1. Genetic Algorithms with Limited Convergence

Jiří Kubalík[1,2], Léon Rothkrantz[1], and Jiří Lažanský[2]

[1] Group of Knowledge Based Systems, Department of Mediamatics, TU Delft, P.O. Box 356, 2600 AJ Delft, The Netherland
[2] Department of Cybernetics, CTU Prague, Technicka 2, 166 27 Prague 6, Czech Republic
email: kubalik@labe.felk.cvut.cz

1.1 Introduction

Genetic algorithms (GAs) are probabilistic search and optimisation techniques, which operate on a population of chromosomes, representing potential solutions of the given problem [1.10]. In a standard GA, binary strings of ones and zeros represent the chromosomes. Each chromosome is assigned a fitness value that expresses its quality considering the given objective function. Such a population is evolved by means of reproduction and recombination operators in order to breed the optimal solution's chromosome. The evolution is running until some termination-condition is fulfilled. The best chromosome encountered so far is then considered as the found solution.

The basic analysis of GA's behaviour is based on a notion of a *schema* [1.15] as a template, which matches certain class of chromosomes. As the population evolves some good schemata are represented by increasing number of chromosomes whilst bad schemata disappear. The fixed positions of those good schemata constitute so-called *building blocks* (BBs), which represent important components of the final solution. The optimum solution emerges when these building blocks are mixed together in an optimal way in some chromosome.

GAs simultaneously carry out exploitation of the promising regions found so far and exploration of other areas for potentially better solution. The weak point of a GA is that it often suffers from so-called *premature convergence*, which is caused by an early homogenisation of genetic material in the population. This means that no more exploration can be performed. There are many factors affecting the convergence of a GA - the used population size, type and rate of application of crossover and mutation operators, encoding used and many others. Inadequate population size can not provide the GA with sufficient amount of genetic material to evolve the optimal chromosome. Improperly designed and set genetic operators can not maintain an optimal balance between exploitation and exploration of the GA.

This paper introduces a novel approach to protect GAs from getting stuck in local optima and so extending the search power of GAs. To achieve this

we have proposed a GA where only limited convergence of the population genotype can take place.

The paper starts with an overview of some recent approaches to the problems related somehow to prevent premature convergence, preserving the population diversity (or better the multimodal diversity), and improving the performance of GAs in general. The next section introduces the proposed genetic algorithm with limited convergence and discusses its aspects. Section 1.4 briefly describes the test problems used for experimental evaluation of the proposed algorithm. Empirical results presented in section 1.5 show that the algorithm performs very well across the representative set of search problems. Especially its explorative power and the ability to keep useful diversity of the population is demonstrated there. Section 1.6 summarises and concludes the paper and mentions interesting topics to be studied in future in this area.

1.2 Related Work

There are many factors that affect the convergence of a GA. The most influencing and the most frequently studied are the used reproduction strategy, the size of the evolved population, the representation, i.e. the mapping of the problem parameters on a binary string in conventional GAs and last but not least the used recombination operators and the frequency with which they are applied and other problem dependent parameters. Let us briefly list some of the approaches that have been recently proposed to tackle the aspects mentioned above.

Dual GAs [1.3], [1.4] represent one possible way to introduce some kind of redundancy into the evolved genetic material. The Dual GAs use a standard binary representation extended with just one bit, called the head bit or meta bit, which is added to every individual in the population. This bit does not code any information of the represented solution. Instead this extra bit is used to determine the way the informative part of the chromosome will be interpreted. If the value of the head bit is 0 then the rest of the chromosome is taken as it is. Otherwise if the head bit is 1 the string is interpreted as its binary complement. This means that two binary complementary strings represent the same solution. In other words the population may contain individuals with completely different genotype but with the same phenotype and so the same fitness values.

It has been obvious from the very beginning of the GA's use that the population size plays a crucial role in determining the convergence quality of GAs. The population should be large enough to provide an adequate initial supply of BBs. Another aspect of population sizing involves the decision making between competing BBs. De Jong [1.5] recognised that the decision making for a particular BB is strongly affected by contributions of other BBs. There are several studies that try to estimate an adequate population size [1.8], [1.12], [1.14]. However, a user should know quite much information

about the problem at hand like the size of the considered BBs, the number of BBs in the problem, the average fitness variance of the considered BB, and some other conservative assumptions in order to be able to use the theoretical estimates.

Another factor that affects the convergence of GA is the selection scheme. Generally the selection should work so that the chromosomes representing better solutions are given bigger chance to take part in the process of generating a new and hopefully better population. The extent to which the better chromosomes are favoured is measured by so-called selection pressure. However, if the selection pressure is too low then the convergence to the optimal solution is too slow. In the opposite case, i.e. if the selection pressure is too high, the population is prone to converge very fast towards some sub-optimal solution. So a commensurate affords have been spend to analyse various selection schemes and to develop the convergence models of GAs under those selection schemes [1.1], [1.11], [1.13], [1.23]. In order to keep the desired distribution of fitness values in the population during the whole run a proper scaling technique [1.22] might be engaged in GA as well.

GAs are very often engaged in solving multimodal optimisation problems. This implies that the ability of the GA to keep just the "raw" diversity is not good enough. Instead the multimodal diversity covering many niches of the search space is required. One early approach to maintain many niches in the GA was based on utilisation of so-called shared fitness values that were calculated using a sharing function [1.9]. The general concept of fitness sharing is based on the idea that each individual's fitness is divided by the number of neighbours within the niche to which the given individual belongs. Thus the goal desired state is such that the population is distributed over a number of different peaks in the search space, with each peak receiving a fraction of the population in proportion to the height of that peak. It results in a situation that the convergence occurs within a niche, but convergence of the full population is avoided.

