A Genetic Algorithm With Self-Generated Random Parameters

Sonja Novkovic[*] and Davor Sverko[**]

[*]Department of Economics, Saint Mary's University, Halifax, Canada
[**]ABS Americas, Halifax, Canada

In this paper we present a version of genetic algorithm (GA) where parameters are created by the GA, rather than predetermined by the programmer. Chromosome portions which do not translate into fitness ("genetic residual") are given function to diversify control parameters for the GA, providing random parameter setting along the way, and doing away with fine-tuning of probabilities of crossover and mutation. We test the algorithm on Royal Road functions to examine the difference between our version (GAR) and the simple genetic algorithm (SGA) in the speed of discovering schema and creating building blocks. We also look at the usefulness of other standard improvements, such as non-coding segments, elitist selection and multiple crossover on the evolution of schema.

Keywords: genetic algorithm, Royal Road functions, control parameters, non-coding segments.

1. Introduction and Motivation

Genetic algorithms (GAs, Holland, 1975, Goldberg, 1989) have proved to be effective search mechanisms. They have been adapted for function optimization in a variety of ways (De Jong, 1992), but one of the lingering problems is that the GA performance depends on initial parameter settings. In most applications the parameters (probability of crossover, probability of mutation and population size) are fixed throughout the run. It has been acknowledged in the literature that variable parameter setting is more effective (see Booker, 1987 and Davis, 1991 for example). Numerous studies have been devoted to the disclosure of the relationship between the GA parameters, as well as parameter "optimization" (see Baeck, 1991, De Jong, 1975, 1980, Grefenstette, 1986, Srinivas and Patnaik, 1994, Wu and Cao, 1997, among others). Tuson and Ross, 1998 provide a comprehensive overview of attempts in the GA literature to optimize GA parameters in order to account for their ability to provide more fit individuals in successive generations. Efforts have been made to create adaptive parameters which become a part of the selection process of the strings, but with adaptive parameter settings the parameters are fitness-dependent, posing a problem for systems in which string fitness depends on the state of the population, such as economic systems (Dawid, 1997).

Our motivation was to create a GA which does not require search for an "optimal" set of parameters, or any parameters for that matter, but to have a reliable GA which will do the job when applied to various real problems (Novkovic and Šverko, 1998). Harik and Lobo, 1999 started from a similar search for a "parameter-less" GA, which in this case is not a GA without parameters, but rather one with a set of parameters that can work reasonably well for many problems. They stopped short of including mutation in their algorithm, recognizing that it is a difficult task. Mutation, however, has to be included for the GA to function as intended, for its abilities are severely limited without it.

Our algorithm is an upgraded version of the algorithm in Novkovic and Šverko (1998). We find that GA-generated random parameters are as good as any in function optimization, while they require relatively little in terms of algorithm alterations and computation. They do not depend on fitness and, therefore, are universally applicable. We have demonstrated elsewhere (Novkovic 1999, Šverko 2003, for example) that a random parameter-based GA is exceptionally effective in finding solutions to complex and difficult practical problems.

In what follows we provide a detailed description of the algorithm and test it on Royal Road functions designed to evaluate the schema processing during genetic search (Mitchell, Forrest and Holland, 1991). We use Royal Road functions because we can pinpoint the effects of the algorithm on specific building blocks and compare them with the performance of the simple GA (SGA) in Forrest and Mitchell, 1992 and Mitchell, Holland and Forrest, 1994. Our intention here is to illustrate that random parameter-based genetic algorithm (genetic algorithm with the residual, GAR) is at least as effective as any alternative with parameters which are known to be efficient, while it does not require a search for "good" parameters. Due to sufficient diversity provided by random mutation, it also proved to be effective where the simple GA was deceived (Novkovic and Šverko, 1998).

In our version of the algorithm, the GA itself creates random parameters. The motivation behind it is a plausible interpretation of non coding segments (Novkovic and Šverko, 1997, 1998), whereby untranslated portions of the DNA are viewed as providers of diversity, and thereby a possible source of the improvement of the species. The *introns* or *nonsense codons* create "genetic residual", i.e. the portions of genes whose function is unknown in nature (see Berg and Singer, 1992 for example), but which we interpret to produce variation of parameters for genetic algorithm. Therefore, a part of string representation of individuals in a population is set to provide new random parameters in each generation for each individual and it does not affect the fitness value in any way. A version of non coding segments widely used in the GA literature, on the other hand, assigns to them no function at all (Levenick, 1991, Forrest and Mitchell, 1992, Wu and Lindsay, 1995, Wu, Lindsay and Smith, 1994). These applications result in limited or no improvement of GA performance with fixed building block representation. Wu and Lindsay, 1997, find non coding segments useful when applied with floating building blocks, so we revisit non coding segments application in Section 3.

One last clarification may be in order: the essence of the GAR is that it functions with a set of random control parameters – both mutation and crossover are executed at variable rates from one mating pair of strings to the next in each generation (Section 2). This job may probably be performed by any good random number generator, but we want the GA to be self-sufficient, and serve the purpose of both the random number generator for the parameters, and the customary search tool. It turns out that the GAR so generated is usually more effective than the SGA, while it contains a much simpler structure than more complex versions of the GA with dynamic adaptive operators.

In this paper we wish to: a) illustrate the relative performance of the GAR on "Royal Road" functions (Mitchell, Forrest and Holland, 1991); b) examine the effect of potentially useful alterations such as the non coding segments reported in Mitchell, Forrest and Holland, 1991, Forrest and Mitchell, 1992, Mitchell, Holland and Forrest, 1994, and Wu and Lindsay, 1995; c) evaluate the combination of GR and elite selection on Royal Road functions, given the effectiveness of this combination in other applications, and d) combine the GAR with a form of variable string representation in order to aggregate the positive impact of the floating building block representation on GA search (Wu and Lindsay, 1997) with the positive impact of the GR.

