainder of 'constraint population', chromB are
      selected from the previous population according to absolute dynamic sharing
      fitness value using 'stochastic universal sampling' method
    Combine chromA and chromB to form new constraint population chrom
    Construct subpopulation from chrom according to phenotypic schemata
        For every subpopulation do
            apply 'intelligent recombination' with neutral mutation' to subpopulation
    Combine all offspring from every subpopulation to form new entire population
```

Table 1. Pseudocode of Intelligent Genetic Algorithms

The six optimal solutions which belongs to optimal solution space $S^*$ is recorded as follows, the corresponding genotype is represented as $g_1^*, g_2^*, g_3^*, g_4^*, g_5^*, g_6^*$.

$$s_1^* = \begin{bmatrix} 497 & 688 & 170 & 12 & 380 & 144 & 176 \\ 310 & 714 & 0 & 231 & 153 & 660 & 216 \\ 0 & 50 & 0 & 114 & 12 & 0 & 6 \\ 0 & 0 & 0 & 454 & 266 & 0 & 282 \\ 0 & 0 & 0 & 0 & 0 & 0 & 128 \end{bmatrix}$$

$$\text{sum} \quad \mathbf{\mathit{807}} \quad 1452 \quad 170 \quad \mathbf{\mathit{811}} \quad \mathbf{\mathit{811}} \quad \mathbf{\mathit{804}} \quad \mathbf{\mathit{808}}$$

$$s_2^* = \begin{bmatrix} 497 & 688 & 170 & 12 & 380 & 144 & 176 \\ 310 & 714 & 0 & 231 & 153 & 660 & 216 \\ 0 & 50 & 0 & 114 & 12 & 6 & 282 \\ 0 & 0 & 0 & 454 & 266 & 0 & 128 \end{bmatrix}$$

$$\text{sum} \quad \mathbf{\mathit{807}} \quad 1452 \quad 170 \quad \mathbf{\mathit{811}} \quad \mathbf{\mathit{811}} \quad \mathbf{\mathit{810}} \quad \mathbf{\mathit{802}}$$

$$\begin{array}{cc} s_3^* = & \left[\begin{array}{ccccccc} 216 & 688 & 170 & 12 & 380 & 144 & 176 \\ 310 & 714 & 0 & 231 & 153 & 6 & 497 \\ 282 & 50 & 0 & 114 & 12 & 660 & 128 \\ 0 & 0 & 0 & 454 & 266 & 0 & 0 \end{array}\right] \\ \text{sum} & \begin{array}{ccccccc} \boldsymbol{808} & 1452 & 170 & \boldsymbol{811} & \boldsymbol{811} & \boldsymbol{810} & \boldsymbol{801} \end{array} \end{array}$$

$$\begin{array}{cc} s_4^* = & \left[\begin{array}{ccccccc} 216 & 688 & 170 & 12 & 380 & 144 & 176 \\ 310 & 714 & 0 & 231 & 153 & 660 & 497 \\ 282 & 50 & 0 & 114 & 12 & 0 & 6 \\ 0 & 0 & 0 & 454 & 266 & 0 & 128 \end{array}\right] \\ \text{sum} & \begin{array}{ccccccc} \boldsymbol{808} & 1452 & 170 & \boldsymbol{811} & \boldsymbol{811} & \boldsymbol{804} & \boldsymbol{807} \end{array} \end{array}$$

$$\begin{array}{cc} s_5^* = & \left[\begin{array}{ccccccc} 216 & 688 & 497 & 12 & 380 & 144 & 176 \\ 0 & 714 & 310 & 231 & 153 & 660 & 282 \\ 0 & 0 & 0 & 114 & 12 & 6 & 128 \\ 0 & 0 & 0 & 454 & 266 & 0 & 170 \\ 0 & 0 & 0 & 0 & 0 & 0 & 50 \end{array}\right] \\ \text{sum} & \begin{array}{ccccccc} 216 & 1402 & \boldsymbol{807} & \boldsymbol{811} & \boldsymbol{811} & \boldsymbol{810} & \boldsymbol{806} \end{array} \end{array}$$

$$\begin{array}{cc} s_6^* = & \left[\begin{array}{ccccccc} 216 & 688 & 497 & 12 & 380 & 144 & 176 \\ 0 & 714 & 310 & 231 & 153 & 660 & 454 \\ 0 & 0 & 0 & 114 & 12 & 6 & 128 \\ 0 & 0 & 0 & 170 & 266 & 0 & 50 \\ 0 & 0 & 0 & 282 & 0 & 0 & 0 \end{array}\right] \\ \text{sum} & \begin{array}{ccccccc} 216 & 1402 & \boldsymbol{807} & \boldsymbol{809} & \boldsymbol{811} & \boldsymbol{810} & \boldsymbol{808} \end{array} \end{array}$$

Also, in each run, the optimal absolute fitness value, number of optima, number of sub-optima, number of phenotype schemata searched, number of subpopulations, constraint population size and entire population size are recorded. The following figures (Figure 1 to 7) from a typical run demonstrate these parameters.