Other works dealing with maintenance of the population diversity date back to the seventies. Cavicchio [1.2] and De Jong [1.5] came up with techniques called preselection and crowding, respectively. Both are inspired by an ecological phenomenon that similar individuals in the natural population compete against each other for limited resources. Dissimilar individuals tend to occupy different niches, so they typically do not compete. As a result, the number of members of a particular niche does not change during the evolution of the whole population of fixed size. In preselection and crowding this is achieved so that the newly created individuals, which are to be inserted into the population replace the most similar individuals in the current population. The two methods differ in that how the individuals to be replaced are found. In crowding the replacement is found among randomly chosen CF individuals. Preselection assumes that a parent is one of the closest individuals to the generated offspring so the parent is replaced if the competing child is bet-

ter. Mahfoud studied several strategies of crowding and preselection in [1.20] and proposed a variation of preselection called deterministic crowding that processes two parents and two offspring at a time, and uses phenotypic similarity measure to determine which offspring competes against which parent. He shows that his method is better in clustering solutions about all peaks of the tested problems.

Mengshoel and Goldberg [1.21] proposed a niching algorithm called probabilistic crowding which is a successor of the deterministic crowding. The two core ideas in probabilistic crowding are (*i*) to hold tournaments between similar individuals and (*ii*) to let tournaments be probabilistic. Theoretical analysis and empirical results showed that probabilistic crowding is a simple, stable, predictable and fast niching algorithm.

A search performance of the GA can be also measured in terms of the size of the search space explored during the run. To efficiently sample the whole search space is the task for recombination operators i.e. crossover and mutation operators. In standard genetic algorithms the crossover is considered to be of the primary role in an exploration process whilst the mutation is used to preserve the diversity in the population and to preserve a loss of information [1.10], [1.22]. However, the utility of crossover and mutation changes with a population size. A mutation can be more useful than crossover when the population size is small whilst a crossover is more useful when the population is large.

A proper use of recombination operators is even harder since there are many different variants of crossover. The implementations differ between each other in many aspects. The disruption effect of crossover, first analysed for 1-point crossover in [1.15], is the probability that the crossover will disrupt long schemata. This is important to predict how the promising schemata will be propagated to subsequent populations. De Jong and Spears [1.6] characterised recombination in terms of productivity and exploration power. Those characteristics describe the ability of crossover to generate new sample points in the search space. De Jong and Spears also derived a heuristics that the more disruptive crossovers (which are also more productive and explorative) are better when the population is small and less disruptive operators are better when the population is large relative to the problem size. For more recent work on the role of recombination operators see [1.26].

Another approach is to dynamically adapt the rates of the utilisation of multiple operators during the run. Spears [1.25] used an extra tag-bit attached to every individual to store the information about which crossover (1 is 2-point and 0 is uniform crossover) should be preferred when crossing two parental chromosomes. So if the parents both have 1 at the tag-bit position then the 2-point crossover is used. Similarly if both have 0 then uniform is used. Otherwise one of the two operators is chosen with a probability 0.5. Spears found that this adaptive approach always had a performance

intermediate between the best and worst of the two single recombination operators.

Srinivas and Patnaik [1.27] used adaptive probabilities of crossover and mutation to achieve the twin goals of maintaining diversity in the population and sustaining convergence capacity of the GA. They increased the probability of crossover and mutation when the population was stuck at local optima and decreased the probabilities when the population was scattered in the solution space. They also considered the need to preserve "good" solutions. This was attempted by having low probabilities for highly fit individuals while poor solutions will encounter high probability of crossover and mutation rate.

Kubalík and Lažanský proposed so-called partially randomised crossover operators (PRX) as an enhancement of the traditional crossover operators used for binary representation [1.16], [1.17], [1.18]. The enhancement of the crossover functionality is in a modified treatment of a common schema of the parents, which are the bits common to both parental chromosomes. Standard operators work so that both offspring inherit unchanged all the bits of the common schema. As the population converges, the common schema occupies a growing portion of the parent chromosomes and so the crossover can produce only little new. The PRX operators do not strictly preserve the common schema of the parents, since in one of the two generated offspring, a portion of the common schema is changed with the probability that evolves during the run. Thus, the population is not saturated with superior building blocks but also with their randomly chosen binary complements. The diversity of the population is permanently maintained, which helps to preserve the SGA from getting stuck in a local optimum and enhance the exploration of the search space beyond the limits imposed by the pure evolution.

The PRX operators were engaged in genetic algorithms with permanent re-initialisation of parental common schemata proposed in [1.19]. The algorithm forks the search into two directions when creating a new population from the old one. To do so the main generational cycle consists of two steps. First, the primary population is evolved for M generations with the use of the PRX crossover in order to produce both the direct and randomised offspring. The randomised offspring are stored in the secondary population. Since it is derived from the primary population, it still has much in common with the "main stream" evolution so it does not represent quite random samples of the search space. As such the secondary population represents a source of hopefully useful diverse genetic material. Then the secondary population is refined through a short evolution running for N generations. Finally these two populations are merged together into one new primary population and the new main iteration can be launched. The empirical results show that in such a way a diversity of the population can easily be maintained while converging faster and to better solutions than with the standard GA.

1.3 Genetic Algorithms with Limited Convergence

This section describes a novel approach to improve the performance of GAs. The main goal of the proposed algorithm called the *genetic algorithm with limited convergence* (GALCO) is to maintain the fruitful diversity of the evolved population during the whole run and so to preserve a GA from getting stuck in local optima. To achieve this a concept of imposing limits on the convergence of the population is adopted.

The algorithm inherits most of the features from the standard GA working on a binary representation. In fact it is a type of the incremental GA where only one crossover operation is performed in each generation cycle (the step from the old to the new population). As such it uses a standard selection strategy to choose the parents, a classical crossover operator (2-point crossover is used here, as commented further) for generating new chromosomes, and special rule for inserting of the offspring chromosomes into the population. What makes the GALCO unique is just the way the convergence of the population is maintained within specified boundaries.

There is a limit imposed on the maximum convergence of every position of the representation. Let us denote the vector of genes at the i-th position of the chromosome over the whole population as the i-th column of the population. Then the limit is expressed as a symmetric integer interval $[PopSize/2 - C, PopSize/2 + C]$, where $PopSize$ is the population size. The parameter C denotes the *convergence rate*. Its value is the input parameter to the algorithm and can be chosen from the range 0 to $PopSize/2$. Strictly speaking $2 \times C$ defines the maximal allowed difference of the frequency of ones and zeros in every column of the population. So the ratio of ones and zeros must be $1:1$ during the whole run in the case of $C = 0$ or the ratio can change up to $0 : PopSize$ in favour of either ones or zeros for $C = PopSize/2$. This is a principal condition of the algorithm. To keep the condition valid during the whole run a special insertion rule for incorporating offspring into the population has been used.

