Pergamon

*Computers Ops Res.* Vol. 25, No. 11, pp. 895–911, 1998

Printed in Great Britain
PII: S0305-0548(98)00024-0
0305-0548/98 $19.00 + 0.00

# THE MINIMAL DECEPTIVE PROBLEM REVISITED: THE ROLE OF "GENETIC WASTE"

Sonja Novkovic[1]†‡ and Davor Šverko[2]§

[1]Department of Economics, Saint Mary's University, Halifax, N.S., Canada
[2]Department of Mechanical Engineering, Concordia University, Montreal, Quebec, Canada

*(Received January 1997; in revised form February 1998)*

**Scope and Purpose**—Genetic algorithm (GA) is used in a number of fields for effective search of complex spaces, or as a learning algorithm. Based on natural genetics, it feeds on its ability to process large amounts of information very quickly. The performance of a GA is satisfactory in most cases, in that it finds solutions in the neighborhood of the global optimum. Some problems, however, are known to be GA-hard and deceive the GA to converge to sub-optimal solutions. We devise a version of GA through the addition of the "*genetic waste*", based on the perceived role of "*nonsense codons*" (or "noncoding segments"), which alleviates the problem by allowing sufficient diversity in the population. This modification allows the algorithm to change the parameters in each run without external interference. Furthermore, the parameters are independent of the fitness values and are thereby applicable to a wide class of problems.

**Abstract**—Using Goldberg's [10] minimal deceptive problem, we devise a simple method for setting the initial parameters of a genetic algorithm based on the role of nonsense codons in natural genetics. "Genetic waste" provides random parameters throughout the simulations, enabling the abandonment of trial and error search for the optimal combination of initial parameters and thereby adding credibility to the genetic algorithm as a tool of adaptive search. 

*Key words:* Genetic algorithm, control parameters, nonsense codon, minimal deceptive problem

## 1. INTRODUCTION¶

Genetic algorithm simulations [13] have been widely applied in recent years in a number of fields, either to capture the dynamics of adjustment, which reflects adaptive behavior and learning [14], or as an effective search algorithm in optimization problems. The mechanism of the simple genetic algorithm (GA) is guided by three operators: reproduction, crossover and mutation. Combining the survival of the fittest with randomized search, the GA creates individual strings in each new generation by exchanging parts of chromosomes between pairs of randomly selected mates from the old generation. The algorithm makes copies of strings with probability of selection proportional to their performance (measured by a fitness function), so that the more successful strings are more likely to contribute genetic material to the offspring of the next generation, while poorly performing strings are more likely to recede. Potential problems in the algorithm's performance mainly result in its convergence to sub-optimal solutions. Premature convergence may be attributed to insufficient exploration of the search space, leading to early domination of exceptionally good strings, while convergence to sub-optimal solutions later in the run may be attributed to insufficient exploitation of good solutions. One can view insufficient exploration as a lack of diversity in a population, implying

†To whom all correspondence should be addressed. Tel.: +1-902-4205607; Fax: +1-902-4205129; E-mail: s.novkovic@stmarys.ca.
‡Sonja Novkovic obtained a Ph.D. in Economics from McGill University and is currently Assistant Professor in Economics at Saint Mary's University in Halifax. Her research interests include genetic algorithms, bounded rationality and adaptive agents. She has published in *Computational Economics* and *Economic Analysis*
§Davor Šverko obtained M.A.Sc. in Mechanical Engineering from Concordia University. He is currently a research assistant, with interest in system dynamics and genetic algorithm optimization.
¶We wish to thank the anonymous referees for their valuable comments.

that above average solutions will "take over" too soon and valuable information will be lost in early generations. De Jong [8], Goldberg [11] and others introduced scaling to avoid premature convergence, while Booker [3] used variable crossover rates to alleviate the problem. Further, some problems are known to be GA deceptive ( [11], p. 45) due to a particular coding structure or due to the chosen fitness function. They are termed GA-hard if they lead the GA to a sub-optimal solution.

Generally speaking the GA as an effective search technique is intended to eliminate the need for a trial and error approach to the search of complex spaces, yet most of its applications include trial and error to determine the three essential parameters — mutation rate, crossover rate and seed for the random number generator. Very small variations in the initial parameter setting may significantly alter the point of convergence, since there is a large class of GA-hard problems in which the GA may indeed not converge to the vicinity of the global optimum. It is, therefore, difficult to justify the GA results when solutions are unknown *ex ante* and cannot be obtained by exact methods. Typically, one would replicate the GA a sufficiently large number of times on the same problem using different random number seeds. Similarly, numerous runs are needed to determine the best crossover and mutation probabilities for particular problems [4, 8, 9, 11, 12], for proposed ways to choose the two parameters). The purpose of our presentation is to avoid just that — the time consuming replication of GA runs, with no guarantee that the parameters chosen for replications ever produce the best results.

Our approach, instead, is based on natural gene structures, essentially leading to randomized crossover rate, mutation rate and the seed number selection, rather than exogenously pre-determined ones. We show that the deception due to the coding structure is alleviated when sufficient *parameter diversity* is introduced. For parameter setting we rely on "genetic waste", known as "nonsense codon" in natural genetics [2, 19] (see Section 4). Others have been drawing on the natural systems to enhance the performance of the GA [21, 22]. The use of "non-coding segments" in those works is similar to our approach in that it recognizes the importance of the untranslated portion of the gene structure, but it differs from "genetic waste" in that we define its specific role, while noncoded segments simply enlarge the string presentation with no specific assigned function. Both of these approaches improve the GA performance, but more research is clearly needed to establish the extent of these improvements. Grefenstette [12] and Srinivas and Patnaik [18], DeJong [7], Song *et al*. [17], Wu and Cao [23], among others, provide algorithms which optimize the parameters, many resorting to adaptive determination of the optimal parameter setting. The existing approaches in the literature are less satisfactory in two ways than the one presented here: first, they attempt to optimize the parameters, recognizing that there is a whole range of "good" parameter settings and a trade-off in performance as the control parameters are changed (probability of crossover and probability of mutation, in particular); and second, adaptive schemes depend on fitness values, making them unacceptable for a class of problems in which the fitness value for each string depends on the state of the population, as is the case in most economic problems.

In this paper, we combine the "genetic waste" operator with the preservation of the best string in the generation (the "elitist selection" operator), which enables us to store useful information. For problems which require convergence, we scale down the mutation rate as the algorithm approaches its final runs, to decrease the impact of potentially large mutation rates. With the combination of these three forces, we develop a widely applicable GA, whose performance is independent of the initial parameter setting. Our approach adds to the analytical complexity of the GA, so we resort to empirical analysis in this presentation.

The paper starts out with a brief description of the simple genetic algorithm, with proportional selection, one-point crossover and mutation operators, and the description of Goldberg's [10] minimal deceptive problem (MDP). We then illustrate some of the difficulties of GA execution which depend on initial parameter setting. Section 4 sketches the role of "nonsense codon" in genetics, and the way we perceive "genetic waste" as its alternative in the functioning of a GA, while we provide a detailed description of the algorithm with "genetic waste" in Section 5. Section 6 is an illustration of the robustness of the GA with "waste" and Section 7 concludes.

## 2. SIMPLE GENETIC ALGORITHM

The genetic algorithm is a randomized search algorithm based on evolutionary genetics. It is represented by a population of $n$ strings of finite length, $l$, whose elements (alleles) are coded, usually in binary alphabet $\{0, 1\}$. Its functioning is guided by three standard operators, namely selection, crossover and mutation. *Selection* operator is a random process of selection of a mating pool of individual strings, which will provide genetic information to the future generation. The standard form of selection [5, 11] is called proportional selection, where better than average performing strings have a greater probability of selection into the mating pool. Performance is measured by a fitness function ($f$). The process is an artificial version of the "survival of the fittest". Here, less than average performing strings, measured by their fitness value, have a greater chance of receding.

*Crossover* operator will exchange genetic material between a pair of strings with probability $p_c$. A point of crossover, $k$, is chosen somewhere in the interval $[1, l-1]$. Alleles are then exchanged between positions $k+1$ and $l$. For example, if we have two strings of length $l = 7$, crossing at $k = 4$. Two new strings emerge as follows:

$$1\,1\,1\,1\,|1\,1\,1 \Longrightarrow 1\,1\,1\,1\,0\,0\,0$$
$$0\,0\,0\,0\,|0\,0\,0 \Longrightarrow 0\,0\,0\,0\,1\,1\,1.$$

*Mutation* changes the allele with probability $p_m$, turning 0 into 1, and vice versa. This operator is a source of population diversification.

The three operators guide the GA in its randomized search. Three exogenous parameters have to be determined by the programmer, namely the random seed number which initiates the "roulette wheel" in the process of selection, the crossover probability and the mutation probability.

To explain the strength of the GA search, Holland introduced the notion of *schemata*, representing similarity templates which describe subsets of strings with similarities at certain string positions. With binary coding, a schema may be written as a string over the alphabet {0, 1, *}, where * is a "wild card" or a "don't care" symbol. A schema 0*1, for example, matches two strings {001, 011}. A population of size $n$ can hold up to $n2^l$ schemata, implying an immense amount of information processing within a relatively small population. The number of schemata contained within a population depends on its diversity, and while initially the information set is potentially huge, later in the runs as the algorithm converges, new information can be added only by mutation.

Holland's Schema Theorem establishes the impact of GA operators on the growth or decay of particular schemata in a population. The conclusion is that short, highly fit schemata (the *building blocks*) spread exponentially in the population [11], and this is considered to be one of the reasons for the power of the GA as a search mechanism.