The paper is organized as follows. Section 2 describes the algorithm. In Section 3 we compare the SGA and the GAR versions on the Royal Road problem, with and without the non-coding segments. In order to assess the usefulness of additional algorithm complexity, we then combine the GAR with other GA refinements, some of which were also applied by Mitchell, Holland and Forrest, 1994 in their search of the GA which would outperform hill-climbing: Section 4 deals with elitist selection, while Section 5 examines the effects of variable length representation and a multiple point crossover. Conclusions follow in Section 6.

2. A Genetic Algorithm With the "Genetic Residual" (GAR)

In this section we briefly reproduce the description of the structure of the GAR from Novkovic and Šverko, 1998, with some refinements. In addition to the standard operators – selection, crossover and mutation, the GAR incorporates the "genetic residual" (GR) part of the chromosome. The GR is decoded separately, it does not affect the fitness value, and it provides random parameters for the algorithm. Essentially, one can think of the GAR as two GA-s running in parallel: one providing random probabilities of crossover and mutation (the GR), while the other encoding the problem at hand – here referred to as the 'active' part of the string.

The algorithm is a standard GA, with binary representation of strings, proportional selection, σ-scaling (Tanese, 1989, Forrest and Mitchell, 1992), one point crossover, and mutation. Initial population is created randomly. Each string of length L in a population of n strings contains an "active" part of length l, and the residual of length $(L - l)$, as illustrated by Figure 1.

The residual, GR, provides random parameters, and is subject to crossover and mutation on its own. It is decoded in two parts: alleles $(l+1)$ to (m) as the probability of mutation and $(m + 1)$ to (L) as the crossover probability. The length of the GR depends on the computing abilities at hand, as well as the desired increment for decoding of the parameters[1]. Residuals are selected randomly to the mating pool, independently of the active string and, therefore, independent of the fitness. Figure 2 illustrates the crossover of the GR.

Crossover of the GR occurs with certainty ($p_c = 1$), while for mutation of this part of the string,

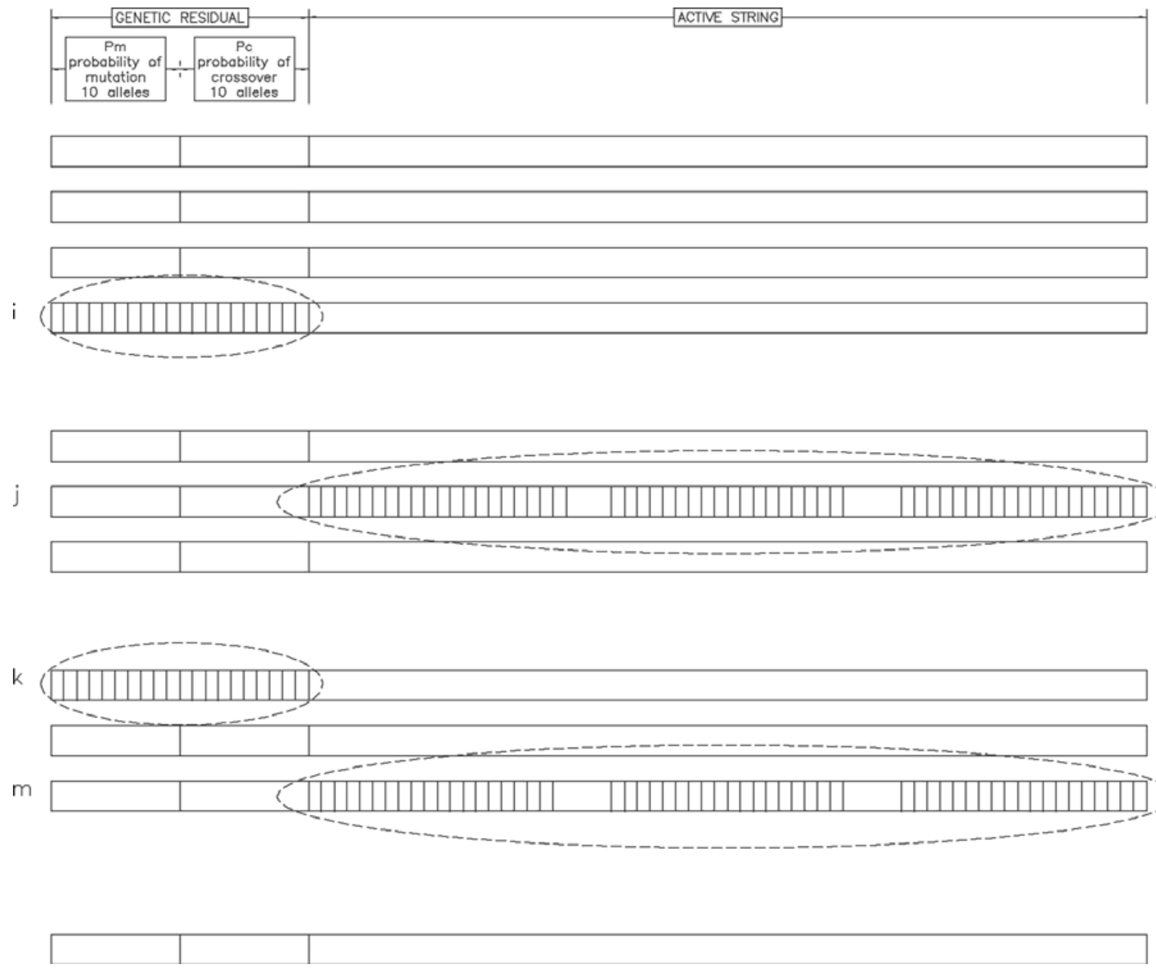
GENETIC RESIDUAL
ACTIVE STRING
Pm probability of mutation 10 alleles
Pc probability of crossover 10 alleles
i
j
k
m

Fig. 1. Initial population of n strings is created at random. Each string of length L consists of the active part of length l which is decoded according to the problem at hand, and the GR part of length $(L - l)$ which provides the parameters and is not problem-specific. Here, the GR includes the probability of crossover and the probability of mutation.