The functional scheme of the GALCO is shown in Fig.1.3. First an initial population of chromosomes is generated. It is made sure that the distribution of ones and zeros does not violates the convergence constraint at any position of the chromosome. In our case the evolution starts from maximally diverse population i.e. every column of the population consist of an equal number $PopSize/2$ of ones and zeros regardless of the chosen convergence range. The body of the algorithm is through the steps 2-5, which realizes the generational cycle of the incremental GA. In the step 2 a pair of parental chromosomes is chosen according to the used selection strategy (a tournament selection is used here). Then the parents are crossed over using the 2-point crossover to yield two new chromosomes. Note that there is no parameter specifying the rate of application of the crossover needed since the parents always undergo this operation. The 2-point crossover was chosen intentionally since it is the least disruptive recombination operator among the standard operators. So

```
Step 1  Generate initial population of size PopSize
Step 2  Choose parents
Step 3  Create offspring using the 2-point crossover
Step 4  Insert the offspring into the population according
        to the following rule
        if (max(child1,child2) > max(parent1,parent2))
        then replace both parents with the children
        else{
          find(current_worst)
          replace_with_mask(child1, current_worst)
          find(current_worst)
          replace_with_mask(child2, current_worst)
        }
Step 5  if (not finished) then go to Step 2
```

Fig. 1.1. A functional schema of the GALCO algorithm

it is best suited for preserving the promising building blocks when mixing genes of two parental chromosomes. It is supposed to be a good counterpart to the artificially maintained rather high diversity in the population (as it will be shown in section 1.5 the best performance of the algorithm is achieved with the convergence rate $C << PopSize/2$). There is no explicit mutation operator used in the algorithm.

The most important action of the algorithm comes in step 4. Here the offspring is inserted to the population according to the insertion rule, which follows two main objectives: (*i*) to use as much of the genetic material of the newly generated individuals as possible and (*ii*) not to violate the maximal allowed convergence rate. In practice this is implemented so that both children replace their parents iff at least one of the children has better fitness than both parents. Otherwise the children replace the worst individuals of the current population using the *replace_with_mask* operator described below.

The effect of replacing the parents with both offspring in the "then part of the rule" is such that the distributions of ones and zeros do not change in any column of the population. This is obvious since the genetic material of parents and their children is invariant through the application of the crossover operation. Thus it is ensured that if the old population does comply with the desired convergence range the new population must comply with the convergence range as well.

Slightly more complicated situation arises when the children are both of rather poor quality. Replacing the parents with their offspring irrespectively of the offspring quality would cause problems with a slow convergence to the optimal solution. Note that the case the parents produce fitter offspring is much less frequent than the opposite case i.e. that both offspring are worse than the better parent. So the breeding phase (crossover plus replacement of parents) would be in most cases counterproductive. Moreover the elitism i.e. preserving of the best individual in the population could not be ensured.

```
for(i=0;i<chrom_length;i++){
    change = child.genes[i]-current_worst.genes[i]
    if(PopSize/2 - C < conv[i] + change < PopSize/2 + C) then{
      convergence[i] = convergence[i] + change
      current_worst.genes[i] = child.genes[i]
    }
}
```

Fig. 1.2. A functional schema of the *replace_with_mask* operator

Apparently if the best individual in the population was selected as a parent and did not succeed to generate at least the same fit offspring the best so far solution would disappear from the population when replaced by the offspring. On the other hand the evolution should not be restricted only to those rather singular moments when the parents breed something better than they are. Some reasonable way to use the newly generated chromosomes whatever good they are might be very useful since it would considerably increase the rate of sampling of the search space.

The trade-off between exploration power and exploitation ability of the algorithm is accomplished by placing the generated offspring into the population using the *replace_with_mask* operator. Generally any chromosome of the population can be replaced. Here, the chromosomes representing the least fit solutions are chosen in order to reduce the loss of quality due to the operation performed on the replaced individual. Note that it is necessary to start seeking for the worst individual from a randomly chosen position in the population in order to ensure that any out of the multiple worst individuals can be chosen with an equal probability. If one did not take care of it the first or the last (depending on the implementation of the search routine) worst individual in the population would be chosen all the time that would restrict the sampling effect of the operator.

The operator works so that it traverses the chromosome of the worst individual and replaces its i-th gene with a corresponding gene of the child's chromosome if and only if such a change is legal. So the bit is replaced if this does not make the frequency of ones and zeros of the corresponding column of the population to exceed the allowed convergence range. Otherwise the original gene of the worst chromosome retains in the population. It is apparent that the role of the *replace_with_mask* operator is two-fold, it enables the pair of offspring, which do not improve the parents' best as well as the worst individual in the population to contribute to the population genotype. Note that usually neither the inserted nor the replaced chromosome remains unchanged when the operator is applied. Instead the genetic material of the two chromosomes is mixed in the resultant chromosome. So the operation should be seen as a merging of two chromosomes rather than an inserting of a new one in the population. One should expect that at the beginning of the run the genes of the new offspring would dominate in the so-composed

chromosome while in the later stages of the run more and more genes of the old chromosome would be retained. Particular characteristics depend on many factors such as the used value of the parameter C, the character of the solved problem etc. Apparently the greater the C is the more of the offspring's genes are used in the early phase of the run and vice versa. An example of how the bias can change during the run will be shown in section with experiments. Also the impact of the use of the operator on the performance of the whole algorithm will be empirically tested there.

