Towards a Genetic Algorithm for Function Optimization

Sonja Novkovic[‡] and Davor Šverko

Abstract:

This article analyses a version of genetic algorithm (GA, Holland 1975) designed for function optimization, which is simple and reliable for most applications. The novelty in current approach is random provision of parameters, created by the GA. Chromosome portions which do not translate into fitness 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 our algorithm on Royal Road functions to examine the difference between our version (GAW) and the simple GA (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.

Key words: *Genetic algorithm, Royal Road functions*, *optimization, control parameters, non-coding segments*

[‡]Department of Economics

Saint Mary's University

Halifax, N.S.

Canada, B3H 3C3

s.novkovic@stmarys.ca

902-4205607 (tel)

902-4205129 (fax)

[*]ABS Americas

16855 Northchase Drive

Houston, TX 77060

dsverko@eagle.org

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 (see De Jong, 1992), but one of the remaining problems is that the GA performance depends on initial parameter settings. In most applications the parameters[1] are fixed throughout the run. It has been acknowledged that variable parameter setting is more effective (see Booker, 1987 and Davis, 1991 for example). Tuson and Ross (1998) provide an overview of attempts in the GA literature to optimize parameters in order to account for their ability to provide more fit individuals in successive generations. In other words, with adaptive parameter settings the parameters are fitness-dependent[2]. We find, on the other hand, that 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 widely applicable. This point is illustrated in what follows on Royal Road functions (Mitchell, Forrest and Holland, 1991) because we can pinpoint the effects of the algorithm on specific building blocks and thereby compare it with the performance of the simple GA (SGA) in Forrest and Mitchell (1992) and Mitchell, Holland and Forrest (1994). In Novkovic and Sverko (1998) we have illustrated the effectiveness of a previous version of the algorithm on Goldberg's (1987) minimal deceptive problem. Our intention here, then, is not to find an algorithm which outperforms all others in all cases, but to

[1]Probability of crossover, probability of mutation and population size.

[2]Fitness dependency may cause a problem with systems in which string fitness depends on the state of the population (Dawid, 1997).

illustrate that random parameter-based GAW is at least as effective as any alternative with parameters which are known to be efficient, while it does not require search for “good” parameters.

In our version of the algorithm, the GA itself creates random parameters. The motivation for it was our understanding of non coding segments (Novkovic and Šverko (1997,1998)), initiated by an interview with a Swiss geneticist, M.Radman, who views the untranslated portions of the DNA as providers of diversity, and thereby a possible source of improvement of the species. The *introns* or *nonsense codons* create “genetic waste”, i.e. the portions of genes whose function is unknown in nature (see Berg and Singer, 1992 for example), but which we interpreted to produce variation of parameters for a genetic algorithm purpose. Generally speaking, one can think of these non-translated portions as of sources of diversity, i.e. creators of genetic material which cannot be traced as heritage. Therefore, a part of string representation of individuals in a population is set to provide new random parameters in each generation, and it does not affect fitness value in any way. A version of non coding segments widely used in 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). Their applications result in limited or no improvements of GA performance with fixed building block representation. Wu and Lindsay,1997, find these segments useful with floating building blocks. The use of “genetic waste”, as stated before, was motivated by the desire to do away with fine tuning of the control parameters in a genetic algorithm, yet not to optimize the parameters, as most researchers of the problem have attempted to do (Baeck,1991, De Jong,1975, 1980, Grefenstette,1986, Srinivas and Patnaik,1994, Wu and Cao, 1997, among others). Rather, “genetic waste” provides different crossover and mutation probabilities in each run for each set of mating pairs (see Section 2 below). When non translated portions are given function to diversify the parameters[3], they may result in considerable improvements of the GA[4]. While we do not intend to claim by our limited research so far that the GAW version of the algorithm is better than every other GA in all problems (point raised by Wolpert and Macready, 1997), we would like to illustrate that it is usually more effective than the SGA, and much simpler to create than GA with dynamic adaptive operators. These properties, we believe, make GAW a good candidate for an effective optimization tool.