From Figure 1, we can see that entire population achieves one global optima ( absolute fitness value equals 5 ) when number of generations equals 95.

From Figure 2, we know that moving from generation 95 to the maximum generation 1000, the number of optima obtained is from one to six global optima. Intelligent Genetic Algorithms have succeeded in finding multiple optima in this combinatorial multimodal optimization problem.

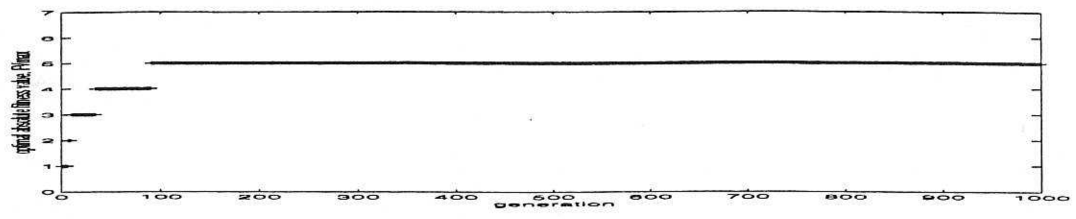

Figure 1. Optimal Absolute Fitness Value ( *Fv*max ) of a typical run

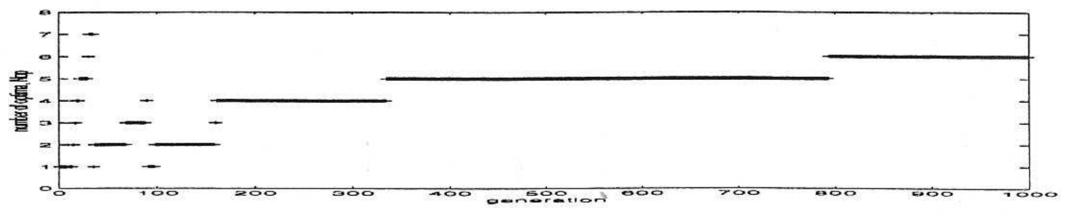

Figure 2. Number of Optima ( Nop ) of a typical run

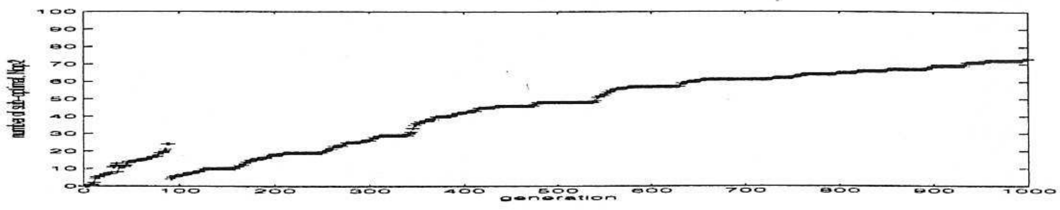

Figure 3. Number of Sub-Optima (Nop2) of a typical run

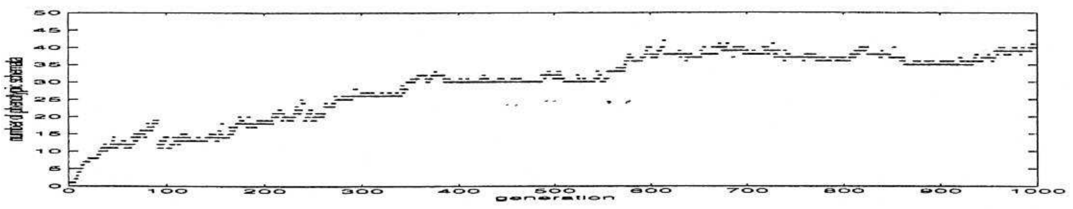

Figure 4. Number of Phenotypic Schemata (Nps) of a typical run

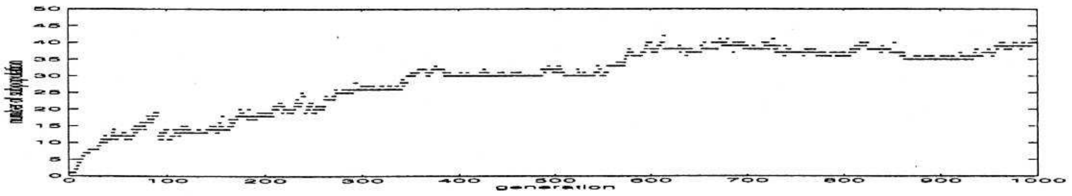

Figure 5. Number of  Subpopulation (Nsp) of a typical run

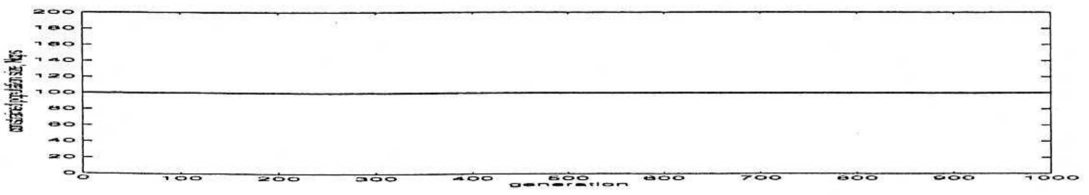

Figure 6. Constraint Population Size of a typical run

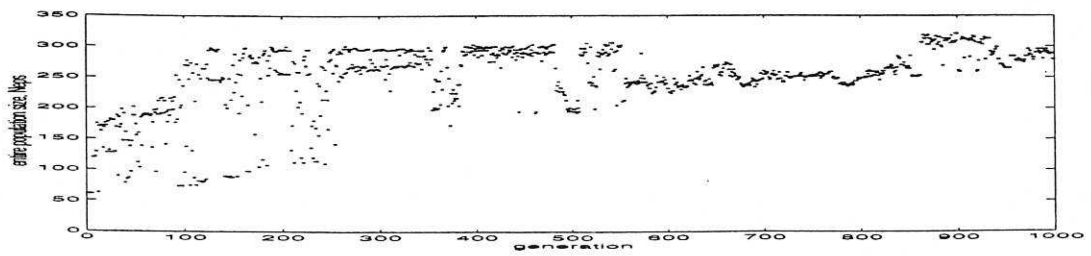

Figure 7. Entire Population Size (Neps) of  a typical run

Figure 3 shows that the number of distinct suboptima is increasing with generation, from generation 95 onwards, the absolute fitness value of suboptima equals to 4, these suboptima are non-dominated individuals and are local optima also.

Figure 4 shows that the number of phenotype schemata normally increases with generation. Sometimes it decreases because some schemata could not be combined with other schemata to form a individual with higher fitness value, and then it dies out in the evolution procedure. Since the number of subpopulations is exactly equal to the number of phenotype schemata searched so far, figure 5 is the same as figure 4.

Figure 6 shows that the constraint population size is a constant, and equals the initial population size (Nchrom =100). This indicates that the sum of the number of both first class and second class non-dominated individuals is no more than 100. For example, when the generation equals 100, the number of first class non-dominated individuals (just global optima) is 7; and the number of second class non-dominated individuals (just local optima) is 73. Thus their sum is equal to 80, which is smaller than 100.