From this point of view such an operation of merging of two chromosomes can be considered as a kind of biased uniform crossover as well, where the bias is either in favour of the old or of the new chromosome. In this sense the whole algorithm belongs to the class of approaches, which try to profit from simultaneous utilisation of the most and the least explorative crossover operators uniform and 2-point [1.25]. Here the 2-point crossover is used as let say primary or explicit recombination operator whilst the form of uniform crossover appears as a side effect of preventing the population from becoming too homogenous. Last but not least an implicit elitism embodied in the algorithm should be mentioned. Any worse individual can replace the best individual of the current population neither by replacing the parents with their offspring (since it happens only if the offspring is better) nor by application of the *replace_with_mask* operator. So the best-so-far fitness value can only improve during the evolution.

To implement those decision makings performed within the operator *replace_with_mask* an integer vector convergence[] is used to store the gene distribution statistics of all columns of the population. The i-th vector element is updated whenever a corresponding gene changes its value. Note that all vector elements must be from $[PopSize/2 - C,\ PopSize/2 + C]$ during the whole run. The vector of convergence statistics can be considered as a mask (this is where the name of the operator is derived from), specifying for each position of the replaced chromosome whether it can be changed or not.

1.4 Test Problems

This section briefly describes the test problems that were used in the experiments described in the next section. The presented selection of the problems was made with intention to cover non-linear function optimisations, deceptive, royal road, hierarchically decomposable, and multiple-optima problems.

First test problem is based on function F101(x,y) taken from [1.31]. The function is defined as follows:

$$F101(x,y) = -x \cdot \sin(\sqrt{|x-y-47|}) - (y+47) \cdot \sin(\sqrt{|y+47+x/2|}) \ ,$$

where the parameters x and y are coded on 10 bits and converted to integer values from the interval (-512, 511).

The function is used as basic building block of the problem. It is non-linear non-separable and highly multimodal function of two variables. This means that the optimal value of one variable can not be determined independently of the other parameter. Here the whole problem is constructed in such a way that it consists of 7 triples $x_1 - y - x_2$, where each one contributes to the overall fitness with the value F101(x_1, y) + F101(x_2, y). So the parameter y is nonlinearly bound with two other parameters x_1 and x_2, which makes the problem even more difficult. The total length of the chromosome is 210 bits. The fitness of the whole chromosome is calculated as the average value of those 7 function contributions. The global minimum value of the problem composed of F101(x, y) is -955.96.

The next test problem is a representative of the deceptive problems, i.e. problems that are intentionally designed to make GAs converge towards some local deceptive optimum. Here, the problem is composed of the deceptive function DF3 taken from [1.30], which is 4-bit fully deceptive function with one global optimum in the string 1110 of the fitness 30. The function has a deceptive attractor 0000 of fitness 10, which is surrounded, in the search space, by four strings containing just one 1 with quite high fitness values 28, 27, 26, and 25. The problem is formed as concatenation of 50 DF3 functions resulting in a 200-bit long chromosome. Thus the global optimum of the problem is 1500. The definition of the search space of the DF3 function is shown in Fig. 1.4.

The used Royal Road problem (RR) is a 16-bit version of the RR1 single-level royal road problem described in [1.7]. The problem is defined by enumerating the schemata, where each schema s_i has assigned its contribution coefficient c_i. The evaluation of an arbitrary chromosome is given as a sum of all contributions of those schemata that are covered by the chromosome. The used RR problem is defined as a concatenation of ten 16-bit schemata of all ones. All building blocks have the equal contribution 16. Only the combination of all ones on the bits pertinent to a given schema contributes to

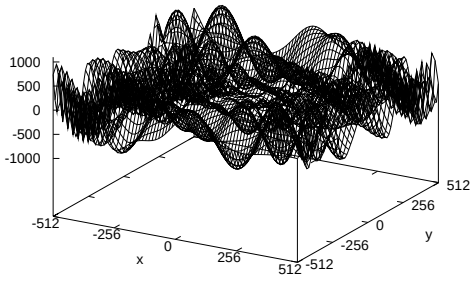

Fig. 1.3. Non-linear and highly multimodal function F101(x, y)

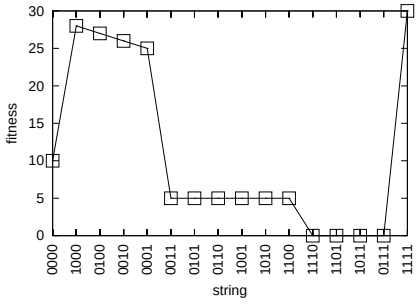

Fig. 1.4. Deceptive function DF3

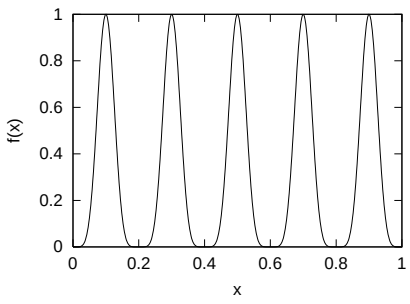

Fig. 1.5. Deb's periodic function F1(x)

the fitness with the nonzero value, any other combination has value 0. So the optimal solution is the string of all ones with its fitness 160.

Another test problem used in our experiments is a representative of hierarchically decomposable problems, namely the hierarchical-if-and-only-if function (H-IFF) proposed in [1.29]. The hierarchical block structure of the function is a balanced binary tree. Each block, consisting of two sub-blocks (a pair of its children), contributes to the overall fitness by the value, which depends on its interpretation (*meaning*) and its level in the tree. Each leaf node, corresponding to a single gene, contributes to the fitness by 1. Each inner node x is interpreted as 1 if and only if its children are both 1's, as 0 iff they are both 0's. In such a case the node contributes to the fitness by value $2^{height(x)}$, where $height(x)$ is the distance from the node x to its antecedent leaves. Otherwise the node is interpreted as *null* and its contribution is 0. It follows from its definition that the function has two global optima - one consists of all 1's and one of all 0's. For our purposes the 256-bit H-IFF function with the global optima of value 2304 was used.

The last test function is the function F1 taken from [1.20] that was used to analyse the ability of the GALCO algorithm to maintain multiple optimal solutions in the population. The function is defined in the interval (0.0, 1.0) as follows:

$$F1(x) = sin^6(5\pi x),$$

where the parameter x is coded on 30 bits so that the string of all zeros represents 0.0 and the string of all ones represent 1.0. The function is periodic with 5 equally-spaced maxima of equal height, see Fig. 1.5.