What we set out to do in this presentation is to a) illustrate the performance of "genetic waste" (GW) interpretation of *nonsense codons* 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 combination of GW and elite selection on Royal Road functions, given the effectiveness of this combination in other applications, and d) combine GAW with a form of variable string representation in order to aggregate positive impact of floating building block representation on GA search (reported in Wu and Lindsay,1997) with positive impact of the GW.

The paper is organized as follows. Section 2 describes the algorithm. In Section 3 we

[3]Non coding segments do not affect fitness, by definition. We do not interpret that to necessarily mean that they have no other function. Therefore, when "nonsense codons" are diversity providers, we term them "genetic waste" (GW). When no function is assigned, the term will be "non coding segments".

[4]On top of being simple, the GAW proved so far to be rather robust, particularly in problems where SGA faces difficulties (such as deception, see Novkovic and Sverko 1998).

compare the SGA and the GAW versions on the Royal Road problem laid out in Forrest and Mitchell, 1992, with and without the non-coding segments. Section 4 deals with elitist selection, while Section 5 examines the effects of variable length representation and a multiple point crossover. Some preliminary conclusions follow in Section 6.

2. A genetic algorithm with "genetic waste" (GAW)

In this section we briefly reproduce the description of the structure of GAW from Novkovic and Šverko,1998, with some refinements. 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, not affecting the fitness value, and which provides different random parameters in each generation. The algorithm is a standard GA, with proportional selection,   -scaling, and one point crossover, unless stated otherwise. Each string of length *L* in a population of *n* strings contains an 'active' part of length *l* and the GW part of length *(L-l)*. See Figure 1.

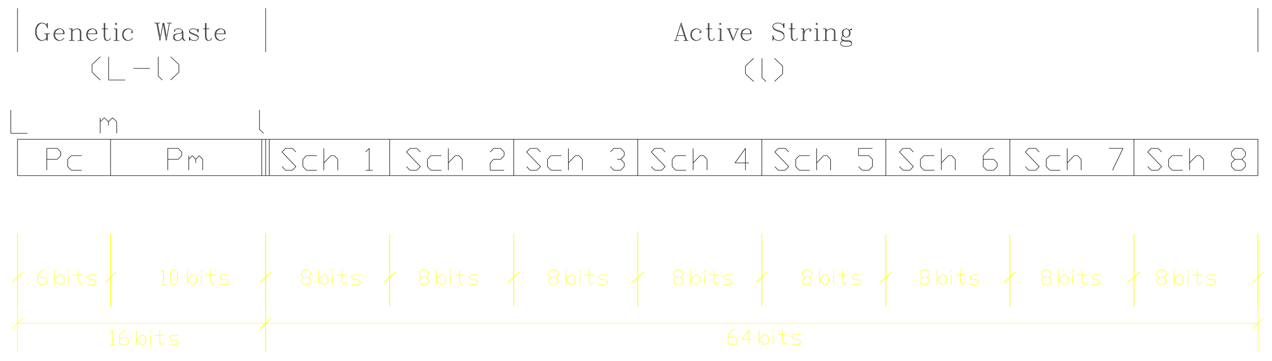

Figure 1: Each string consists of the GW part which provides probabilities of crossover and mutation, and of the active part which translates into fitness. With Royal Road functions the active string is decoded as 8-bit schemas (Section 3)

The GW, which provides random parameters, is subject to crossover and mutation on its own. This part of the string is decoded as probabilities of mutation and crossover, random selection is performed on it (with no relation to fitness of the active part of the string), and obtained parameters are applied to the crossover and mutation of the active string. This way, a whole new population of parameters is created in each generation.

In the initial population the GW part of the string is randomly chosen, together with the active part of the string. It is then 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 each of the parts depends on the computing abilities at hand, as well as the wanted range of values[5] 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, in the same fashion as the active part of the string. The selection procedure of mates for creation of GW is random, i.e. *not* related to 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 for each offspring (one from each of the parents rates, set in the range [0,1] increasing with increment 1/1024).

b. the active part of the string is initially randomly created. The selection of strings for the mating pool is proportional to fitness, and separate for this part of the string. For crossover of the active part of the string, crossover probability of the second mate is applied (provided by the second mate's GW part of the string), while the probability of mutation for each child is used from each

[5]We tested different lengths of the GW and there was no significant difference in performance when we use the length specified below and when we extend the chromosomes.

mate's GW.