## 3. INITIAL PARAMETER SETTING AND SOME PROBLEMS WITH GA PERFORMANCE

The power of short, above average schemata can be challenged in some cases, labelled GA-hard problems [20], when the GA converges to sub-optimal solutions. Goldberg devised a minimal deceptive problem (MDP) as an illustration of the potential for GA divergence from the global optimum. He concludes, however, that such occurrences are rare (i.e. a deceptive problem is rarely GA-hard) and that under a number of conditions the simple genetic algorithm (SGA) in fact does converge to the best result.

Before we illustrate the sensitivity of the GA to initial parameter setting and to a random number generator, let us briefly reiterate the MDP ( [10], reproduced in Ref. [11]). It is a two-bit problem which can cause the GA to diverge from the global optimum. In what follows, the number of fixed positions of a schema $H$ create its order $o(H)$, while its defining length, $\delta(H)$, is measured by the distance between the outermost defining bits. A schema $H = *1**0**$, for example, is of the order $o(H) = 2$ and the defining length $\delta(H) = 3$. Suppose a set of four,

order-two schemata are assigned fitness values as follows:

$$\begin{aligned} &**1*****1* \quad f_{11}\\ &**0*****0* \quad f_{00}\\ &**0*****1* \quad f_{01}\\ &**1*****0* \quad f_{10} \end{aligned}$$

Next we assume $f_{11}$ to be the global optimum, implying:

$$f_{11} > f_{00}; \quad f_{11} > f_{01}; \quad f_{11} > f_{10}.$$

To introduce deception, one wants at least one sub-optimal order-one schemata to be better than optimal, order-one schemata. In other words, one or both of the following conditions must be true:

$$f(0*) > f(1*), \qquad f(*0) > f(*1)$$

The deception thus consists of the globality condition, ($f_{11}$ is the maximum), and one deception condition, ($f(0*) > f(1*)$ is assumed to be true). Two types of deception are then possible:

Type I: $f_{01} > f_{00}$

Type II: $f_{00} > f_{01}$.

Goldberg shows that Type II MDP is a more difficult one for the GA without mutation and with sure crossover. He uses fitness values

$$f_{11} = 1.1; \qquad f_{00} = 1.0; \qquad f_{01} = 0.9; \qquad f_{10} = 0.5,$$

to illustrate that with equal proportions of all strings in the initial population ($P^0_{ij} = 0.25$, where $P^0_{ij}$ represents the proportion of the string $ij$ in population 0), the best string eventually takes over and the algorithm converges to the global optimum. When the initial proportions of the second-best schema (00) are large, it takes over and crowds out the best (11). Figures 1 and 2

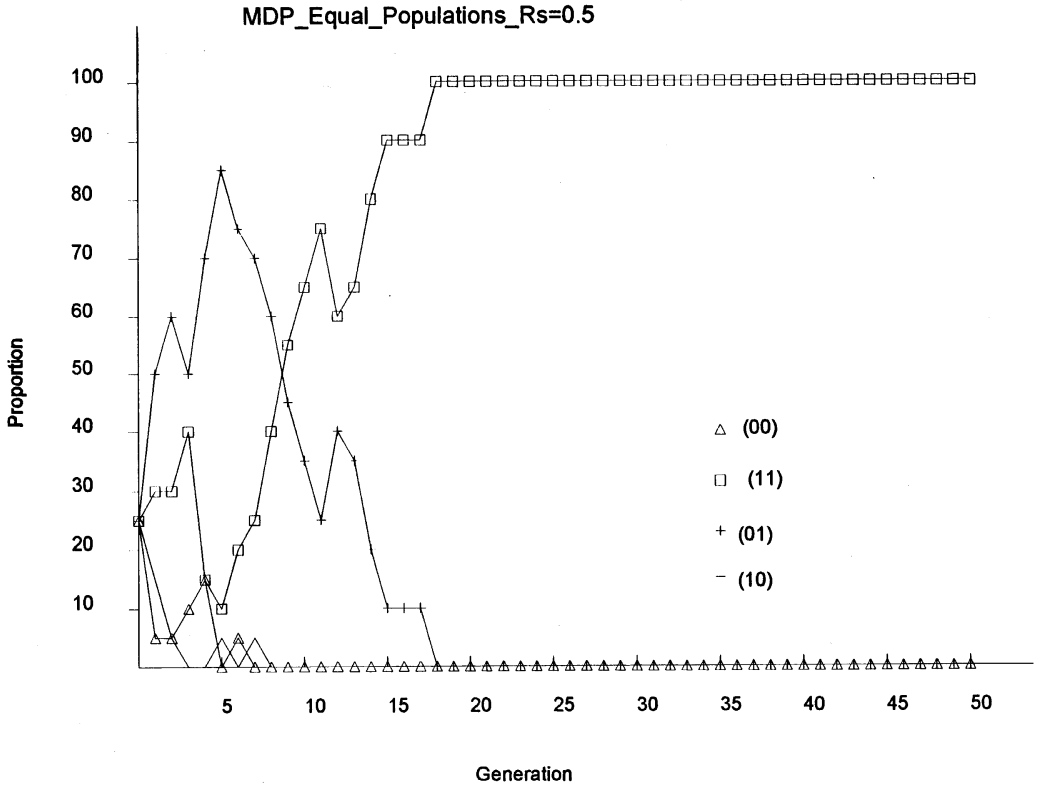


Fig. 1. MDP with equal initial proportions of the four strings; $r_s = 0.5$.

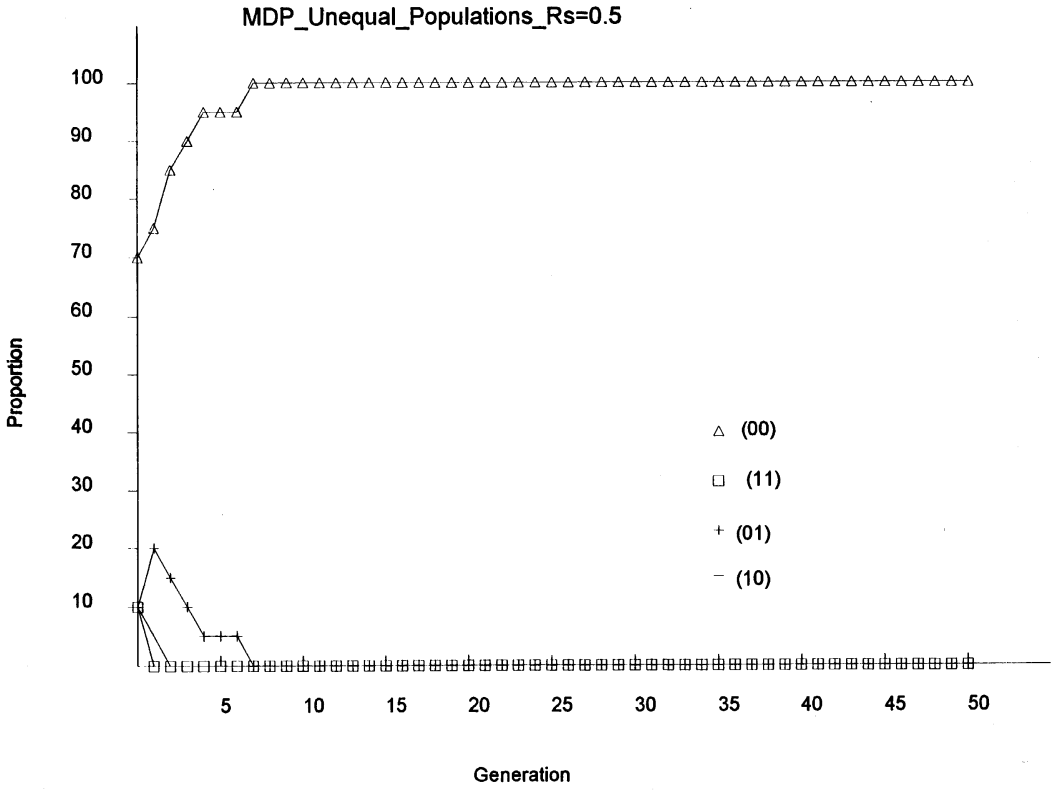


Fig. 2. MDP with unequal initial proportions of the four strings; $r_s = 0.5$.

illustrate these two cases, the former showing equal initial proportions, while the latter employs the unequal initial proportions $P^0_{00} = 0.7$, $P^0_{01} = P^0_{10} = P^0_{11} = 0.1$.

In the above simulations, we reproduced Goldberg's results with initial parameters fixed at $r_s = 0.5$ for the seed number, probability of mutation at $p_m = 0$, and crossover probability at $p_c = 1$. To check for sensitivity to the choice of the seed number, we further generated nine simulations with equal initial proportions of the four schema, with $r_s$ between 0.1 and 0.9, changing with increments of 0.1. In all cases, except for $r_s = 0.2$ and $r_s = 0.9$, the best schema (11) took over and the GA converged to the global optimum. In the other two cases, schema (01) took over very early and the GA converged to a sub-optimal solution, third in order of fitness values. Figures 3 and 4 illustrate these two cases — MDP for random seed 0.2 and 0.9 produces GA convergence to a different sub-optimal solution. Starting with equal proportion of the four schema in the initial population, schema (01) whose fitness value is $f_{01} = 0.9$ takes over (11) with $f_{11} = 1.1$ and (00) with $f_{00} = 1.0$.