1.5 Experiments

This section presents an empirical analysis of the GALCO algorithm and shows some interesting aspects of this approach. The series of experiments were carried out to reveal how the factors such as the convergence rate C,

the population size *PopSize*, and the utilisation of the *replace_with_mask* operator affect the performance of the algorithm. GALCO is also compared to the standard form of the incremental genetic algorithm (SIGA). Here, SIGA always replaces the worst individuals of the population by the generated offspring. Contrary to the GALCO, SIGA employs a simple mutation operator. The comparisons are based upon the quality of the achieved solutions as well as the convergence characteristics of the algorithms. Both algorithms run for the same number of fitness function evaluations, which is a commonly used condition when different evolutionary algorithms are compared. Each experiment were replicated 20 times and the average best-of-run values and average convergence courses are presented in tables and graphs.

Let us first focus on how the performance of the algorithm depends on the chosen convergence rate C. Results achieved with various values of C are in Tab. 1.1. We can observe that small values of C (means that rather low convergence is allowed) give better results than the bigger ones in general. This confirms our assumption that the less the population can get homogenous the higher the chance that better results will be generated is. However this assertion holds just when rather extreme values of C are considered. The trend across the whole interval of C is such that starting from big C the performance improves as the C decreases until some optimum value of C is reached (results written in bold). Further when the C decreases the quality of the obtained solutions does not improve any more whilst the time needed to find the optimal solution increases.

Apparently this is caused by the fact that the more diverse the population is kept (an extreme case $C = 0$ is discussed further) the more the convergence towards the optimal solution slows down. Thus we can see that the average best-of-run value achieved with $C = 1$ is worse than that achieved with the optimal $C = 20$ in the case of F101. In the case of DF3 and H-IFF problems the optimal average best-of-run values were achieved even with $C = 1$ but it

Table 1.1. An effect of varying convergence range on a performance of the GALCO

C	F101	DF3	H-IFF	RR
225	-918	1480	1344	26
200	-920	1490	1675	37
100	-930	1496	2304	56
50	-941	1500	2304	140
20	**-943**	**1500**	**2304**	150
10	-942	1500	2304	152
5	-942	1500	2304	154
1	-941	1500	2304	**160**
0	-927	1498.5	1453	86

* The population size 500 was used.

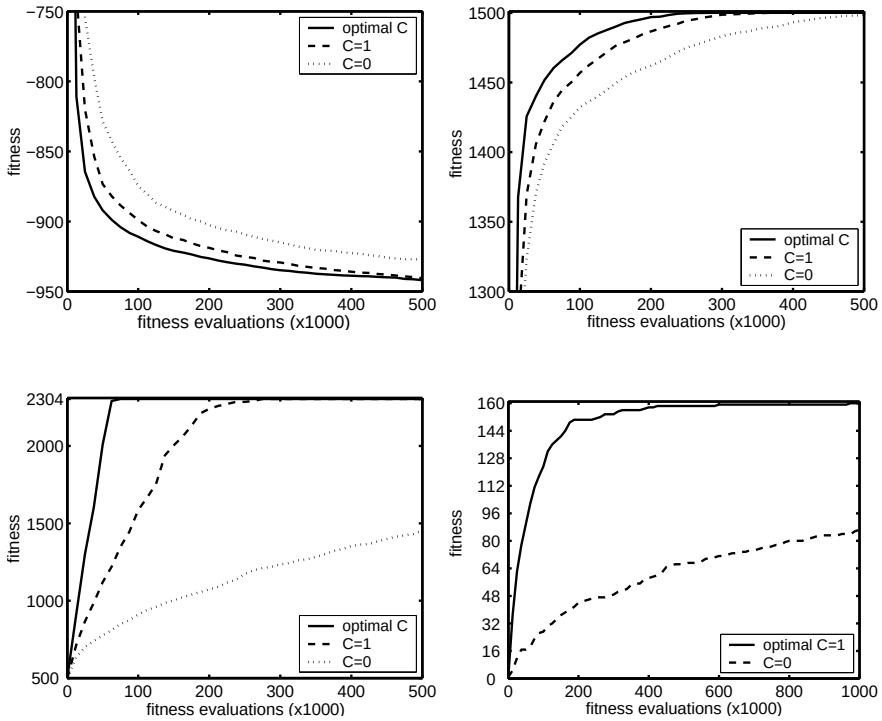

Fig. 1.6a–d. Mean convergence curves obtained with the optimal convergence rate C, convergence rate C=1 and C=0 on the problem F101 (**a**), DF3 (**b**), H-IFF (**c**) and RR (**d**). The population size 500 was used

took considerably longer time to find them. See Fig. 1.6, where the average convergence curves with the optimal C and $C = 1$ are shown.

Let us have a look at the results on RR problem now. This is the problem that requires the least convergence rate to get the best results. This results from the nature of the benchmark. There are no smaller building blocks available besides those predefined 16-bits ones, which would guide the algorithm towards the complete optimal solution. So the algorithm can only rely on maximal diversity of individuals that would maximise a chance that all building blocks will be generated and combined in one optimal chromosome.

It is also interesting to see the effect of the *replace_with_mask* operator. The results of experiments carried out with disabled replacement operator are in the last row of Tab. 1.1 where $C = 0$. It is evident that the operator greatly improves the performance of GALCO if used with a reasonable frequency. There are considerable differences in average best-of-run values obtained on problems F101, H-IFF and especially on RR. In the case of the deceptive

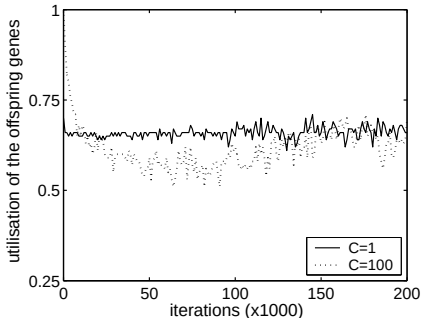

Fig. 1.7. Demonstration of a utilisation of the genetic material of the offspring undergoing the *replace_with_mask* operation. Graphs show an evolution of the ratio of genes taken from the offspring to the total number of genes, in which the chromosomes that are merged by the *replace_with_mask* operator (the offspring and the worst individual in the population), mutually differ. The used population size was 250. The view of the first 200.000 iterations of the algorithm is provided

problem the best-of-run solutions achieved without *replace_with_mask* operator are quite close to the optimum. However, the two variants of GALCO algorithm, the one that employs the *replace_with_mask* operator and the other one that does not, differs in the number of function evaluations needed to come to the optimum solution. This is shown in Fig. 1.6.