[1] The length of GR used in all applications is 10 alleles, changing values with the increment of $1/1024$. Fifteen alleles would change the values by $1/32768$ etc. We tested different lengths of the GR and there was no significant difference in performance between the 10 alleles, or when we extended the chromosomes.

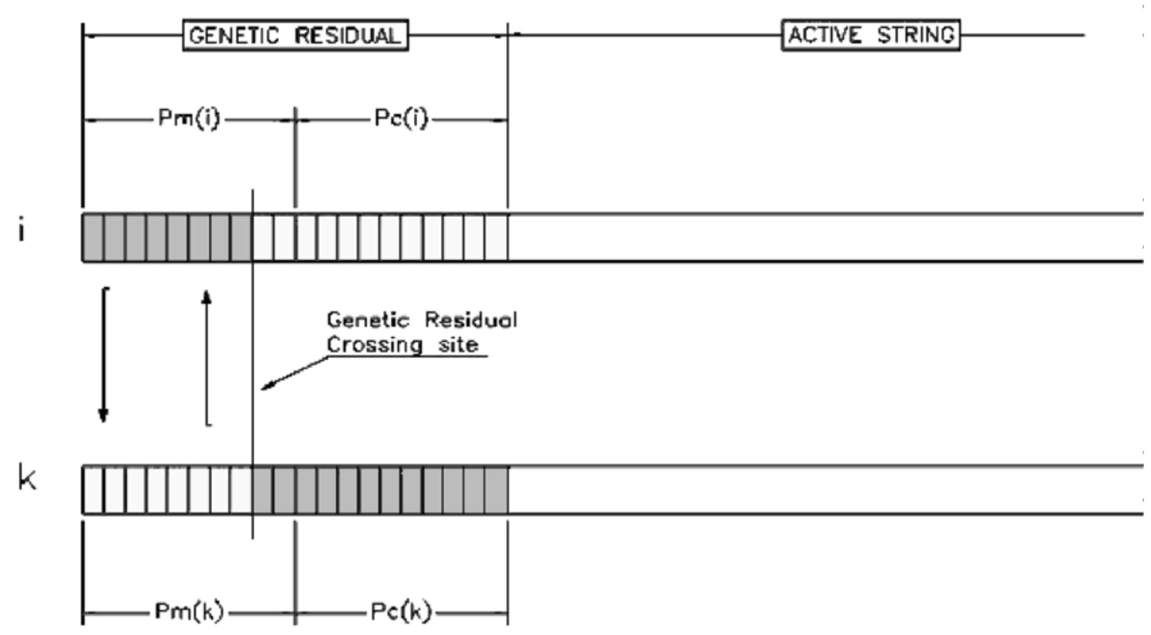

Fig. 2. Crossover is applied to the GR part of the string with certainty. The decoded probability of crossover is then applied to the active string and alleles are exchanged separately.

different probability of mutation is used for each offspring – one from each parent's mutation probability, set in the range [0,1]. This range is deliberately higher than the customary mutation probabilities in the literature, since the GR provides diversity.

For the active part of the string, selection into the mating pool is proportional to the fitness value. Once parents are selected, single crossover is applied – the strings cross with probability provided by the GR of one of the parents, decoded in the range [0,1]. The probability of mutation for each child is used from each mate's GR, in the range [0,0.01].

The enhanced algorithm (GAR) provides increased diversity of the population by varying control parameters in each run, as illustrated in Section 3. This feature may not be intuitive, considering the distribution of random parameters is uniform. Even though statistically equal to SGA with average (fixed) probabilities, the GAR typically shows improvements over the simple version of the algorithm. An important advantage of the GAR over the SGA (and other versions of enhanced GA used in the literature) is that parameter values are automatically provided, doing away with search for the best combination. To that extent, the algorithm is universally applicable.

3. The GAR and the SGA With Non Coding Segments

3.1. The SGA and GAR Compared

As an illustration of the GAR performance, we use the Royal Road functions (Mitchell, Forrest and Holland, 1991 and Forrest and Mitchell, 1992), because they provide a convenient tool for examination of the impact which the potentially disruptive rates of crossover and mutation of the GAR may have on the building blocks, as schemas are explicitly defined in these functions. We examine two functions, R1 and R2 (Figure 1, adopted from Forrest and Mitchell, 1992), defined as

$$R(x) = \sum_{s \in S} c_s \cdot \sigma_s(x)$$

with x representing a bit string, $c_s = \mathit{order}\ (s)$ is the value assigned to the schema s, and $\sigma_s = 1$ if x is an instance of s, and 0 otherwise. In Figure 3, R1 is represented by schemas s_1 through s_8, while R2 includes all 14 schemas.

We first run the generational SGA with one point crossover to repeat the results of previous experiments, and then run the GAR with

s_1 = 11111111**; c_1 = 8

s_2 = ********11111111**; c_2 = 8

s_3 = ****************11111111**; c_3 = 8

s_4 = ************************11111111********************************; c_4 = 8

s_5 = ********************************11111111************************; c_5 = 8

s_6 = **11111111****************; c_6 = 8

s_7 = **11111111********; c_7 = 8

s_8 = **11111111; c_8 = 8

s_9 = 1111111111111111**; c_9 = 16

s_{10}= ****************1111111111111111********************************; c_{10} = 16

s_{11}= ********************************1111111111111111****************; c_{11} = 16

s_{12}= **1111111111111111; c_{12} = 16

s_{13}= 11111111111111111111111111111111********************************; c_{13} = 32

s_{14}= ********************************11111111111111111111111111111111; c_{14} = 32

s_{opt}= 1111111111111111111111111111111111111111111111111111111111111111

Fig. 3. Royal Road functions – an optimal string is broken up into eight building blocks. R1 (x) is computed by summing the coefficients c_1 to c_8, while R2 (x) adds c_1 to c_{14}.

variable probabilities of mutation, as described in Section 2, for comparative performance. The following parameters are employed for SGA: Population size 128, Probability of mutation 0.005, String length 64, Probability of crossover 0.7, Number of runs 200, Max. expected offspring 1.5. The above parameters are used for the simple algorithm, with σ-scaling (Tanese, 1989, Forrest and Mitchell, 1992), restricting maximum expected offspring by any string to 1.5.