The impact of a seed number and the random number generator on the GA efficiency is considerable, as is the impact of the other two parameters. With the probabilities of mutation and crossover fixed, changing the seed number will noticeably affect the performance even in this simple problem. As for the other parameters, Venkatachalam [20] uses Type II MDP to illustrate improved performance when crossover rate is selected to be either $p_c = 0.0$ or $p_c = 1.0$. When the lower rate is set at $p_c = 0.1$ instead of 0.0, the algorithm fails to converge to the global optimum.

We wish to point to another problem which emerges when parameters are chosen by trial and error. Suppose that $R$ represents the set of random seeds which lead the GA to the global optimum, $f_{11}$, while $R'$ is the complementary set of seeds which result in a sub-optimal solution. Further suppose that $f^T(r_s)$ represents average fitness of the final generation, $T$, with initial seed $r_s$. The two sets can then be defined as:

$$R = \{r_i \in (0, 1) | f^T(r_i) = f_{11}\}, \quad R' = \{r_j \in (0, 1) | f^T(r_j) < f_{11}\} \qquad R \cap R' = \{\varnothing\}$$

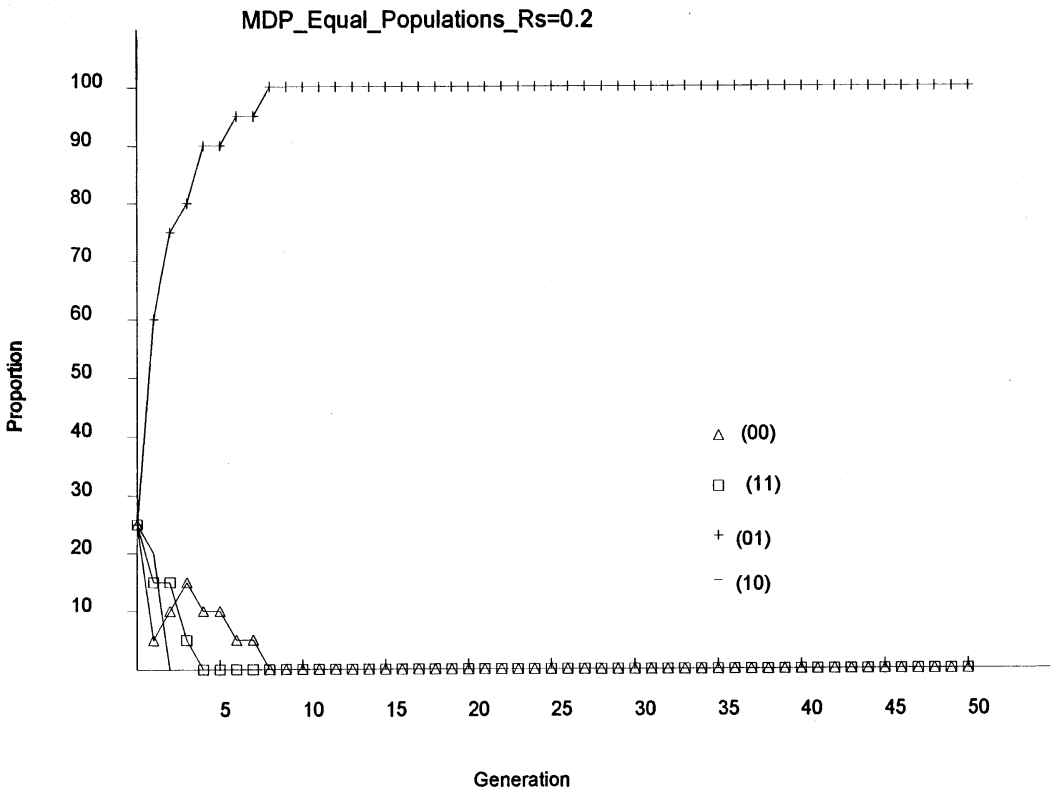


Fig. 3. MDP with equal initial proportions of the four strings; $r_s = 0.2$.

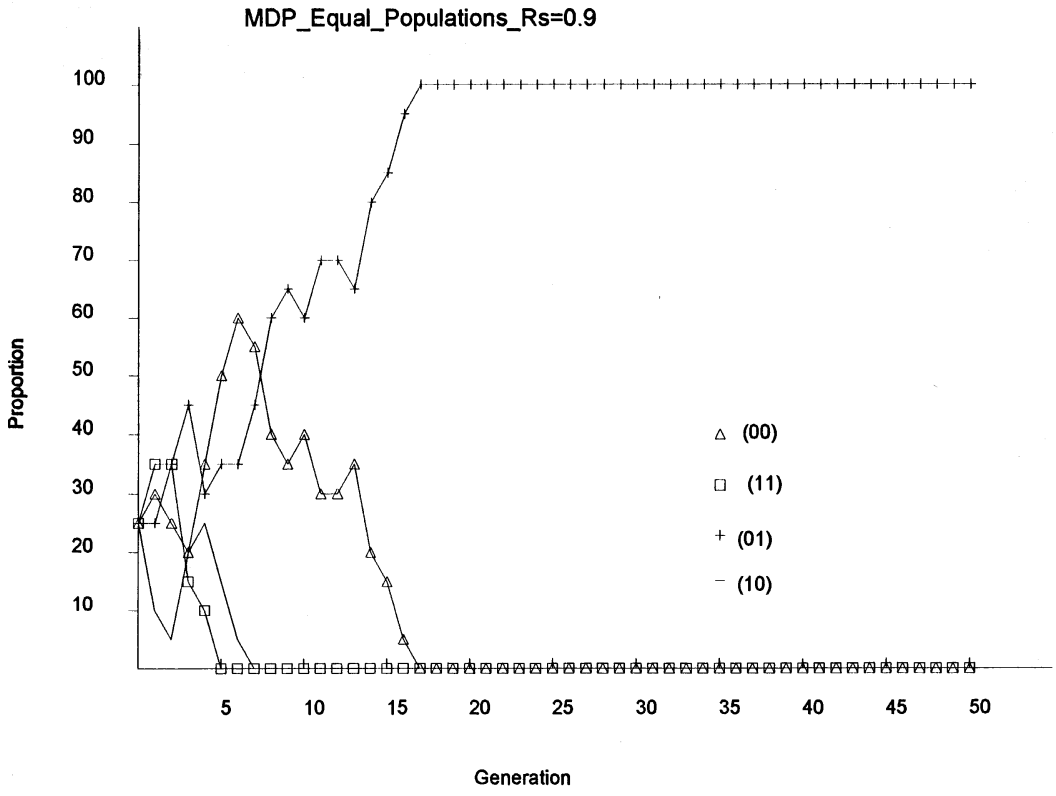


Fig. 4. MDP with equal initial proportions of the four strings; $r_s = 0.9$.

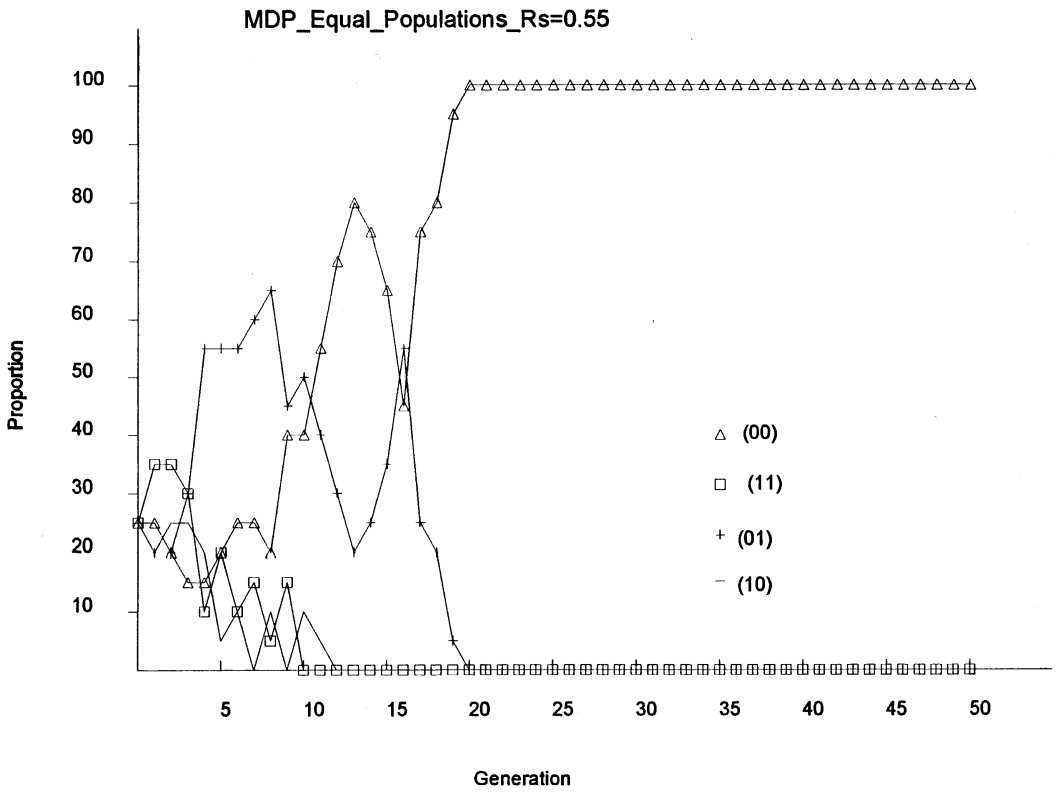


Fig. 5. MDP with equal initial proportions of the four strings; $r_s = 0.55$.