Figure 7 shows that the size of the entire population which is constructed by the offspring of all subpopulations in the last generation is dynamic. This indicates that when a non-dominated individual with a higher fitness value is obtained, many non-dominated individuals with a lower fitness value may die out.

## 5. Simple Fitness Landscape Analysis

Following the definition of fitness landscape in Part I, a simple analysis method is provided to indicate the actual positions of six optima in the fitness landscape.

The Hamming distance between the genotype of any two optimal solutions, $\mathbf{H}_o$, the mean Hamming distance between one optimal solution's genotype and the other five, $\mathbf{H}_{mo}$ , and the mean fitness value of *1-mutant* neighbors of an optimal solution's genotype, $F_{1m}$, are calculated, and the results are summarized in the following table:

| $H_o$ | $g_1^*$ | $g_2^*$ | $g_3^*$ | $g_4^*$ | $g_5^*$ | $g_6^*$ |
|---|---|---|---|---|---|---|
| $g_1^*$ | 0 | 2 | 5 | 6 | 7 | 10 |
| $g_2^*$ | 2 | 0 | 3 | 4 | 5 | 8 |
| $g_3^*$ | 5 | 3 | 0 | 1 | 6 | 9 |
| $g_4^*$ | 6 | 4 | 1 | 0 | 7 | 10 |
| $g_5^*$ | 7 | 5 | 6 | 7 | 0 | 7 |
| $g_6^*$ | 10 | 8 | 9 | 10 | 7 | 0 |
| $H_{mo}$ | 5.0 | 4.4 | 4.8 | 5.6 | 6.4 | 8.8 |
| $F_{1m}$ | 3.3968 | 3.4286 | 3.4286 | 3.4286 | 3.3968 | 3.3651 |

Table 2 Hamming distance parameters

For example, the Hamming distances between the genotype of optimum $g_1^*$ and the genotype of optima $g_2^*$, $g_3^*$, $g_4^*$, $g_5^*$, $g_6^*$ , are 2, 5, 6, 7, 10, and their mean Hamming distance is 5.0. The six optimal solutions are clustered relatively far apart in the fitness landscape. Also, the mean fitness value of all $\binom{l}{1} = 63$ 1-mutant neighbors of $g_1^*$ is 3.3968. Comparing with the optimal fitness value 5, it is clear that even the 1-mutant neighbors cannot provide suitable information about where the optimum, $s_1^*$, is in the fitness landscape as the fitness landscape is quite rugged. Furthermore, to demonstrate the previous discussion, a measure of search difficulty, *fitness distance correlation* (FDC), [Jones, & Forrest, 1995] is introduced and examined in relation to genetic algorithm performance in the fitness landscape.

Jones and Forrest (1995) suggested that the relationship between fitness and distance to the goal is very important in GA searching. They viewed GA fitness function values as estimates of the distance to the nearest global optimum of the search. FDC is one method of quantifying this relationship. To measure the correlation of fitness function value with distance to a global optimum, a simple way is to examine the problem with known optima, sample a number of individuals, and compute the correlation coefficient, r , given the set of (fitness, distance) pairs.

That is, given a set of $F = \{f_1, f_2, \ldots, f_n\}$ of $n$ individuals' fitness values and a corresponding set of $D = \{d_1, d_2, \ldots, d_n\}$ of the $n$ Hamming distance to the nearest global optimum, the fitness distance coefficient $r$ is calculated as:

$$r = c_{FD} / (s_F\, s_D)$$

where
$$c_{FD} = \frac{1}{n} \sum_{i=1}^{n} (f_i - f_m)(d_i - d_m)$$

is the covariance of $F$ and $D$, and $s_F$ , $s_D$ , $f_m$ and $d_m$ are the standard deviations and means of $F$ and $D$ respectively.