These results confirm our intuition that not only a crossover operator applied explicitely to promising parents enables the good building blocks to combine. Also a mixing of the genetic information of two rather poor chromosomes by the replacement operator is very fruitful. One should see that in this way the worst individual in the population is given a high chance that its genes or complete building blocks will be combined with other genetic material. Note that without the replacement operator such a chance would be very small since the worst individual is almost unlikely to be chosen as a parent.

Fig. 1.7 shows an evolution of a utilisation of genes taken from the offspring chromosome. We can see that the characteristics differ from each other in a way one would expect. When a small convergence rate is used (i.e. $C = 1$) the proportion of genes taken from the offspring is stable (around the value $2/3$) during the whole run. The bigger C is used (situation for $C = 100$ is shown here) the more of the offspring genes can be used in the early stages of the run. At the beginning of the run the offspring chromosome completely replaces the worst chromosome of the population. As the population converges i.e. the distribution of ones and zeros in the population columns reaches the boundary values (either $PopSize - C$ or $PopSize + C$) the utilisation of the offspring genes drops down and then stabilises around the value $2/3$ as in the former case. Note, that the proportion of the worst individual genes is the complement to 1.0. So a considerable number of genes of the worst individual are used as well.

Another factor, which strongly affects a performance of GAs, is the size of the population. General trend in standard GAs is that the bigger the used population is the better results can be achieved. This is because the big population provides better supply of necessary fundamental building blocks when

Table 1.2. An effect of varying population size on a performance of the GALCO

PopSize	F101	DF3	H-IFF	RR
20	-928	1500	2304	68
50	-937	1500	2304	147
100	-943	1500	**2304**	154
150	**-949**	**1500**	2304	159
200	-946	1500	2304	160
250	-945	1500	2304	160
500	-941	1500	2304	160
1000	-928	1500	2304	**160**

* The convergence range $C = 1$ was used in these experiments.

it is initialised and is less subject to premature convergence. On the other hand, the evolution of large population towards high-quality solutions may be too slow and at the expense of more computations performed (measured for instance by a number of performed fitness function evaluations or crossover operations).

The results of experiments carried out in order to investigate a sensitivity of the GALCO to population size are presented in Tab. 1.2.

There are two important observations. First, the performance of the algorithm depends on the population size in a similar way as other GAs. As the population size increases the quality of the obtained results improves until an optimal population size is reached (the results written in bold). Further increasing of the size slows down the convergence to the optimal solution. In the case of F101 much worse solutions were found with population size 1000 than with the population size 150. Similarly much slower convergence to optimal solutions were observed when large populations ($PopSize > 200$) were used on DF3 and H-IFF problems. Again when solving the RR problem the best results were achieved with the biggest tested population size. A strange phenomenon can be observed on the DF3 problem. There, the optimal solution was found even with a small population of 20 individuals. Such an unusual success of the algorithm on this benchmark could be easily

Table 1.3. Results of the SIGA algorithm with varying population sizes

PopSize	F101	DF3	H-IFF	RR
500	-877	1449	1234	149
1000	-907	1464	1348	150
1500	-909	1473	1382	144
2000	**-912**	**1476**	**1501**	147

* Other parameters of the SIGA were $P_{cross} = 1.0$, $P_{mut} = 0.01$, and the number of fitness function evaluations was 500.000.

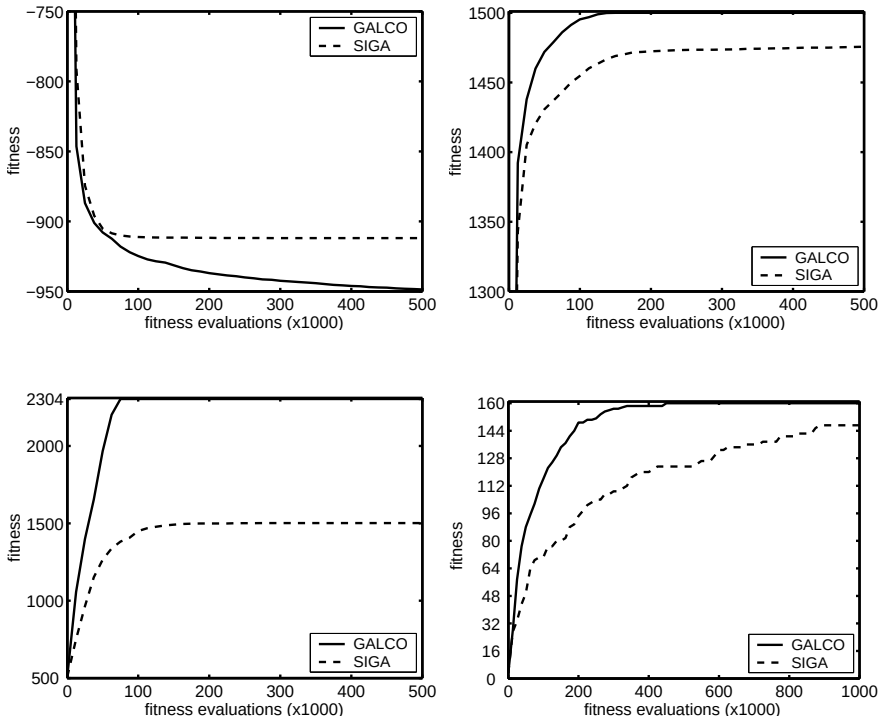

Fig. 1.8a–d. Comparison of the convergence characteristics of GALCO and SIGA on the problem F101 (**a**), DF3 (**b**), H-IFF (**c**) and RR (**d**). It shows the convergence characteristics corresponding to the experiments marked bold in Tab. 1.2 and Tab. 1.3, respectively.

explained. Recall that the DF3 function is a fully deceptive function with the deceptive attractor in the string, which is a binary complement of the global desired optimum. This means that the deceptive building blocks (those with all 0's or at most one 1) tend to gradually proliferate in the population. However the saturation of the population with those building blocks is limited due to the $C = 1$. So in reaction to the increased number of deceptive building blocks the number of building blocks composed of all 1's grows as well in order to keep the balance of 1's and 0's in the population. Obviously such a mechanism should work whenever a string representing a local optimum is close to the binary complement of the global optimum string.