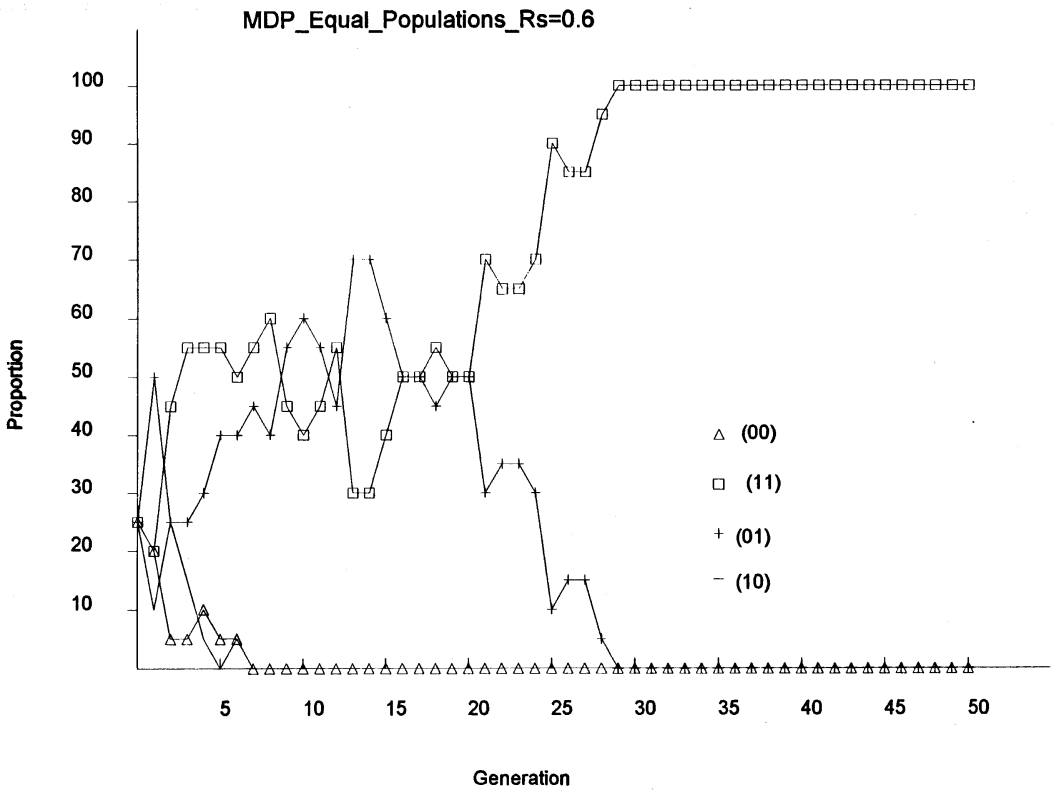


Fig. 6. MDP with equal initial proportions of the four strings; $r_s = 0.6$.

The two sets, $R$ and $R'$, are discontinuous, implying that we can always find a seed $r_j \in R'$ in the neighborhood of two adjacent rates $r_i \in R$ and $r_{i+1} \in R$, where 1 represents one step of chosen magnitude. In other words, while $r_i$ and $r_{i+1}$ both result in algorithm convergence to the global solution, we can find a rate $r_j$, applying a smaller step, which will lead to a sub-optimal result. Figures 1, 5 and 6 illustrate that rates 0.5 and 0.6 lead to schema (11) takeover, while $r_s = 0.55$ leads to schema (00) takeover, and thus to convergence to a sub-optimal solution.

The above example illustrates the importance of the choice of the random seed number (and the random number generator) for performance of the SGA. This dependence does not cease with more elaborate versions of the GA, as long as initial parameter values are fixed. Once we add changes in the rates of crossover and mutation†, the concern only becomes more profound.

## 4. NONSENSE CODON AND THE ROLE OF DIVERSITY

To solve the problem of initial parameter setting, we rely once again on natural genetics. The basis for our approach is in diversification and its source. DNA and some sequences of messenger RNA are the sole carriers of genetic information and therefore DNA is known to be the source of heredity. On the other hand, codewords which do not correspond to any material used in the message are known as "nonsense codon" see [2, 19] (here termed "genetic waste" for GA use). If we think about these non-translated portions (which have a definite role in the efficiency of translation) as sources of diversity (i.e. genetic material which cannot be directly attributed to the parents), we can include these untranslated sequences in the population and improve algorithm performance. In other words, while recombination of DNA enables evolution and learning, genetic nonsense is a source of diversity and thereby a source of random improvement of the "species".

As a diversity provider, nonsense translates into "genetic waste" in terms of the GA, which does not decode into fitness information, nor does it reproduce at exponentially increasing rates. We want to maintain diversity in the population, stressing the stochastic nature of the GA operators. This part of the genetic makeup provides population variance and ensures that the performance of the GA does not depend on the programmer's choice of initial parameters, a role which proved to be so vital in the search for the best solutions. The importance of the genetic waste is twofold: one, it ensures random selection of parameters throughout the runs, thus relieving the researcher of engineering the impossible — finding the best combination of parameters and it reinforces the credibility of the GA as a search method, in particular with problems whose solutions are unknown *ex ante*. Inclusion of genetic waste to create a GAW (Genetic Algorithm with genetic Waste) increases algorithm robustness, entirely removing its dependence on initial parameter setting. This approach differs from adaptive parameter setting [18] in that the parameters are independent of fitness values; they are not adaptively determined, but random throughout the runs.

A difficulty may arise with GAW when one considers problems which require population convergence, as the genetic waste is set up to provide variable mutation rates within a specified range. While mutation is useful for population diversification and, therefore, for processing a larger pool of information, later in the runs we need to decrease its impact in order to maintain uniformity. For that reason one can combine genetic waste with a scaling function, to decrease the effect of potentially disruptive mutation. We describe one such "damping" function in the next section. In problems where the search for the best solutions does not require convergence, the GAW can be used very effectively as a stand-alone tool in the exploration of the search space.

## 5. GA WITH GENETIC WASTE (GAW)

In addition to the standard operators — selection, crossover and mutation, the GAW incorporates the "genetic waste" (GW) part of the chromosome, which is decoded separately,

†See Novkovic [15] for an illustration of GA sensitivity to mutation rates in a five-period simulation model. Most of the literature deals with crossover and mutation only, choosing the seed number by trial and error (Refs. [8, 9, 12, 18, 20] and others).

not affecting the fitness value, and which provides random parameters (here the random seed number, mutation probability and crossover probability) that are different in each generation. Each string of length $L$ in a population of $n$ strings contains an "active" part (termed "active string") of length $l$ and the GW part of length $(L-l)$. We focus here only on the GW part of the string, since coding of the active string depends on the problem, while the GW remains universal in any kind of problem solving by GA.

The following operations are included in the application of GW:

1. coding
2. selection mechanism
3. application of parameters created by GW to selection of active strings
4. identification and maintenance of diversity in GW

In the initial population the GW part of the string is randomly chosen, together with the active string. It is then decoded in three parts: alleles $(l+1)$ to $(r)$ as the random seed number, $(r+1)$ to $(m)$ as the probability of mutation and $(m+1)$ to $(L)$ as the crossover probability. The length of each of the parts depends on the computing abilities at hand, as well as the wanted range of values for the parameters. The specific process applied here can be described as follows:

(a) *The GW part of the string* is randomly created in the initial generation (generation 0), like the active string. Seed for the random number generator for the initial population is an exogenous parameter. Selection for the mating pool of the first generation (generation 1) employs a different seed for each string, so that the seed from GW is taken from the string in generation 0 which is at the same position as the one being selected for generation 1. If mate number 3 is being selected, for example, then the GW of the string number 3 in the previous generation is used for the seed number. It is worth emphasizing that the selection procedure of mates for the GW is random and that it is *not* proportional to the fitness value. Crossover of the GW occurs with certainty ($p_c = 1$), while for mutation of this part of the string a different probability of mutation is used ($p_m$), one from each mate.

(b) *The active part of the string* is initially created using the exogenous random seed. The selection procedure is proportional to fitness; the seed number of the string corresponding to the position of the one being created is used, as above. Thus, selection of mates is separate for the two string parts, as is their crossover and mutation. Crossover probability of the second mate is then applied, while the probability of mutation for each child is used from each mate's GW.

We now turn to each of the three parameters:

*Random seed* number initiates the roulette wheel selection. In order to be able to repeat the simulation, one uses a pseudo-random number generator, trading off some efficiency in the search process. We differentiate between the pseudo-random number (PRN) and "real"-random number (RRN) according to the way in which they are generated. The RRN is generated by the time function of the code compiler, triggering the generator in a non-repeatable sequence, while the PRN generator ensures exact imitation of the sequence when the code is restarted. A simple GA uses an initially fixed random seed number to trigger the PRN generator. Therefore, "random" search through the domain, and the selection of individuals into the mating pool will be limited. To make up for limitations of a fixed $r_s$ in SGA, one can increase the population size and/or mutation rates. While this will diversify the population, it may be computationally costly, as well as inefficient. Here, we randomize the random seed itself, so in each generation a different "trigger" is applied, ensuring a more diversified selection process. Notice that the GW may be used to generate random numbers directly, instead of the $r_s$, which triggers the wheel.

Proportional selection ensures that the more fit individuals get a greater probability to be selected into the mating pool. The process of selection of the GW part of the string, as pointed out earlier, is independent of the measure of fitness. Rather, mates for the pool of strings which provide genetic material for the GW are randomly selected, ensuring that the GW part of the string creates a whole new population of parameters in each generation.

*Probability of mutation*. The SGA version of the algorithm uses the mutation probability ($p_m$) set at a fixed rate. Once the population converges, mutation is the only source of new information, but it also disrupts algorithm convergence. Therefore, in problems where search for equilibrium typically requires convergence, one can change the rate of mutation to decrease

over time. Some authors apply two fixed rates at different times, others introduce an exponentially decreasing rate. In both cases, mutation has to be predetermined initially.