When we run the GAR version, σ-scaling remains, and so do the population size and the number of runs. String length now increases by 20 alleles for the GR, used for provision of random parameters, which eliminates the need to provide fixed parameters *ex ante*. Let us note, however, that a larger population size would produce better results for both versions of the algorithm[2], but to be consistent, we apply the parameters used by Forrest and Mitchell, 1992. As stated earlier, our intention is to illustrate that a random parameter-based GAR is at least as effective as any alternative with parameters which are known to be efficient, with the advantage that one does not have to search for those parameters with the GAR.

The results are reported in Table 1 for the SGA, and Table 2 for the GAR; the numbers in brackets represent standard errors. For performance criteria we use the number of generations required until the optimum is found (the number of function evaluations is proportional to the number of generations, so we omit it from the tables). Our results for SGA differ somewhat from those obtained by Forrest and Mitchell, 1992 and Wu and Lindsay, 1995, most likely

	R1-SGA generations	R2-SGA generations
Average	566 (21)	665 (25)
Std. Dev.	292	357
Median	514	616

Table 1. Generational SGA, one point crossover, σ-scaling, based on Forrest and Mitchell 1992.

	R1-GAR generations	R2-GAR generations
Average	440 (15)	469 (18)
Std. Dev.	210	260
Median	405	420

Table 2. GAR with σ-scaling.

[2] We tested the population size 1024, to find that the SGA result improves three-fold and becomes comparable to that of the GAR with equal population.

due to differences in the program structure and randomness of the GA search process, but together with the results in Table 2 they illustrate our point: when the GAR is used, the algorithm performance is on average better than with the SGA with the reported very good parameters, confirming the findings of some previous studies[3].

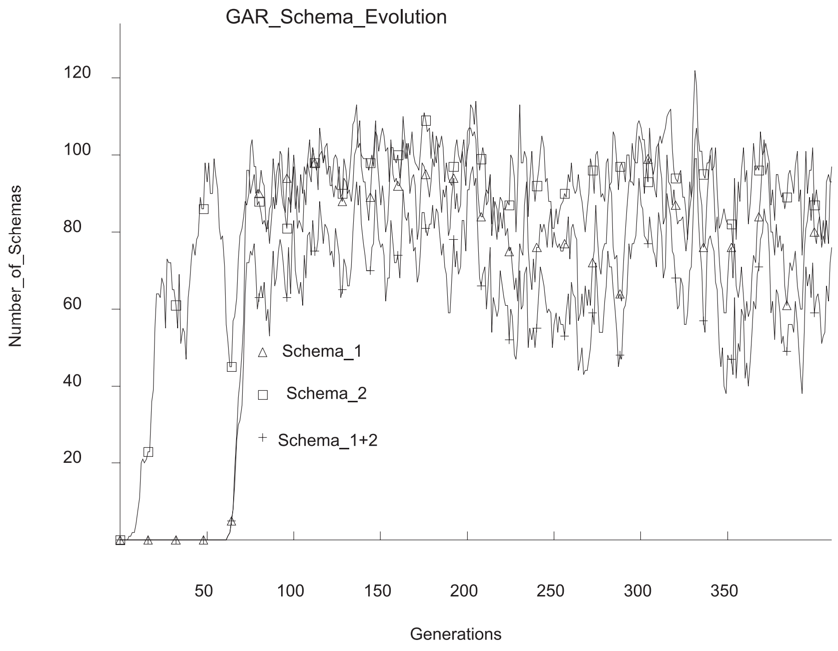

Fig. 4. Evolution of schemas 1, 2, and 9. The intermediate level schema appears soon after both low-order schemas are found. The number of schemas in the population varies much more than with the SGA (Forrest and Mitchell, 1992), indicating lower stability.

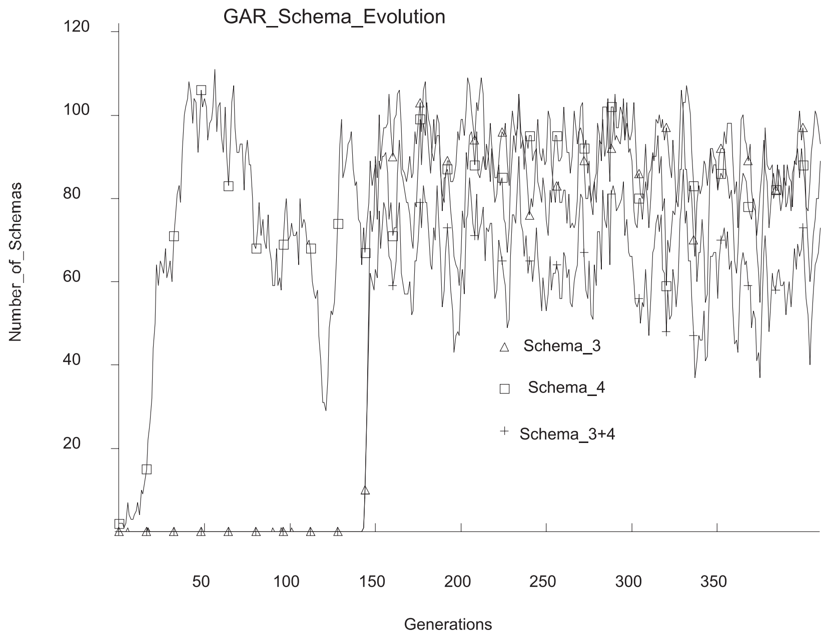

Fig. 5. Evolution of schemas 3, 4, and 10. The intermediate level schema appears soon after schema 3 is present in sufficient numbers (around 140 generations).