The second observation is that the algorithm does not require very large populations to be able to solve the problems. In fact rather small populations (100-200 for our test problems, except the RR problem) can be used with GALCO to effectively search a complex solution space. This becomes evident when compared to the results obtained with the standard incremental GA

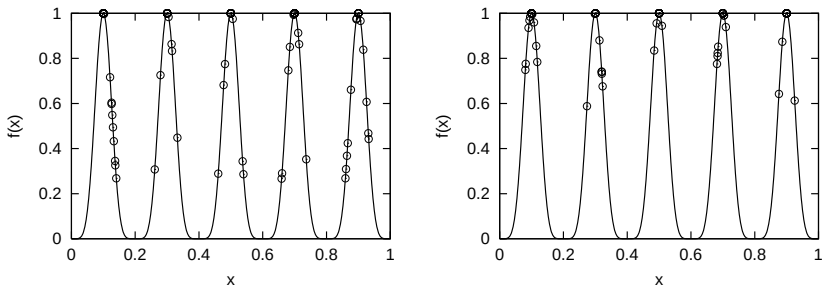

Fig. 1.9a–b. The final distribution of 100 solutions after 50000 (**a**) and 500000 (**b**) fitness function evaluations. Proportions of the population after 50000 and 500000 evaluations are 24-18-19-19-20 and 26-16-22-17-19, respectively

(SIGA), see Tab. 1.3. Whilst the SIGA needs a large population to come up with good solutions the GALCO profits from its inherent explorative power described above. A comparison of the GALCO and SIGA in terms of the average convergence characteristics is shown in the Fig. 1.8. We can see that the GALCO outperforms the SIGA on all test problems. Especially remarkable difference between the two algorithms can be observed on F101, DF3 and H-IFF problems, where SIGA did not give any comparably good result even with population size 2000.

As we mentioned in the previous section some experiments were performed to test the GALCO algorithm on its ability to discover many different global optima. The plots in Fig. 1.9 show how the population samples the solution space of the F1 function after 50000 and 500000 performed fitness function evaluations. It turned out that the algorithm is able to maintain a number of samples for each peak of the function. As the evolution goes on the individuals in the population become better and better. An important observation is that the proportions of the population are almost stable during this process.

1.6 Summary and Conclusions

We have introduced a novel approach for preserving population diversity. It is based on an idea that the population is explicitly prevented from becoming too homogenous by simply imposing limits on its convergence. This is done by specifying the maximum difference between frequency of ones and zeros at any position of the chromosome calculated over the whole population. The algorithm is of a type of the incremental GA where only one crossover operation is performed in each generation cycle and the newly created individuals replace some individuals of the current population.

A specific replacement strategy is used to keep the desired distribution of ones and zeros during the whole run. The pair of newly generated chromo-

somes replace the parents if the best-fit chromosome out of those four ones is the child. Under such an replacement operation the population genotype stays invariant. Apparently the evolution of the population would be too limited if we relied on accepting only those new chromosomes, which improve the fitness of their parents. To further boost up the exploration power of the algorithm a special replacement operator was introduced, which is used to insert as much of the offspring genetic material as possible into the population. Actually the currently worst-fit chromosome of the population is merged with the new one. In fact this is just another recombination operation providing a means for better exploiting of genetic material of the poorly fit individuals.

The proposed algorithm was experimentally tested on a representative set of test problems. The results revealed several interesting aspects of the algorithm's behaviour. First, the best performance of the algorithm was achieved with rather low convergence rate ($C << PopSize/2$). This means that the distribution of ones and zeros is almost half-to-half in every column of the population. This is in agreement with the intuition that the more equal the distribution of ones and zeros is used the less the algorithm is prone to prematurely converge so the better results can be achieved. Due to its replacement-recombinative component the algorithm is able to go on to generate new sample points of the search space even after the global optimum has been found. It was also shown that the algorithm is capable of maintaining multimodal diversity of the population. So representatives of various optima can co-exist in the population during the whole evolution.

An interesting aspect of the proposed algorithm is that it does not require any tuning of the mutation or crossover probabilities. Apparently any probability of crossover less than 1.0 does not make any sense because once a pair of parents is chosen they should be crossed over otherwise the whole action does not have any effect. There is no explicit mutation operator used in the algorithm. The variability of the population genotype is maintained by preservation of the gene distribution combined with the high generative ability of the recombination crossover and *replace_with_mask* operators. For the same reason the algorithm work efficiently with rather small populations in comparison with standard GA. It was shown that from some value of the population size its further increasing is counterproductive and the process of finding the optimum solution slows down.

Concluding this paper we can say that the GALCO algorithm exhibits many nice features. However they were presented and confirmed only in an empirical way. Future research in this area should focus on theoretical analysis of the algorithm that would help us to better understand its behaviour and to estimate its convergence characteristics. Note that for our evaluation purposes we have used test problems composed of tightly linked building blocks. The applicability of the algorithm to the problems with low linkage, i.e. the problems with building blocks spread across the chromosome, should be studied as well.

Acknowledgements

The authors would like to thank Petr Posik for many valuable comments and Stepan Janda for his help with carrying out the experiments.

The first author acknowledges the support provided by the TU Delft within a frame of the Research Fellowship Program. The research was funded by the project No. 102/02/0132 of the Grant Agency of the Czech Republic.