For stochastic mutation rates, we use alleles ($r + 1$) to ($m$) of the GW string to decode them into $p_m$. The number of alleles used in this procedure limit the range of mutation probabilities, but the possibilities obviously are enormous. The stochastic nature of mutation probabilities makes the mutation operator potentially even more disruptive to convergence, but it greatly increases the wealth of information contained in the population and enables the GA to convey a more effective search. In order to decrease the disruption we apply an additional operator which curbs mutation. We find that the most appropriate scaling will cause no disruption to mutation for a long time, during which substantial information can be processed, while it declines its impact sharply late in the runs, not to hinder convergence. We therefore multiply the random $p_m$ by an arctan function, normalized to the [0, 1] interval, and specified in Appendix A. With this procedure, random mutation rates are applied in about two thirds of the total number of generations, after which point the (still randomly) selected probabilities of mutation are decreased by half and continue to decline. Figure 15 in Appendix A illustrates the shape of the scaling function with variable factors, which can be chosen by the programmer, to decrease the impact of mutation at a desired rate.

*Probability of crossover* is the third parameter decoded from the GW, alleles ($m + 1$) to ($L$). Like the other two parameters, the $p_c$ is random rather than fixed exogenously as in the SGA. Even though $p_c$ has proven to be less disruptive to SGA than mutation and is typically set at $p_c = 1$, we find it useful to randomize the crossover probability as well. Dorsey and Meyer [9] discuss optimal parameter setting on a range of problems and find no statistical evidence for rejection of a customary $p_c = 1$. They do observe, however, that the best rates vary across problems and that for some problems significant differences in performance result when different crossover rates are applied. We argue that one cannot trust the GA result when the outcome is unknown if different rates produce different solutions, simply because it is impossible to check all the combinations of parameters by trial and error. This is particularly true for context dependent functions, for which an individual's fitness depends on the state of the population. Economic problems with adaptive agents, for example, mainly contain such functions, since individuals affect each other's fitness. In our version of the GW, crossover probability can range from 0 to 1, changing with the step of 0.016, and thus giving the GA a much larger set of parameters to search than we could possibly explore by trial and error. We set the range for mutation probability between 0.1 and 0 (step 1/10240), and random seed number, like the crossover, between 0 and 1.

*Diversity*. Creating a randomized procedure for the selection of parameters, allows us to increase the effectiveness of GA search. Diversity proved to be essential in simulations for the GA to capitalize on the wealth of information provided by various schemata. For the GA to conduct a highly explorative search through its stochastic operators, the operators need to maintain high variance. In order to ensure that they do, we continually perform calculations of standard deviation, re-generating GW if diversity falls below a given threshold. Specifically, GW will be regenerated if coefficient of diversification (standard deviation/mean) falls below 30%, but typically, no regenerating is required, since the GW-created diversity is much larger than that.

Before turning back to the minimal deceptive problem, let us reiterate some important points. According to the Schema Theorem, the reason for the effectiveness of the GA is in its implicit parallelism, whereby the algorithm explores a wealth of information of a much larger size than the population itself and processes above average performing schemata in exponentially increasing numbers. The Schema Theorem itself is not sufficient for best performance. There is a consensus, even though analytical proofs have yet to be developed, that the GA has more success with difficult problems if there is a way to preserve useful information, while diversifying the population for learning.

We find that a combination of the following elements assures satisfactory performance: sufficient diversity, preservation of useful information and decline in random mutation rates (when convergence is required). The source of diversity we find to be in changing parameters contained in the "genetic waste", which then induces variability in the population of chromosomes, due to higher rates of mutation and crossover, on average. While multiple point

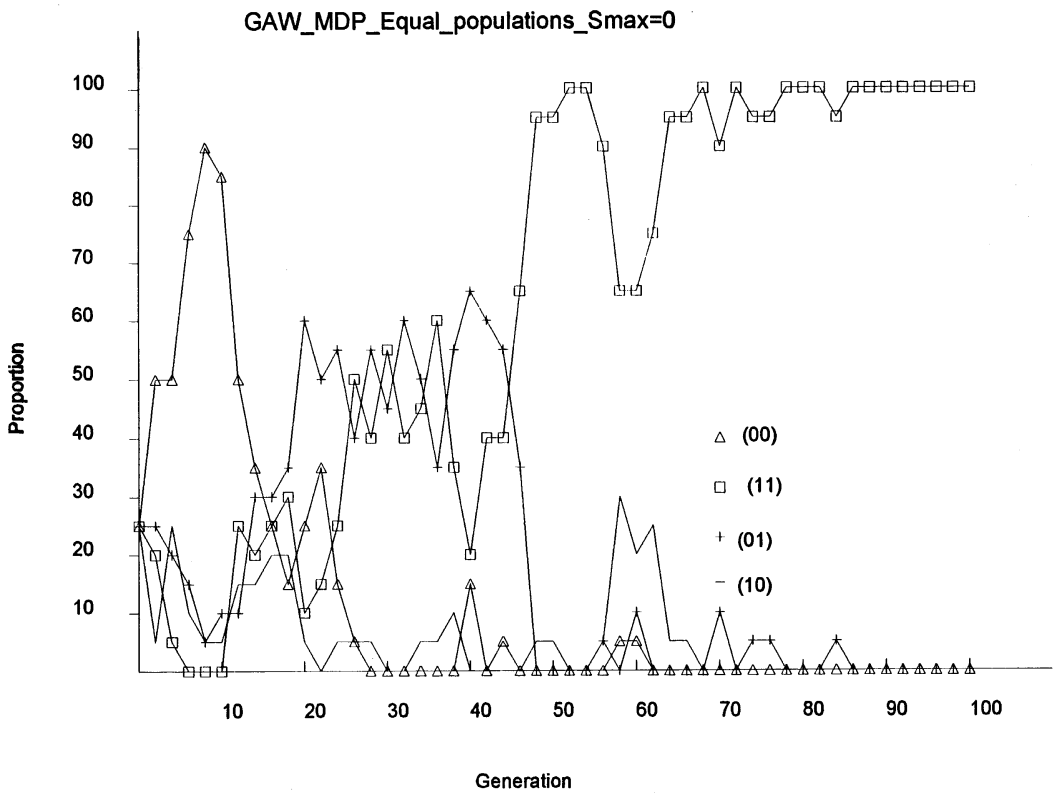


Fig. 7. MDP with equal initial proportions of the four strings — GAW.

crossover, for example, may degrade the performance of the GA [8], random probability of (a single point) crossover improves it. Therefore, we use GAW, scale mutation rates according to Fig. 15 when convergence is required and add elitist selection in order to preserve useful information as follows:

If $a^*(t)$ is the best individual in population $A(t)$ of size $n$, in population $A(t+1)$ we create $(n-2)$ new strings, while two copies of $a^*(t)$ are automatically transferred.

We term this elitist selection operator $S_{\max}$ in what follows.

## 6. THE MINIMAL DECEPTIVE PROBLEM REVISITED

We now turn to the MDP. As illustrated in Section 3, the simple version of the model applies only selection and crossover, where proportional selection depends on relative fitness, as well as the seed for the random number generator. When one includes a fixed mutation rate, the algorithm performance is unlikely to improve†.

Using the GAW version of the algorithm, we include the three parameters in the genetic waste. For comparison with the simple version of the MDP, we first run the algorithm without mutation and elite selection and then we examine the impact of addition of these two operators. When GW is used for crossover and random number generation only, deception can be expected to remain when the initial population contains large proportions of the second-best string. This is so because our random crossover rates are smaller, on average, than the $p_c = 1$ in the original version of the model. There is no reason for this alone to greatly improve algorithm performance, aside from the fact that one need not adjust the rates in search of the best results.

When we include the mutation probability, the algorithm will always find the best solution, whether it existed in the initial population or not. Whether the algorithm converges to the global optimum depends on the disturbance itself and decreasing the impact of mutation will

† We run the MDP with $p_m = 0.001$, 0.01 and 0.1 with no improvement when initial proportions of the second-best string are large. Larger rates of mutation will change the population structure, but there is no convergence because of the large disruption.

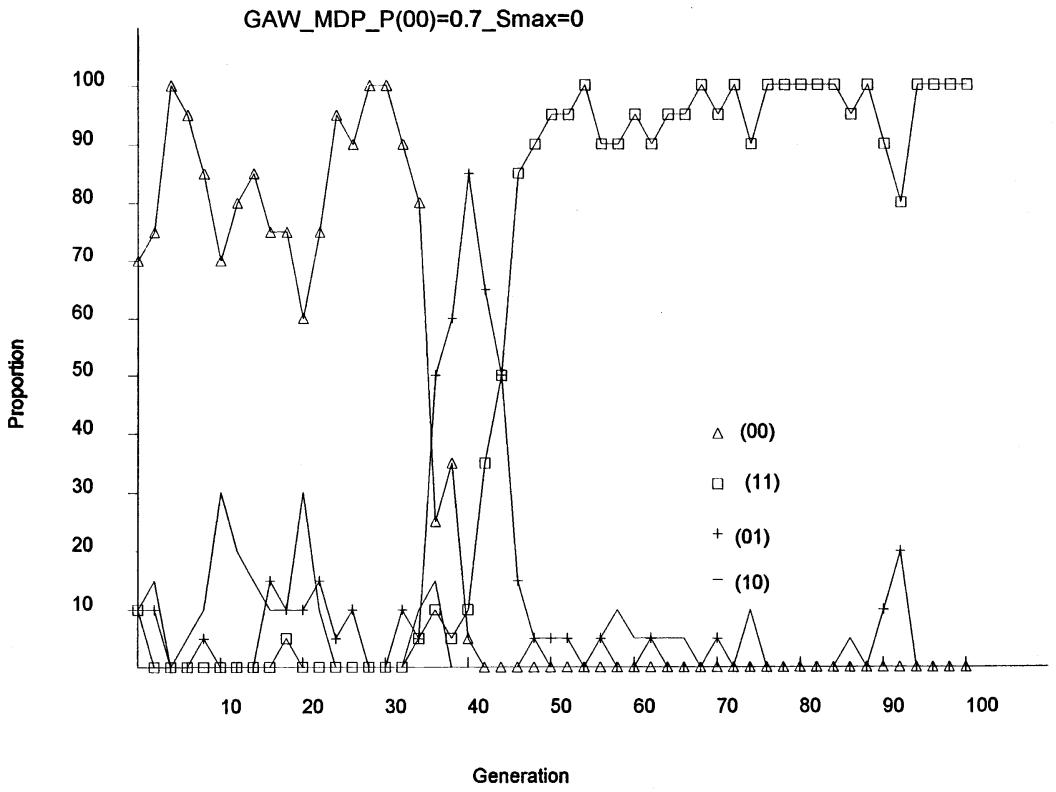


Fig. 8. MDP with unequal initial proportions of the four strings — GAW.