In this paper, the problem is one of finding maxima, so we might expect that fitness value increases as distance to a global optima decreases which implies the coefficient $r$ is always negative. With an ideal fitness function, $r$ will be -1.0, meaning that the fitness landscape around this global maximum is very correlated and smooth, whereas if $r$ is close to zero, the fitness landscape around this global maximum is very uncorrelated and rugged.
In terms of Hamming distance measure, the nearest individuals to a global optima are the 1-mutant and 2-mutant neighbors. Since the length of global optimum is $l = 63$, there are $\binom{l}{1}=63$ 1-mutant, and $\binom{l}{2}=1953$ 2-mutant neighbors. For any of the six optima, there are in total 2016 nearest neighbors and it is straightforward to calculate their corresponding fitness values. The fitness distance coefficients of the six optima in the multimodal combinatorial optimization problem are summarized as:

| | $g_1^*$ | $g_2^*$ | $g_3^*$ | $g_4^*$ | $g_5^*$ | $g_6^*$ |
|---|---|---|---|---|---|---|
| r | -0.1890 | -0.1882 | -0.1921 | -0.1994 | -0.1923 | -0.2103 |

All the global maxima are negative and close to the zero, therefore, the fitness landscape is very uncorrelated and rugged. That is to say, the global maximum is very difficult for a simple GA to search. From another viewpoint, it demonstrates the search capability of intelligent genetic algorithms which can find all six optima in such an experiment.

## 6. Discussion, Conclusion and Future Prospects

### 6.1. Discussion:

This discussion considers a number of properties of Intelligent Genetic Algorithms:

6.1.1. Explicit Parallelism in intelligent genetic algorithms works through their spatial population structures. The advantages are threefold: firstly, the entire population is split into subpopulations naturally, and an algorithm is specified which uses only local rules and local information, permitting recombination only within the subpopulation, and making it massively parallel. Secondly, subpopulations are based on the phenotypic schemata, super-conservative selection strategy always maintains good phenotypic schemata. Genotype schemata are mapped from phenotypic schemata, thus reinforcing the power of intelligent genetic algorithms to process many building blocks or genotype schemata in the search space simultaneously. This power is stronger than that in implicit parallelism [Goldberg, 1989] in simple genetic algorithms. Thirdly, our simulation results show that both convergence speed and quality are greatly improved in comparison with a simple genetic algorithm [He & Mort, 1997a].

### 6.1.2. Global Optima

In intelligent genetic algorithms, the hybrid drift model through the complementary interaction between 'genetic drift' and 'neutral drift' may explain the mystery that population can be internalized to escape away from local optima to global optima.

Intelligent genetic algorithms utilize a 'hybrid drift' model to design a multi-dimensional fitness landscape through a relative fitness vector. When the relative fitness vector of one individual is non-dominated by that of any other one in the entire population , 'neutral drift' is in effect. If this individual is ranked as first class or second class, it is guaranteed to have at least one offspring in the next generation through super-conservative selection. In other words, any superior non-dominated individuals are regarded as having equal-(relative)-fitness values, thus allowing the exploration of the different adaptive peaks nearby. On the other hand, when one individual is dominated by another, genetic drift is in effect which implies this individual will die out and be replaced by new non-dominated one. This model has proved effective in leading populations to escape from local optima to global optima in this typical combinatorial multimodal optimization problem

### 6.1.3. Efficiency and Robustness

Intelligent genetic algorithms have a centralized distributed structure: specifically they have centralized constraint populations and distributed subpopulations. IGAs use constraint populations in order to limit entire population size and 'super-conservative selection' strategy to choose superior non-dominated individuals, helping to reduce computation complexity efficiently.

From Figure 3, we see that the number of distinct suboptima is increasing with generation in approximately linear fashion. This fact demonstrates the linear speedup of intelligent genetic algorithms, and is derived from the explicit parallelism and distributed algorithm in an IGA. It is seen that this increases the efficiency and robustness of IGA greatly.

Intelligent Genetic Algorithms solve the given combinatorial multimodal optimization problem without too much a priori knowledge of the fitness landscape,(such as number of optima and their distribution) and require few parameter settings, which is very convenient to users. These algorithms are independent of encoding of the phenotype, and are suitable for solving most GA-hard and GA deceptive problems which are regarded as challenging problems for the application of GAs.