[3] In the context of other applications, such as the minimal deceptive problem in Goldberg, 1987, the GAR finds better solutions than alternative GAs (Novkovic and Šverko, 1998).

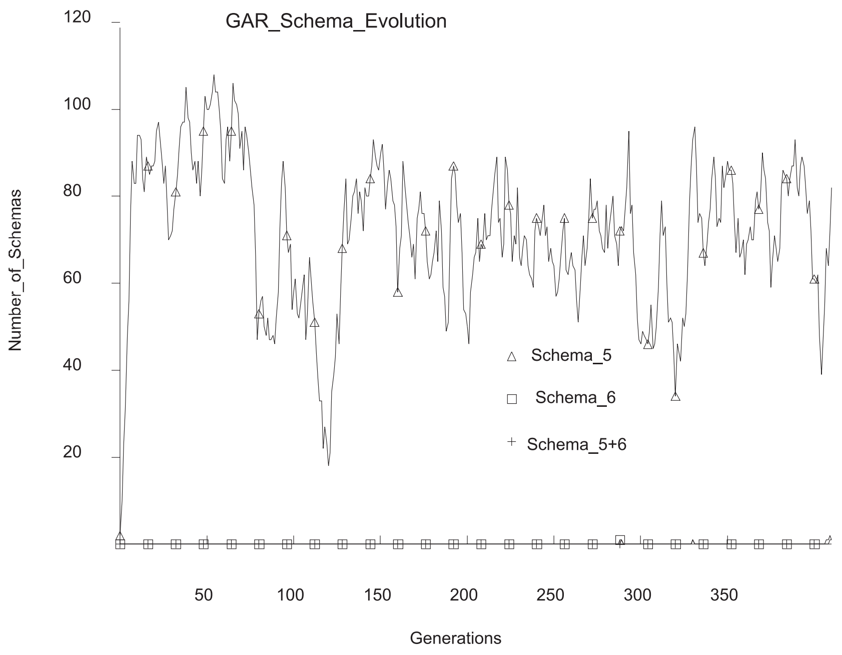

Fig. 6. Evolution of schemas 5, 6, and 11. Schema 6 is found late in the run (288 generation) and lost until rediscovered at the end of the run. This is the cause of prolonged search for the optimum.

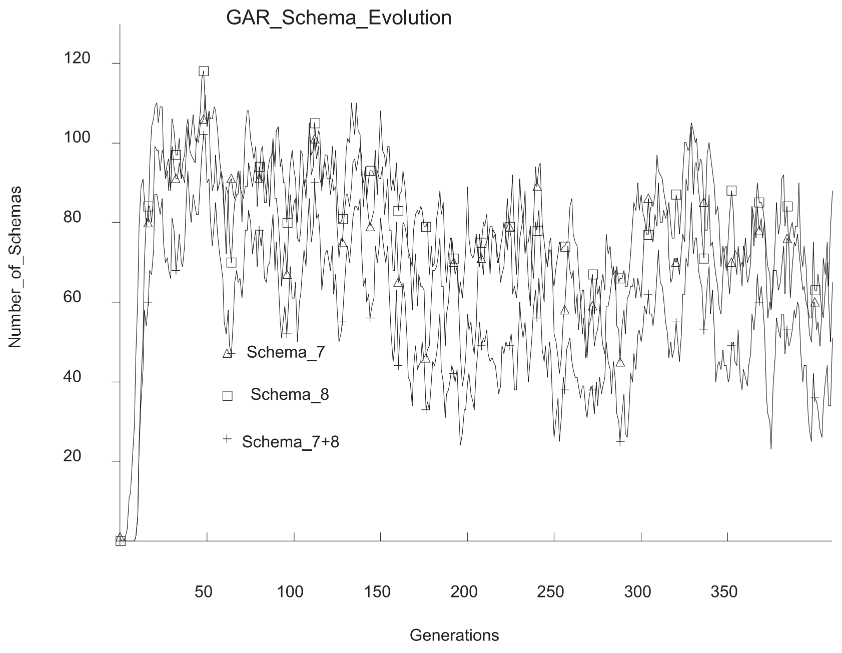

Fig. 7. Evolution of schemas 7, 8, and 12. All three schemas appear very early and maintain presence, ever though with high variability.

An illustration of the evolution of schema for the GAR is given in Figures 4 to 7. The algorithm found the optimum in 410 generations in a single run, which is representative of any other run on average.

The above figures illustrate that the GAR displays more variability in the numbers of schemas it preserves relative to the SGA (Forrest and Mitchell, 1992, Figure 3, p.116). Decreased stability compared with the SGA does not adversely affect its overall searching ability. Like the SGA, the search time of the GAR was prolonged by its inability to find one low-level schema. The time to find intermediate level schemas is typically very short once the low-order schemas are present. We conclude that more variability brought about by the GAR structure does not prevent “hitchhiking” (Forrest and Mitchell, 1992), but it may help find schemas faster due to the potentially larger mutation[4] applied on some strings.

3.2. Non Coding Segments

Non coding segments are applied next, as in Forrest and Mitchell, 1992, Mitchell, Holland and Forrest, 1994 and Wu and Lindsay 1995. We use them between each schema and of equal length (8 alleles). Forrest and Mitchell report no improvement when non coding segments are used. We reiterate their results in Table 3, while Table 4 reports the results when non coding segments are added to the GAR version of the algorithm, also demonstrating no significant change. Further exploration of the combination of non coding segments and diversity provided by the genetic residual is needed, before any conclusive results can be reported. If the intuition that non coding segments restrain the disruption of the crossover is correct (Forrest and Mitchell, 1992), then the combination of this effect with our potentially fairly disruptive operator (GR) should be more effective than the introns combined with the SGA. Even though the combination of the GAR with non coding segments does not seem to be significantly beneficial with Royal Road functions, one should not *a priori* dismiss it in different problems.