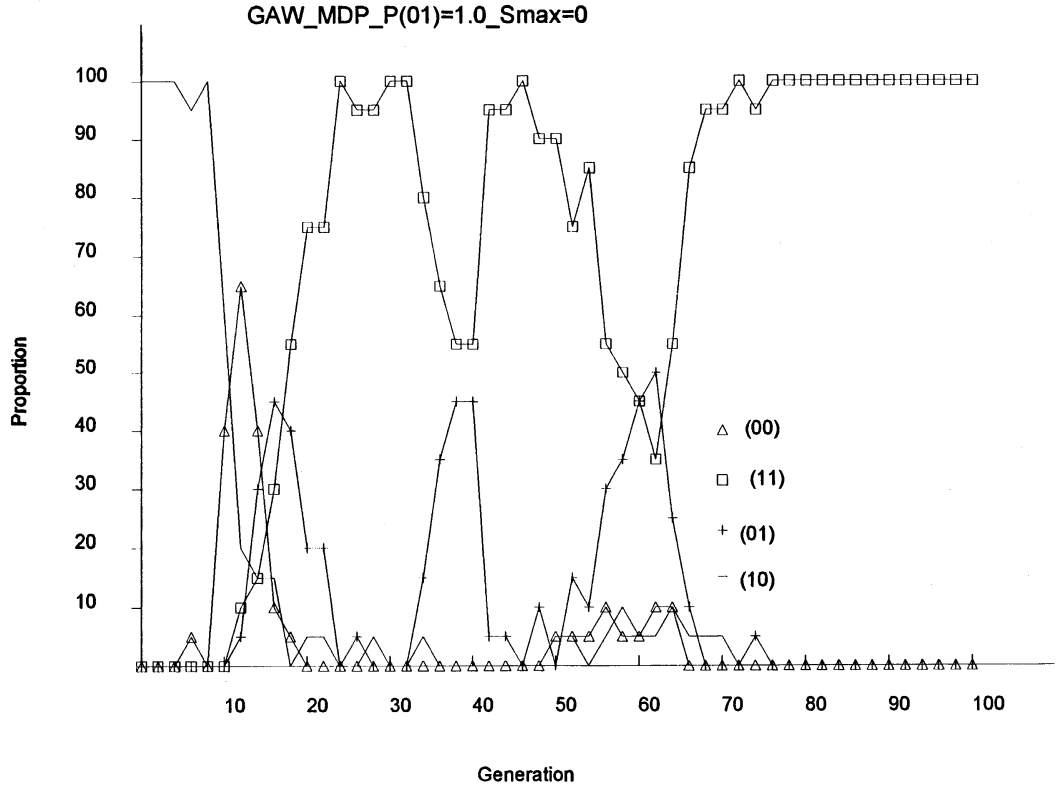


Fig. 9. MDP with initial $P_{10} = 1$; GAW.

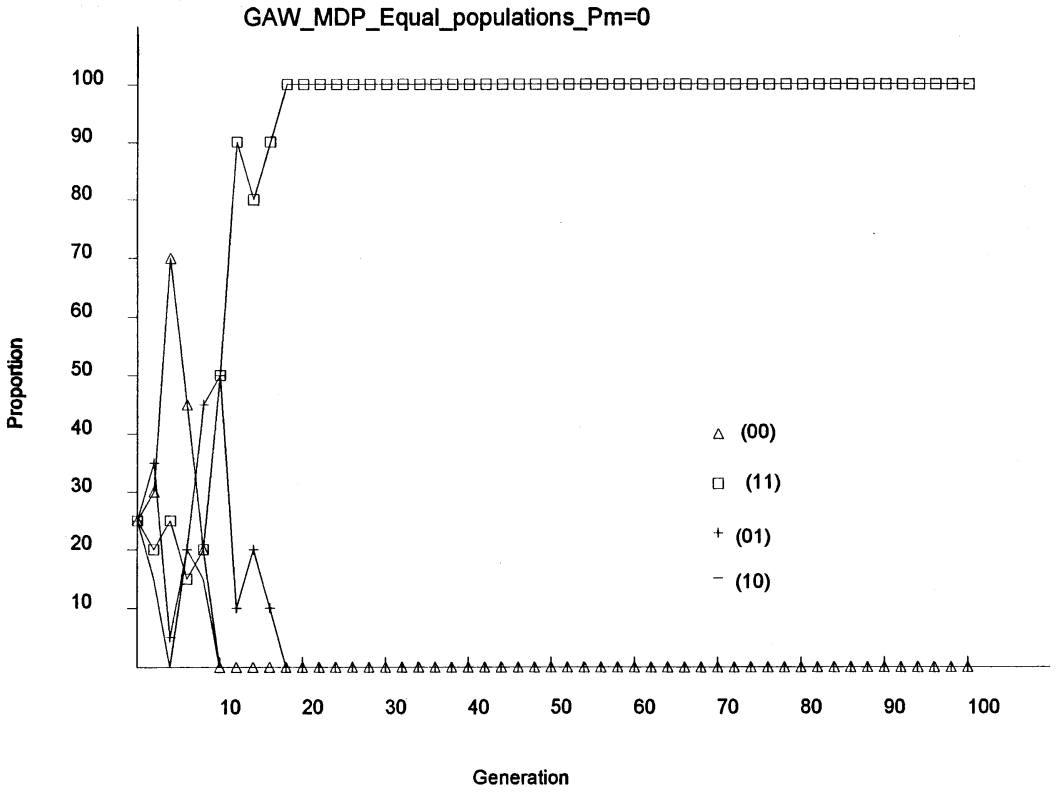


Fig. 10. MDP with equal initial proportions of the four strings, $P_m = 0$ and elitist selection.

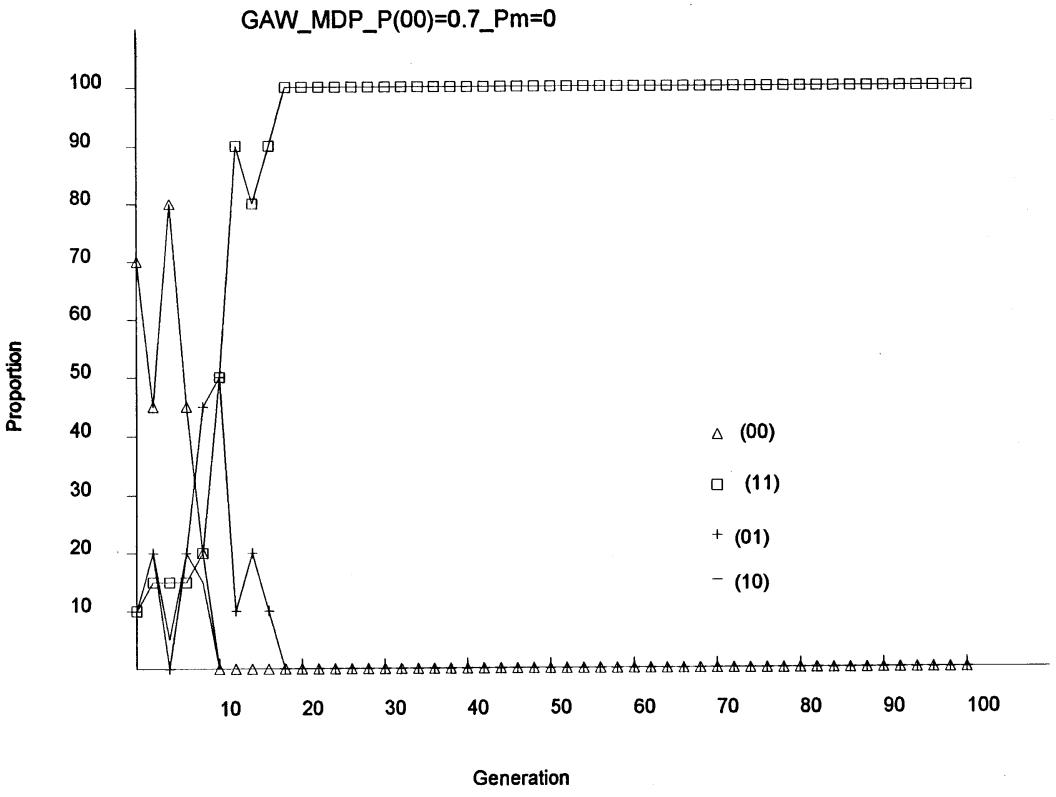


Fig. 11. MDP with unequal initial proportions of the four strings, $P_m = 0$ and elitist selection.