### 6.2. Conclusion:

Combinatorial Multimodal Optimization Problem is highly-discrete, non-linear, and discontinuous, and can be regarded as one of the most challenging problems in GA applications. As far as we know, very few publications have addressed the problem or provided satisfactory algorithms and solutions. There are marked differences from continuous multimodal optimization functions [Goldberg & Richardson, 1987; Deb & Goldberg, 1989; Ohkura and Ueda, 1995] in which the number of fitness peaks and their distribution information is known and which can be used to divide populations into different subpopulations exploring relevant fitness peaks. Exact information about the fitness landscape cannot be known beforehand, so this kind of problem is a serious challenge to GA applications.

Experimental results and simple fitness landscape analysis demonstrate that Intelligent Genetic Algorithms have the capability of generating sufficient genetic innovations, preserving promising genetic diversity, guiding population escape from local optima and exploring multiple global optima effectively and efficiently in this difficult combinatorial multimodal optimization problem.

### 6.3. Future Prospects

Research is in progress to address the following issues:

1) providing formal analysis of intelligent genetic algorithms.

2) analyzing the statistical properties of intelligent genetic algorithms in the fitness landscape

3) developing intelligent genetic algorithms applied to the multiobjective optimization problems and the multimodal optimization problems which traditional genetic algorithms find difficult,

3) utilizing intelligent genetic algorithms to attack GA-hard and GA-deceptive problems [Davidor,1991; Liepins & Michael, 1991].

**Acknowlegdements**

The authors wish to thank Dr. Jane Shaw of the Department of Automatic Control and Systems Engineering for her valuable suggestions in the preparation of this paper.

### References

Andersson, M. (1994): *Sexual Selection*. Princeton University Press.

Baker, J.E. (1987): Reducing Bias and Inefficiency in the Selection Algorithm. *Proceeding of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates, Publishers.

Collins, R. J., & Jefferson, D. R. (1991): Selection in massively parallel genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann.

Darwin, C. (1862): *On the various contrivances by which orchids are fertilized by insects.* John Murray.

Darwin, C. (1871):*The descent of man, and selection in relation to sex (2 vols.)* John Murray.

David, Y. (1991): Epistasis Variance: A Viewpoint on GA-Hardness. In *Foundations of Genetic Algorithms.* Gregory J.E.Rawlins (Ed.), Morgan Kaufmann Publishers,

Davidor, Y., Yamada , T.,and Nakano, R. (1993): The Ecological framework In Improving GA performance at virtually zero cost. *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann.

De Jong, K.A .(1975): An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. (Doctoral Dissertation , University of Michigan). *Dissertation Abstracts International*, 36 (10), 5140B. (University Microfilms No. 76-9381).

Deb,K. and Goldberg, D. E. (1989): An Investigation of Niche and Species Formation in Genetic Function Optimization. *Proceedings of the Third International Conference on Genetic Algorithms.*

Eshelman. L.J (1991): The CHC adaptive search algorithms : How to have safe search when engaging in non-traditional genetic recombination. *Foundations of Genetic Algorithms.* Gregory J.E.Rawlins (Ed.), Morgan Kaufmann Publishers, pp.265-283.

Eshelman, L.J., & Schaffer, J. D. (1993): Crossover's niche. In Forrest (Ed.),pp.9-14.

Fanti, M. P., Maione, M., Piscitelli, G. and Turchiano, B (1996): Heuristic Scheduling of jobs on a multi-product batch processing machine. *International Journal of Production Research*, Vol. 34. No. 8, pp2163-2186.

Fonseca C. M., and Fleming P. J. (1995a): An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation*, vol. 3, no 1, pp 1-16.

Fonseca, C.M., and Fleming, P. J. (1995b): Multiobjective genetic algorithms made easy: *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems:* pp.44-52, The Institution of Electrical Engineers.

Forrest, S. (Ed.). (1993): *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann.