	R1-SGA with non coding segments generations	R2-SGA with non coding segments generations
Average	704 (27)	662 (25)
Std. Dev.	380	350
Median	621	601

Table 3. The SGA with non-coding segments.

	R1-GAR with non coding segments generations	R2-GAR with non coding segments generations
Average	438 (19)	454 (19)
Std. Dev.	270	274
Median	377	378

Table 4. The GAR with non-coding segments.

4. The Elite Selection

Generally speaking, elite selection improves algorithm performance (De Jong, 1975, Goldberg, 1989). Various forms of elite selection have been applied in the literature, most often the one where the string with maximum fitness is given a 100% chance of survival, i.e. it is carried to the next generation in one or more copies. Elite selection is used to avoid losing good solutions to disruptive operators. In problems of different nature (Novkovic and Šverko, 1998) random parameter setting was combined with the elite selection, which improved GA performance, as expected. We want to see how the elite selection affects the GAR here, given that in Novkovic and Šverko, 1998 it proved to smooth the approach to the optimum and decrease diversity of the population, offsetting added population variance caused by mutation. With Royal Road functions, low-order schemas are known *ex ante*, and fitness measure depends on their appearance in the string. Elite selection which preserves the string with maximum fitness to date does not prove exceptionally effective on these functions, as it does not prevent the disappearance of low-order schemas from the population, even though it improves the result somewhat. See Table 5.

	R1-GAR with elite selection generations	R2-GAR with elite selection generations
Average	339 (12)	354 (13)
Std. Dev.	172	180
Median	307	316

Table 5. GAR with elite selection, preserving the string with maximum fitness for the next generation.

[4] Even though mutation may be the same on average, with the GAR some strings will be exposed to high mutation, while others to low, rather than all to an equal (average) rate, thereby producing different mating pairs in consecutive generations. For example, two strings, one with $p_{mut} = 0$, and the other with $p_{mut} = 1$, will not produce the same mates as two strings with $p_{mut} = 1/2$.

While two strings may have equal (maximum) fitness, they may contain different low-order schemas, one of which is scarcely present in the population and as such is more valuable for formation of high order schemas. Mitchell, Forrest and Holland, 1991 report that the waiting time for intermediate-level schemas to appear in the population is prolonged by loss of lower-level schemas. Assigning flat fitness value to s_1–s_8 will not prevent loss of relatively scarce low-order schemas in the population. The problem perseveres with the GAR, as illustrated in Section 3. Yet, one can make a case that the approximate 20% improvement in performance is worth applying the elite selection[5]. Still, a more appropriate form of elite selection for Royal Road functions would preserve a copy of each schema as it appears, but this kind of elite selection cannot be used in general, as we typically do not have prior knowledge about the placement of the schema. Mitchell, Holland and Forrest, 1994 create the "idealized genetic algorithm", IDA, making use of a similar elitist selection, with the aim to construct a type of GA which will outperform hill-climbing algorithms. It is no surprise then that a GA with this kind of string preservation does well. We added this feature to both the GAR and the SGA, to conclude that this "idealized" variant benefits the GAR more. Tables 6 and 7 illustrate.

	R1-SGA with idealized elite selection generations	R2-SGA with idealized elite selection generations
Average	457 (19)	525 (23)
Std. Dev.	279	328
Median	402	443

Table 6. SGA with the "idealized" elite selection which preserves each low-order schema once it appears in the population.

	R1-GAR with idealized elite selection generations	R2-GAR with idealized elite selection generations
Average	139 (8)	194 (8)
Std. Dev.	109	115
Median	97	163

Table 7. GAR with the "idealized" elite selection, preserving each low-order schema once it appears in the population.

An observation can be made that the elite selection adds efficiency to GA, but fixed control parameters of the SGA, which were extremely good for the original version, are no longer appropriate. The parameters used in Forrest and Mitchell, 1992 were, arguably, optimal for a given population size, but with addition of the elitist selection the problem changed and another set of "optimal" parameters is required to improve the algorithm performance. This is exactly what can be avoided with the use of random parameters, as in the GAR.

The form of elite selection presented above was motivated by the loss of low-order schemas from the population. Although unusable in general, its inclusion here improves the chances that the algorithm will capitalize on the presence of low-order schemas in the population. Intermediate level schemas may, however, still disappear and defer finding the optimum. Of course, one can combine different types of elitist selection with the GAR. With Royal Road functions, fitness is assigned to parts of the string, and we use that information. In practice, different fitness assignment will be relevant, and one should use whatever information is available to preserve the most valuable individuals in future generations. In general, if it does not help, elite selection with preservation of strings with maximum fitness does not hinder the performance.

5. Variable Length Representation

Unless elite selection (Section 4) is used, GA performance is impeded by the loss of low level schemas, even after they initially appear in the population. We observed that most often only one low level schema is missing for a prolonged time, extending the time required to find the best solution. When elite selection is applied, intermediate level schemas may still disappear. This motivated us to consider variable building block representation (Wu and Lindsay, 1997). Our version of floating representation is less computationally demanding than in Wu and Lindsay, but it suits well the Royal Road function

[5] We obtain an improvement of similar magnitude for SGA (22% for R1 and 14% for R2).

representation. We add one tail segment to the string, essentially creating a ring representation connecting the string head to tail. The algorithm checks for fitness of eight 8-tuples, closing the circle and sliding down one allele to repeat the process. This one-bit slide proved to be more efficient (by about 30%) than the 8-tuple (schema) slide, so this is what we use here.

We first look at a zero-length tail segment, i.e. we close the original string (64 alleles) in a circle, and we witness a change from the original mean of 469 generations for R2 (Table 2) down to 369 generations. Improvement could be expected, since more information is contained in the variable representation of building blocks, even this simple – the GA explores the overlapping bits seven more times than before. Tables 8 and 9 illustrate results for the GAR with variable representation when a tail with 8 bits is added vs. when 64 bits are added to the original 64-bit string.