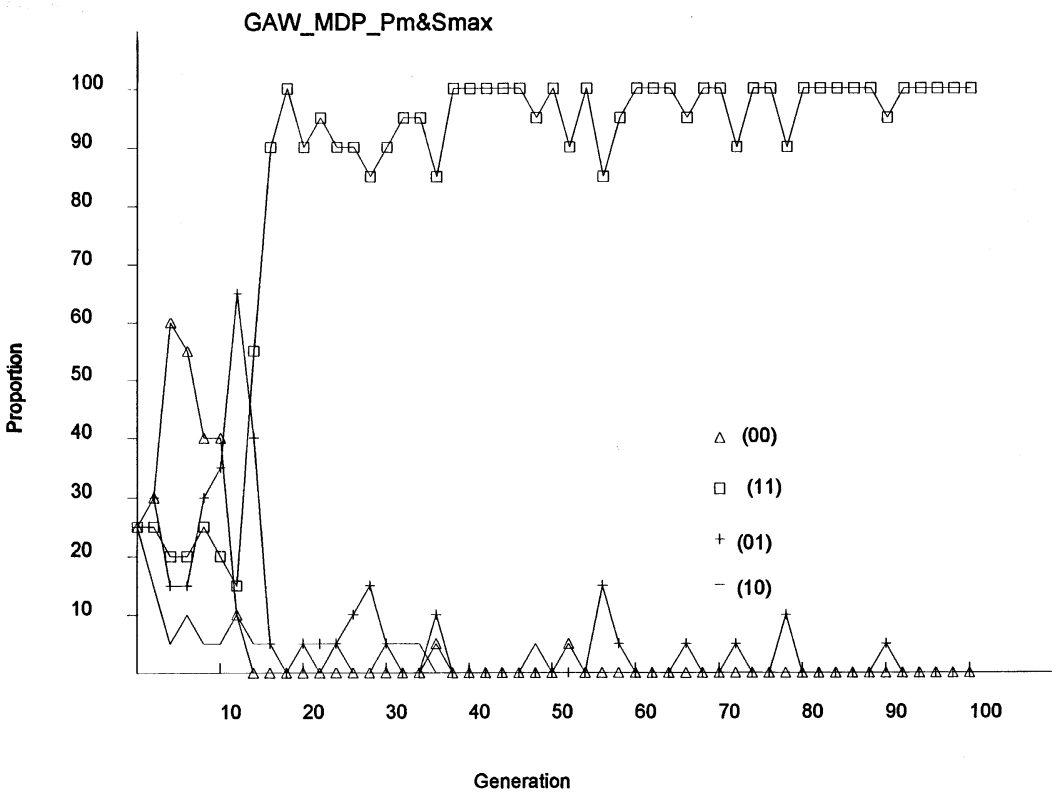


Fig. 12. MDP with equal initial proportions of the four strings, elitist selection and GAW.

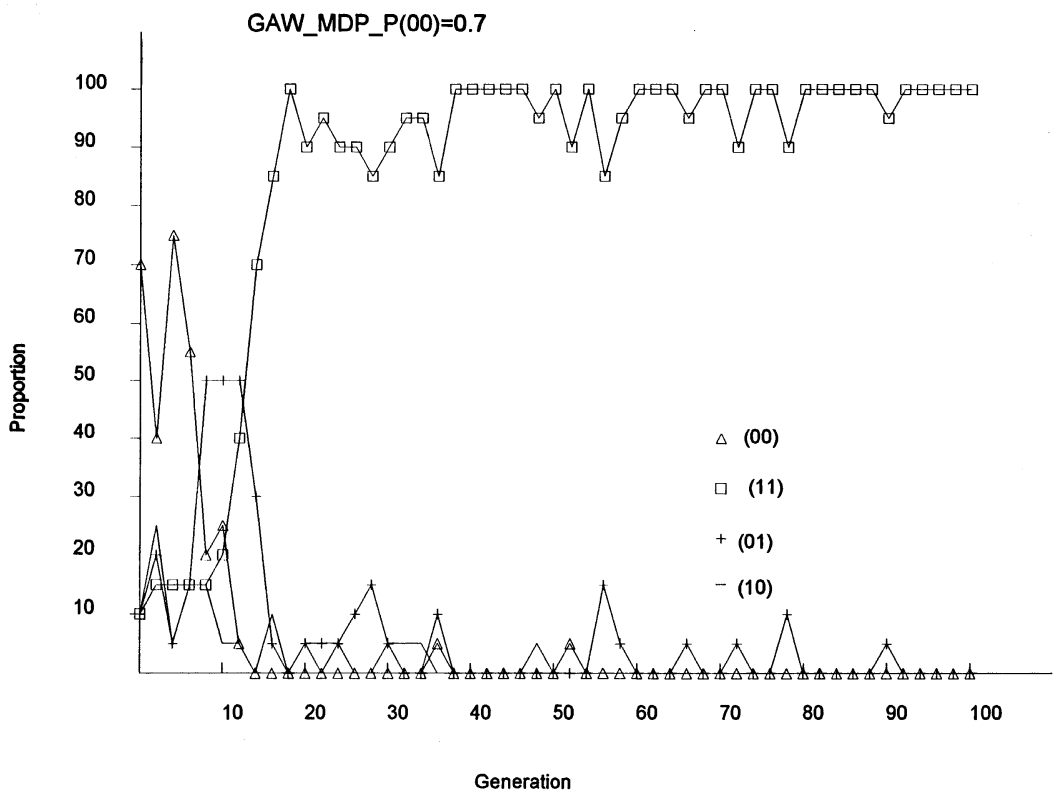


Fig. 13. MDP with unequal initial proportions of the four strings, elitist selection and GAW.

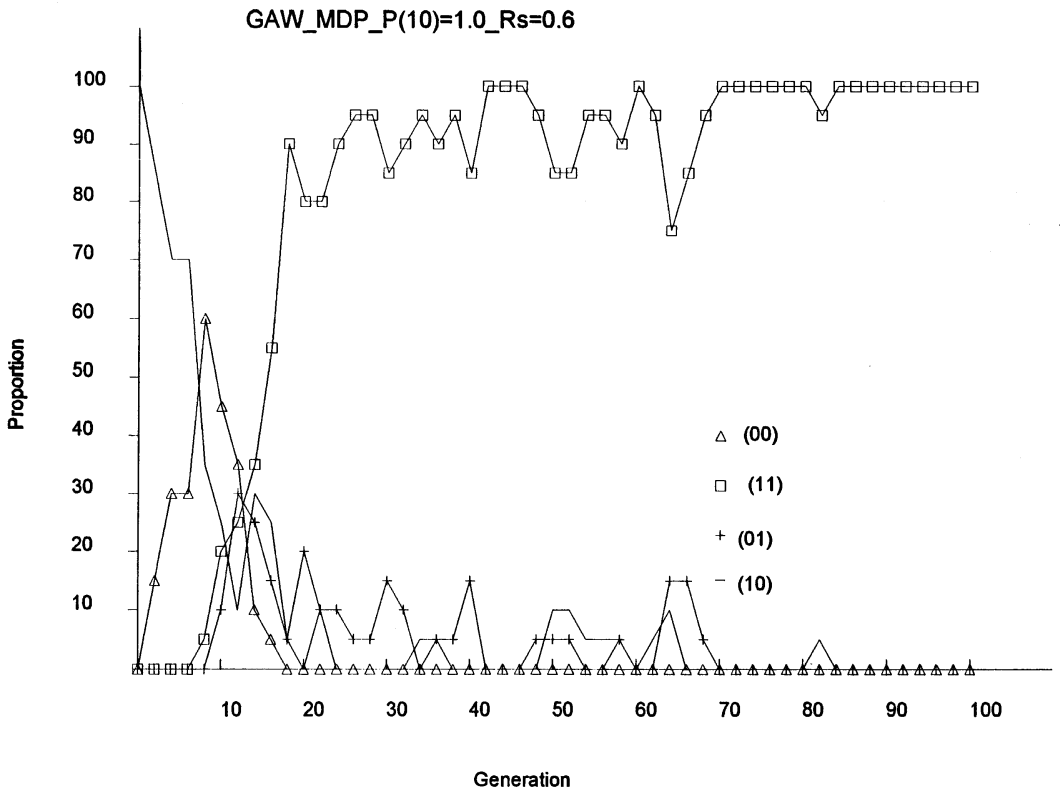


Fig. 14. MDP with initial population of string (10) only, elitist selection and GAW.

aid the GAW convergence. Figures 7–9 illustrate the impact of the inclusion of the seed number, crossover probability and mutation probability into the GW. Mutation is restrained over the latter third of the runs, according to the arctan function in Fig. 15, with factor 1.

Figure 7 represents equal initial proportions of the four schemata, Fig. 8 shows the initial population with 70% of the second-best schema (00), while Fig. 9 shows the initial population which contains schema (10) only. In all cases GW finds the best solution and the algorithm converges to global optimum. To ensure that the disturbance is not too large in more complex problems, we include the elitist selection operator, $S_{\max}$, by preserving two copies of the best string in each generation. This operator stores useful information and increases the probability that the strings will exchange genetic material with the best string in the population. It also smooths the approach to the best solution (if unique), decreasing the variance. Figures 10 and 11 illustrate the impact of this elitist selection operator on algorithm convergence, without mutation.