Goldberg, D. E. and Richardson, J. (1987): Genetic Algorithms with Sharing for Multimodal Function Optimization. *Proceedings of the Second International Conference on Genetic Algorithms.* John J. Grefenstette ( Ed ).

Goldberg, D. E. (1989): *Genetic Algorithms in Search , Optimization, and Machine Learning.* Addison-Wesley Publishing Company.

Goldberg, D.E., and Richardson, J. (1987): Genetic Algorithms with Sharing for Multimodal Function Optimization, *Proceedings of the Second International Conference on Genetic Algorithms.* Grefenstette, J. J. (Ed.), P.41-49..

He Liwen and Mort, Neil. (1997a): Genetic Algorithm for Scheduling Optimization on a Multi-Product Batch Processing Machine. 4th IFAC Workshop on *Intelligent Manufacturing Systems*, Seoul, Korea, pp165-170.

He Liwen and Mort, Neil. (1997b): Approximate Optimization Algorithms for Scheduling on a Multi-Product Batch Processing Machine, submitted to *Journal of Scheduling*.

Holland, J. H. (1975): *Adaptation in natural and artificial systems*. Ann Rbor: The University of Michigan Press.

Jelasity, M. and Dombi, J. (1995): GAS , a Concept on Modeling Species in Genetic Algorithms. *Artificial Evolution*. European Conference, AE95, Brest, France.

Jones, T. and Forrest, S, (1995): Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. *Proceedings of 6th International Conference on Genetic Algorithms*. University of Pittsburgh. Larry, J. Eshelman (Ed.)

Kimura, M. (1983): The neutral theory of molecular evolution. In M. Nei & R.K. Koehn (Ed.), *Evolution of genes and proteins*, pp.213-233. Sunderland, MA: Sinauer.

Liepins, G. E. & Vose, M. D. (1991): Deceptiveness and Genetic Algorithm Dynamics, *Foundations of Genetic Algorithms*. Gregory J.E.Rawlins (Ed.), Morgan Kaufmann Publishers, pp.36-52.

Luling, R., Monien, B., Reinefeld, A. and Tschoke , S. (1996): Mapping Tree-Structured Combinatorial Optimization Problems onto Parallel Computers. In *Solving Combinatorial Optimization Problems in Parallel* . Lecture Notes in Computer Science. Vol. 1054, Afonso Ferreira and Panos Pardalos(Eds), 1996.

Miller, G. F, (1994): Exploiting Mate Choice in Evolutionary Computation : Sexual Selection as a Process of Search , Optimization , and Diversification. *Evolutionary Computing* . C. Fogarty (Ed.), AISB Workshop, Leed , UK.

Muhlenbein, H. (1991): Evolution in Time and Space - The Parallel Genetic Algorithm, *Foundations of Genetic Algorithms*. Gregory J.E.Rawlins (Ed.), Morgan Kaufmann Publishers, pp265-283.

Ohkura, K and Ueda, K. (1995): A Genetic Algorithms with Neutral Mutations for Massively Multimodal Function Optimization. *IEEE International Conference on Evolutionary Computation*, Australia, pp.361-366.

Papadimitriou, C. and Steiglitz, K. (1982): *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall Inc., Englewood Cliffs, NJ.

Schneider. G., Schuchhardt. J., Wrede. P.(1996): Evolutionary Optimization in Multimodal Search Space. *Biological Cybernetics*. Vol.74, No. 3, pp.203-207.

Shaw. K.J. and Fleming P.J., (1996): 'An initial study of practical multi-objective production scheduling using genetic algorithms. *Proc. UKACC Int Conf on Control '96, Exeter*, pp.448-453.

Sirinivas, M. and Patnaik. L. M. (1994): Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *IEEE Transaction on Systems , Man and Cybernetics* . Vol. 24, No.4.

Todd P. M. and Miller, G. F. (1991): On the Sympatric Origin of Species: Mercurial Mating in the Quicksilver Model. *Proceedings of the Fourth International Conference on Genetic Algorithms*. University of California, San Diego.