While extending the genome length by eight bits improves the average performance, longer string representation does not benefit it as much. The reason is that we just transform fixed building block representation, since the low order schemas still have to be together in a block for the best performance. As the string length increases, the role of crossover operator decreases, as it becomes more difficult to obtain a more fit combination of schemas from different mates when their building blocks are potentially far apart. We therefore conjecture that multiple point crossover is necessary when longer strings are used (Spears and De Jong, 1991, Schaffer and Eshelman, 1991). We first isolate the effect of a multiple crossover on a 64-bit string. While more than one crossover point increases GA efficiency, there is little difference in results if a fixed number of crossing sites are selected, or if each mating pair is exposed to randomized selection of the number of sites, when a different number of crossing points (between 1 and the number of 8-tuples in the string) is selected for each pair of mates. The first three rows in Tables 10 and 11 illustrate this point for R1 and R2, respectively.

	R1- 72 bit string generations	R2- 72 bit string generations
Average	139 (5)	199 (8)
Std. Dev.	72	117
Median	122	161

Table 8. GAR with variable building block representation. Eight alleles are added to the string in a ring representation.

	R1- 128 bit string generations	R2- 128 bit string generations
Average	269 (11)	402 (16)
Std. Dev.	151	228
Median	229	360

Table 9. GAR with variable building block representation: 64 alleles are added to the string in a ring representation.

R1				
crossing sites	2	4	8	Random
No NCS				
Average	310 (12)	269 (14)	283 (11)	317 (12)
Std.dev	183	198	162	179
Median	275	210	245	262
With NCS				
Average	308 (13)	330 (15)	398 (16)	381 (16)
Std.dev	184	223	229	235
Median	259	262	333	339

Table 10. A multiple point crossover on R1. Mean, standard deviation and median for GAR without non coding segments – NCS (first 3 rows, 64 bits) and with non coding segments (last 3 rows, 128 bits). Number of crossing sites 2, 4, 8, or randomly selected.

R2				
crossing sites	2	4	8	Random
No NCS				
Average	341 (14)	335 (14)	328 (12)	350 (13)
Std.dev	200	204	181	196
Median	296	271	285	300
With NCS				
Average	334 (15)	338 (14)	449 (19)	414 (18)
Std.dev	223	200	275	259
Median	287	281	376	354

Table 11. A multiple point crossover on R2. Mean, standard deviation and median for GAR without non coding segments (first 3 rows, 64 bits) and with non coding segments (last 3 rows, 128 bits). Number of crossing sites 2, 4, 8, or randomly selected.

R1		
crossing sites	1	Random
No NCS		
Avg	1170 (135)	187 (24)
Std. dev	958	172
Median	904	151
128- NCS		
Avg		208 (20)
Std. dev		146
Median		194

Table 12. The impact of a multiple point crossover on R1 with string length 1024 bits (the first 3 rows). Addition of non coding segments (the last 3 rows) doubles the string length to 2048 bits.

The bottom three rows of Tables 10 and 11 show the results of implementation of multiple crossing sites on strings with non coding segments (total genome length is 128 alleles). Addition of non coding segments and a large number of crossover points (8 or random) is less effective than a smaller number of crossing sites (2 and 4). When string length increases due to addition of alleles which can translate into fitness, large number of crossing sites becomes the most effective.

Tables 12 and 13 illustrate. The first three rows of Table 12 show the mean, standard deviation and the median for one crossing site and for a random number of crossing sites on R1 (1024 bit string length). When NCS are added, the string length doubles, but the GA is equally efficient as with 1024 bits and a multiple crossover.

R2				
crossing sites	1	16	128	Random
No NCS				
Avg	858 (87)	248 (19)	193 (10)	183 (14)
St.dev	620	139	76	99
Median	738	219	175	153
128- NCS				
Avg				305 (29)
St dev				206
Median.				258

Table 13. The impact of a multiple point crossover on R2 with string length 1024 bits (the first 3 rows). Addition of non coding segments (the last 3 rows) doubles the string length to 2048 bits.

Table 13 shows similar results for R2. Large string representation without non coding segments is extremely efficient when combined with a multiple point crossover. The addition of non coding segments to a long string does not significantly hamper the results.

6. Concluding Remarks

In this paper we present the GAR version of a genetic algorithm, where non translated portions of the DNA provide random control parameters, and we apply it to Royal Road functions in order to capture the effect of random parameters on schema processing. The GAR varies the probabilities of mutation and crossover, rather than use fixed parameters, and rather than invest in search for optimal parameter setting, which is costly and ambiguous. The GR (genetic residual) part of the genetic makeup ensures that the performance of the GA does not depend entirely on the programmer's *ex ante* choice of control parameters (probabilities of crossover and mutation, in particular). Due to potentially high mutation, GAR proves most efficient when combined with elitist selection.

We also investigate the impact of non coding segments and variable string representation on the algorithm to conclude that the former are not exceptionally effective with Royal Road functions. Variable string representation, on the other hand, has a positive effect on algorithm performance, particularly if shorter strings are used. When long strings are created, the multiple point crossover improves the speed of search. Contrary to findings by Wu and Lindsay, 1997, inclusion of non coding segments in variable string representation has the same effect as with fixed representation, but this may be a result of our version of the variable string length.

It is clear that the population diversity brought about by varying of the control parameters throughout the runs is likely to improve the GA performance. But, more importantly, there is no need to conduct search for successful parameter setting prior to GA application. Combined with some exploitation-inducing operator, such as elite selection, the GAR produces excellent results and can be safely used for efficient search. One can combine other helpful alterations to increase the speed of search of the algorithm. Further research should illuminate the most successful combinations, as well as their shortcomings.