References

1.1 Baker, J. E.: Reducing Bias and Inefficiency in the Selection Algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, 1987, pp. 14-21

1.2 Cavicchio, D. J.: *Adaptive search using simulated evolution*, Ph.D. Thesis, Univ. of Michigan, Ann Arbor, 1970

1.3 Collard, P., Gaspar, A.: Royal-road landscapes for a dual genetic algorithm, in W. Wahlster, editor, *ECAI 96: 12th European Conference on Artificial Intelligence*, Wiley & Son, 1996, pp 214-217

1.4 Collard, P., Escazut, C.: Fitness Distance Correlation in a Dual Genetic Algorithm, in W. Wahlster, ed., *ECAI 96: 12th European Conference on Artificial Intelligence*, Wiley & Son, 1996, pp. 218-222

1.5 De Jong, Kennet A.: *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral Thesis, Dept. of Computer and Communication Sciences, Univ. of Michigan, Ann Arbor, 1975

1.6 De Jong, K. A., Spears, W.: A formal analysis of the role of multi-point crossover in genetic algorithms, *Annals of Mathematics and Artificial Intelligence* (Switzerland: J C Baltzer A G Scientific Publishing Company), pp. 1-26, 1992

1.7 Forrest, S., and M. Mitchell.: Relative building-block fitness and the Building Block Hypothesis, in Whitley, L. D. (Ed.), *Foundations of Genetic Algorithms 2*, pp. 109-126. San Mateo, CA: Morgan Kaufmann., 1993

1.8 Goldberg, D. E.: Optimal Initial Population Size for Binary-Coded Genetic Algorithms,. TCGA Report No. 85001, University of Alabama, Department of Engineering Mechanics, 1985

1.9 Goldberg, D. E., Richardson, J. J.: Genetic algorithms with sharing for multimodal function optimization, *Genetic Algorithms and Their Applications: Proceedings of the Second ICGA*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987, pp 41-49

1.10 Goldberg D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989

1.11 Goldberg, D. E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms, in Rawlins, G. J. E. (Ed.), *Foundations of Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 1991, pp. 69-93

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

1.13 Goldberg, D. E., Sastry, K.: A Practical Schema Theorem for Genetic Algorithms Design and Tuning, IlliGAL Report No. 2001017, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2001

1.14 Harik, G., Cantu-Paz, E., Goldberg, D. E., and Miller, B. L: The gambler's ruin problem, genetic algorithms, and the sizing of populations, *Evolutionary Computation*, 7 (3), 1999, pp 231-253

1.15 Holland J. H.: *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975
1.16 Kubalík, J., Lažanský, J.: Partially randomised Crossover Operators, In: *Advances in Soft Computing*, R. John & R. Birkenhead (Eds.), Heidelberg: Physica-Verlag, ISBN 3-7908-1257-9, pp 93-98, 2000
1.17 Kubalík, J., Lažanský, J.: A New Genetic Operator Maintaining Population Diversity, in *AIP Conference Proceedings of CASYS '2000*, Dubois, D. (Ed.), American Institute of Physics, ISBN 0-7354-0012-1, pp. 338-348, 2001
1.18 Kubalík, J., Lažanský, J.: A New Genetic Operator with Enhanced Capabilities. In: *Quo Vadis Computational Intelligence?*, Sinck, P. (ed.), Physica-Verlag, Heidelberg, Germany, ISBN 3-7908-1324-9, pp. 305-310, 2000
1.19 Kubalík, J., Lažanský, J., Rothkrantz, L. J. M.: On Extending of Search Power of Genetic Algorithms by Means of Permanent Re-initialisation of Parental Common Schema, in *Proceeding of Mendel 2001*, Brno University of Technology, ISBN 80-214-1894-X, pp. 1-6, 2001.
1.20 Mahfoud, S. W.: Crowding and Preselection Revisited, *Parallel Problem Solving From Nature*, 2, R. Manner and B. Manderick (Eds), Elsevier Science Publishers (North Holland), Amsterdam, pp. 27-36, 1992
1.21 Mengshoel, O. J., Goldberg, D. E.: Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement, in *Proceedings of the Genetic and Evolutionary Computation Conference*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, R. E. Smith (Eds), Morgan Kaufmann Publishers, California, 1999
1.22 Michalewicz Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg, Second edition, 1994
1.23 Miller, B., Goldberg, D. E.: Genetic Algorithms, Tournament Selection, and the Effects of Noise. IlliGAL Report No. 95006, Urbana: University of Illinois, Illinois Genetic Algorithms Laboratory, 1995
1.24 Oei, C. K., Goldberg, D. E., Chang, S. J.: Tournament selection, niching, and the preservation of diversity, IlliGAL Report No. 91011, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 1991
1.25 Spears, W. M.: Adapting Crossover in Evolutionary Algorithms, in *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, J. R. McDonnel, R. G. Reynolds and D. B. Fogel, eds., MIT press, Cambridge, 1995, pp 367-384
1.26 Spears, W. M.: Recombination Parameters, in *Handbook of Evolutionary Computation*, T. Back, D. B. Fogel, and Z. Michalewicz, eds., New York: Oxford Univ. Press and Institute of Physics, 1997, pp E1:3:1-11
1.27 Srinivas, M., and Patnaik, L. M.: Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms, in *IEEE Transaction of Systems, Man and Cybernetics*, vol. 24, 1994, pp 656-667
1.28 Thierens, D., Goldberg, D. E.: Elitist Recombination: an integrated selection recombination GA. In *Proceedings of The First IEEE Conference on Evolutionary Computation*, Vol 1, ISBN 0-7803-1899-4, pp 508-512, 1994
1.29 Watson, R. A., Hornby, G. S. and Pollack, J. B.:. Modeling Building-Block Interdependency. *Parallel Problem Solving from Nature*, proceedings of Fifth International Conference PPSN V, Springer 1998, pp.97-106.
1.30 Whitley D.: Fundamental Principles of Deception in Genetic Search. In: *Foundations of Genetic Algorithms*, G. Rawlins ed., Morgan Kaufmann, 1991, pp. 221-241
1.31 Whitley D., Mathias, K., Rana, S., Dzubera, J.: Evaluating Evolutionary Algorithms, *Artificial Intelligence*, Volume 85, 1996, pp. 245-2761