When the initial population contains one type of schema only (other than globally optimal), the elitist selection in itself is not sufficient in the search for a global optimum, so the two forces — disruptive mutation/crossover and smoothing selection — are best combined. We merge the impact of GW with $S_{\max}$ in Figs 12–14. One can see in all three cases that the GW finds the best schema (11) and carries it into a converging population. Remaining variation is due to potentially high rates of disruption, which are random to the end of the runs. The impact of mutation is dampened by application of the arctan function, but it smooths the algorithm only in the last ten generations.

We also tested the GAW on a class of economic problems [16], where fitness of an individual depends on the state of the population. The GAW proved equally effective, both in case of stable/unstable cobweb models [1] and a GA-hard version with large fixed costs [6].

## 7. CONCLUDING REMARKS

We devised a version of genetic algorithm whose performance does not depend on a specific combination of exogenous parameters. While most of the research effort has been focused on

preserving useful information in the population, the basis for our approach is exploration of diversity and its role in producing good (new) solutions. We rely on natural genetics to solve the problem of GA dependence on initial parameter setting. While chromosomes are the carriers of genetic information, nonsense codon ("genetic waste") is viewed here as a source of diversity. Its role in the translation of messages is carried out through a random set of parameters and, as such, it is included in the population to improve the search abilities of the GA. The stochastic nature of genetic waste creates additional variance in the population, which may disrupt algorithm convergence. The addition of a "damping" function diminishes the impact of random mutations in context-dependent problems which require convergence, while elitist selection ensures the preservation of useful information.

In our presentation, the GAW is tested on Goldberg's [10] minimal deceptive problem to illustrate algorithm effectiveness. It conducts an efficient search of complex spaces with no need for trial and error determination of the "right" exogenous parameters to get us to the global optimum. The algorithm proved to be robust in case of a deceptive problem — it converged to the global optimum regardless of the initial state of the population.

Further testing is clearly needed in order to compare the GW interpretation of nonsense codons with non-coding segments used in the GA literature [21, 22]. Another extension of the algorithm should include a larger class of problems in order to address the question of generality of the GW application.

## REFERENCES


1. Arifovic, J., Genetic algorithm learning and the Cobweb model. *Journal of Economic Dynamics and Control*, 1994, **18,** 3–28.
2. Berg, P. and Singer, M., *Dealing With Genes*. University Science Books, Mill Valey, CA, 1992.
3. Booker, L., Improving search in genetic algorithms, In *Genetic Algorithms and Simulated Annealing*, ed. L. D. Davis. Morgan Kauffman, Los Altos, CA, 1987.
4. Davis, L., ed., *Genetic Algorithms and Simulated Annealing*. Pitman, London, 1987.
5. Davis, L., ed., *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
6. Dawid, H., *Adaptive Learning by Genetic Algorithms: Analytical Results and Applications to Economic Models; Lecture Notes in Economics and Mathematical Systems 441*. Springer-Verlag Berlin Heidelberg, 1996.
7. De Jong, K. A., Adaptive system design: a genetic approach. *IEEE Transactions on Systems Man and Cybernetics*, 1980, **10**(9), 566–574.
8. De Jong, K. A., An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, University of Michigan, 1975.
9. Dorsey, R. E. and Mayer, W. J., Genetic algorithms for estimation problems with multiple optima, non-differentiability and other irregular features. *Journal of Business and Economic Statistics*, 1995, **13**(1), 53–66.
10. Goldberg, D. E., Simple genetic algorithm and the minimal, deceptive problem. In *Genetic Algorithm and Simulated Annealing*, ed. L. Davis. Pitman, London, 1987.
11. Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Co., 1989.
12. Grefenstette, J. J., Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 1986, **16**(1), 122–128.
13. Holland, J. H., *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
14. Holland, J. H. and Miller, J. H., Artificial adaptive agents in eonomic theory. *American Economic Review*, 1991, **81**(2), 365–370.
15. Novkovic, S., A genetic algorithm simulation of a transition economy: an application to insider-privatization in Croatia. *Computational Economics*, 1997, **7**(4), 1–23.
16. Novkovic, S. and Sverko, D., Genetic waste and the role of diversity in genetic algorithm simulations. Paper presented at the *Second Workshop on Economics with Heterogeneous Interacting Agents*, Ancona, May 30–31, 1997.
17. Song, Y. H., Wang, G. S., Johns, A. T. and Wang, P. Y., Improved genetic algorithms with fuzzy logic controlled crossover and mutation, UKACC International Conference on Control, September 2–5. *Conference Publication*, 1996, **427,** 140–144.
18. Srinivas, M. and Patnaik, L. M., Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 1994, **24**(4), 656–667.
19. Szekely, M., *From DNA to Protein: The Transfer of Genetic Information*. John Willey and Sons, NY, 1980.
20. Venkatachalam, A. R., An analysis of an embedded crossover scheme on a GA-hard problem. *Computers and Operations Research*, 1995, **22**(1), 149–157.
21. Wu, A. S. and Lindsay, R. K., Empirical studies of the genetic algorithm with non-coding segments. *Evolutionary Computation*, 1995, **3**(2), 121–147.
22. Wu, A. S., Lindsay, R. K. and Smith, M. D., Studies on the effect of non-coding segments on the genetic algorithm. In *Proceedings of the 6th IEEE International Conference on Tools With Artificial Intelligence*, New Orleans, 1994.
23. Wu, Q. H. and Cao, Y. J., Stochastic Optimization of Control Parameters in Genetic Algorithms, 1997, *IEEE*, **5,** 77–80.

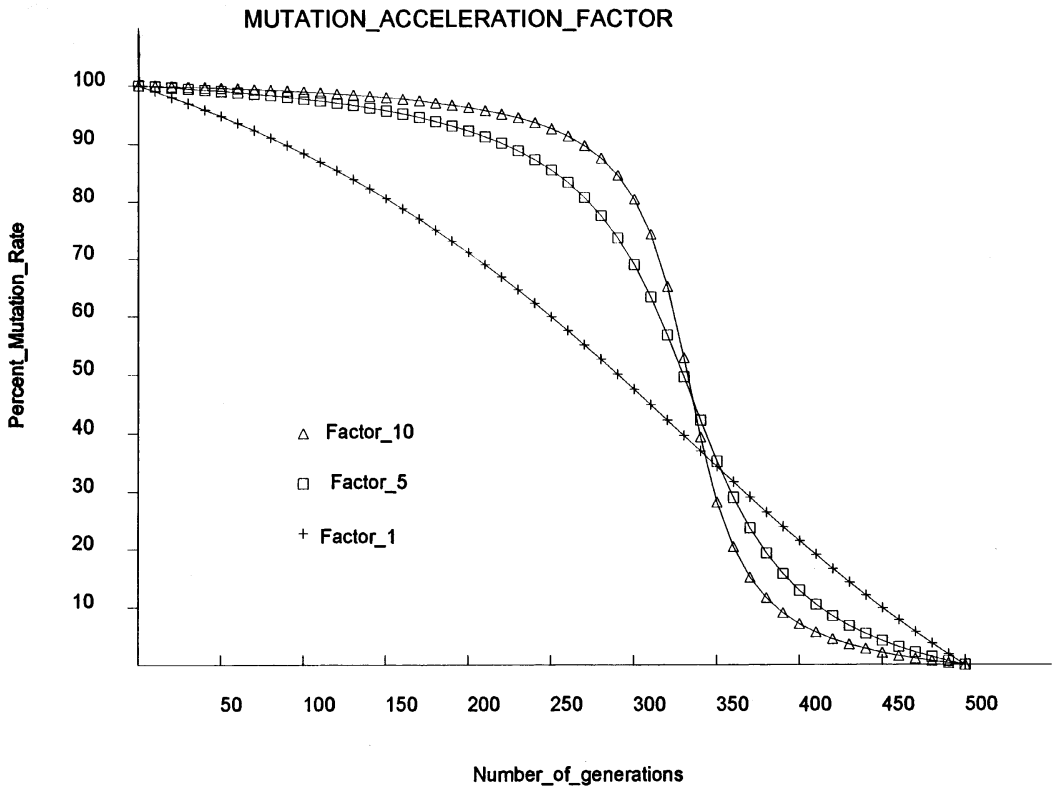


Fig. A1

## APPENDIX A

The original arctan function is defined as:

$$y(x) = \arctan(x), \qquad y(x) = \begin{cases} \frac{\pi}{2} & \text{for } x = -\infty \\ 0 & \text{for } x = 0 \\ -\frac{\pi}{2} & \text{for } x = +\infty \end{cases}$$

After function normalization, we obtain a reduced function, r_arctan, defined over the interval [0, 1] as follows:

$$y'(x) = r\_\arctan(x), \qquad y'(x) = \begin{cases} 1 & \text{for } x = 0 \\ 0 & \text{for } x = \text{max\_gen} \end{cases}$$

The shape of r_arctan function is further distorted in order to move the inflection point to two thirds of the total number of generations (maxgen) and to permit change of the slope according to a given factor. The following manipulations are performed to obtain the final scaling function, $y''(x)$:

$$y_1 = -\arctan\left(-\frac{2}{3}\text{factor}\right) \qquad y_2 = -\arctan\left(-\frac{4}{3}\text{factor}\right) \qquad y'' = \frac{1}{y1 + y2}\left[-\arctan\left(\left(\frac{\text{gen}}{\text{maxgen}} - \frac{2}{3}\text{maxgen}\right)\text{factor}\right)\right] + y2$$

The "factor" can be determined externally. We find values between 1 and 50 to be satisfactory for most problems which require convergence. Figure A1 illustrates the function when the factor takes values 1, 5 and 10.