References

[1] BAECK, T. "Optimal Mutation Rates in Genetic Search" in Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA (1991).

[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*, Davis, L. D. (Ed.), Morgan Kauffman, Los Altos, California (1987).

[4] DAVIS, L. (ED.) *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York (1991).

[5] 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 (1997).

[6] DE JONG, K. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems* Ph.D. dissertation, University of Michigan (1975).

[7] DE JONG, K. "Adaptive System Design: A Genetic Approach" *IEEE Transactions on Systems, Man and Cybernetics*, 10, 9:566–574 (1980).

[8] DE JONG, K. "Genetic Algorithms are NOT Function Optimizers" in Whitley, D. (editor) *Foundations of Genetic Algorithms 2*, Morgan Kaufmann Publishers, pp. 6–18 (1992).

[9] FORREST, S. AND MITCHELL, M. "Relative Building Block Fitness and the Building Block Hypothesis" in Whitley, D. (editor) *Foundations of Genetic Algorithms 2*, Morgan Kaufmann Publishers, pp. 109–126 (1992).

[10] GOLDBERG, D. E. "Simple Genetic Algorithm and the Minimal Deceptive Problem" in Davis, L., editor, *Genetic Algorithm and Simulated Annealing*, 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*, 16, 1:122–128 (1986).

[13] HARIK G. R. AND LOBO F. G. "A Parameter-less Genetic Algorithm" in Banzhaf, W., Daida, Eiben, M. Garzon, V. Honavar, M. Jakiela, R. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 258–267 San Francisco, CA: Morgan Kaufmann (1999).

[14] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan Press (1975).

[15] LEVENICK, J. "Inserting Introns Improves Genetic Algorithm Success Rate: Taking a Cue From Biology", in Belew, R. and Booker L. (editors) *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 123–127 (1991).

[16] MITCHELL, M., FORREST, S. AND HOLLAND, J. "The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance" in *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press, Cambridge, MA (1991).

[17] MITCHELL M., HOLLAND, J. AND FORREST, S. "When Will a Genetic Algorithm Outperform Hill Climbing?" in Cowan, J. D., Tesauro, G. and Alspector, J., editors, *Advances in Neural Information Processing Systems*, 6, Morgan Kaufman, San Mateo CA (1994).

[18] NOVKOVIC, S. "Does Ownership Matter in Evolving Economies?", presented at the 10th Conference of the International Association for the Economics of Participation (IAFEP), Trento, Italy, July (1999).

[19] NOVKOVIC, S. AND ŠVERKO, D. "'Genetic Waste' and the Role of Diversity in Genetic Algorithm Simulations", *Proceedings of the Second Workshop on Economics with Heterogeneous Interacting Agents*, Ancona, May, pp. 30–31 (1997).

[20] NOVKOVIC, S. AND ŠVERKO, D. "The Minimal Deceptive Problem Revisited: The Role of 'Genetic waste"', *Computers and Operations Research*, 25, 11:895–911 (1998).

[21] SCHAFFER, D., CARUANA, R. A., ESHELMAN, L. AND DAS, R. "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization" in *Proceedings of the 3rd International Conference on Genetic Algorithms*, Schaffer, D., editor, Morgan Kaufmann, Los Altos, CA (1989).

[22] SCHAFFER, D. AND ESHELMAN, L. "On Crossover as an Evolutionarily Viable Strategy", in Belew, R. and Booker L. (editors) *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 61–68 (1991).

[23] 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 427:140–144 (1996).

[24] SPEARS, W. AND DE JONG, K. "An Analysis of Multi Point Crossover" in Rawlins, G. (editor) *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 301–315 (1991).

[25] SRINIVAS, M. AND PATNAIK, L. M. "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms", *IEEE Transactions on Systems*, Man and Cybernetics, 24, 4:656–667 (1994).

[26] ŠVERKO, D. "Shaft Alignment Optimization with Genetic Algorithms", *SNAME Propellers/Shafting Symposium 2003*, Virginia Beach, September, pp. 17–18 (2003).

[27] TANESE, R. *Distributed Genetic Algorithms for Function Optimization*, Doctoral Dissertation, University of Michigan, Ann Arbor, MI (1989).

[28] TUSON, A. AND ROSS, P. "Adapting Operator Settings in Genetic Algorithms", *Evolutionary Computation*, 6, 2:161–184 (1998).

[29] WU, A. S. AND LINDSAY, R. K. "Empirical Studies of the Genetic Algorithm With Non-coding Segments", *Evolutionary Computation*, 3, 2:121–147 (1995).

[30] WU, A. S. AND LINDSAY, R. K. "A Comparison of the Fixed and Floating Building Block Representation in the Genetic Algorithm", *Evolutionary Computation*, 4, 2:169–193 (1997).

[31] 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).

[32] WU, Q. H. AND CAO, Y. J. "Stochastic Optimization of Control Parameters in Genetic Algorithms", *IEEE*, 5:77–80 (1997).

Received: February, 2003
Revised: June, 2003
Accepted: July, 2003

Contact address:
Sonja Novkovic
Department of Economics
Saint Mary's University
923 Robie Str.
Halifax, NS B3H 3C3
Canada
Phone: +1 902 420.5607
Fax: +1 902 420.5129
e-mail: s.novkovic@smu.ca

Davor Sverko
ABS Americas
1470-99 Wise Road
Dartmouth, NS
Canada
Phone: +1 902 422.1649
e-mail: dsverko@eagle.org

SONJA NOVKOVIC obtained her Ph.D. in economics at McGill University in Montreal. She is an Associate Professor of economics at Saint Mary's University in Halifax, Canada. Her research interests include genetic algorithms, bounded rationality and economic modeling with adaptive agents.

DAVOR SVERKO has M.A.Sc. degree from Concordia University in Montreal. He is a senior engineer at the ABS Americas in Houston. His research interests include genetic algorithms and applications to mechanical systems optimization.