Probability of mutation. The SGA version of the algorithm uses the mutation probability (p_m) set at a fixed rate. For stochastic mutation rates applied here, we use alleles *(l+1)* to *(m)* of the GW string part to decode them into p_m for each string in each generation. The number of alleles used in this procedure limit the range of mutation probabilities, but the possibilities are obviously enormous. We typically use 10 alleles, which translates into mutation probabilities in the range [0, 0.02], changing with an increment of 1/1024, but this may be changed as required (see previous footnote).

Probability of crossover is decoded from the GW, alleles *(m+1)* to *(L)*. The p_c is also random, rather than fixed exogenously. In our version of the algorithm, the crossover probability can range from [0,1].

The algorithm so enhanced (GAW) provides increased diversity of the population by varying control parameters in each run, as illustrated in section 3. This feature may not be intuitive, as the distribution of random parameters is uniform. An important advantage of GAW 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.

In the following sections we first compare GAW to SGA. In order to assess the usefulness of additional algorithm complexity, we then combine GAW with other GA refinements, some of which were also applied by Mitchell, Holland and Forrest (1994) in search of the GA which would outperform hill-climbing.

3. GAW and SGA with non coding segments

3.1. SGA and GAW compared

As an illustration of the GAW performance, we use the Royal Road functions (Mitchell, Forrest and Holland (1991) and Forrest and Mitchell (1992)) because they are a convenient tool for examination of the impact that the potentially disruptive rates of crossover and mutation of the GAW may have on the building blocks, as schemas are explicitly defined. We examine two functions, R1 and R2 (Figure 2, adopted from Forrest and Mitchell (FM '92)), defined as

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

with x representing a bit string, $c_s = order\ (s)$ is value assigned to the schema s, and $\quad _{s} = 1$ if x is an instance of s, and 0 otherwise. In Figure 2, R1 is represented by schemas s_1 through s_8, while R2 includes all 14 schemas.

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

Figure 2: 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}.

We run the generational SGA with one point crossover to repeat the results of previous experiments, and then run the GAW with variable probabilities of mutation, as described in Section 2 above, for comparative performance. The following parameters were employed:

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 SGA version, with -scaling (Tanese,1989, FM'92), restricting maximum expected offspring by any string to 1.5.

When we run the GAW version, -scaling remains, and so do the population size and the number of runs. String length now increases by 16 alleles (GW), used for provision of random parameters, and eliminating 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[6], but we apply the parameters used by FM'92 for consistency of the comparison. As stated earlier, our intention here is not to find an algorithm which outperforms all others in all cases, but to illustrate that random parameter-based GAW is at least as effective as any alternative with parameters which are known to be efficient.

The results are reported in Table 1 for the SGA, and Table 2 for the GAW; numbers in brackets represent standard errors. For performance criteria we use number of generations and number of function evaluations required until the optimum is found. Our results for SGA differ

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

somewhat from FM ‘92 and Wu and Lindsay, 1995 (WL ‘95), most likely due to differences in program structure and randomness of the GA search process, but together with results in Table 2 they illustrate our point that when GAW is used the performance is no worse, and likely better than with the SGA with very good parameters, confirming the findings of our previous studies (Novkovic and Šverko[7], 1997, 1998).

	R1-SGA		R2-SGA	
	eval.	gen.	eval.	gen.
Average	71961	566	84582	665
	(2633)	(21)	(3222)	(25)
Std. Dev.	37143	292	45456	357
Median	65429	514	78304	616

Table 1: Generational SGA, one point crossover, -scaling

	R1-GAW		R2-GAW	
	eval.	gen.	eval.	gen.
Average	56011	440	60185	469
	(1899)	(15)	(2360)	(18)
Std. Dev.	26788	210	33373	260
Median	51544	405	53834	420

Table 2: GAW with -scaling

[7] In the context of other applications, GAW finds better solutions than alternative GAs, with no need to look for good control parameters.

An illustration of evolution of schema for GAW is given by Figures 3 to 6. The algorithm found the optimum in 410 generations in a single run, which is representative of any other run on average.

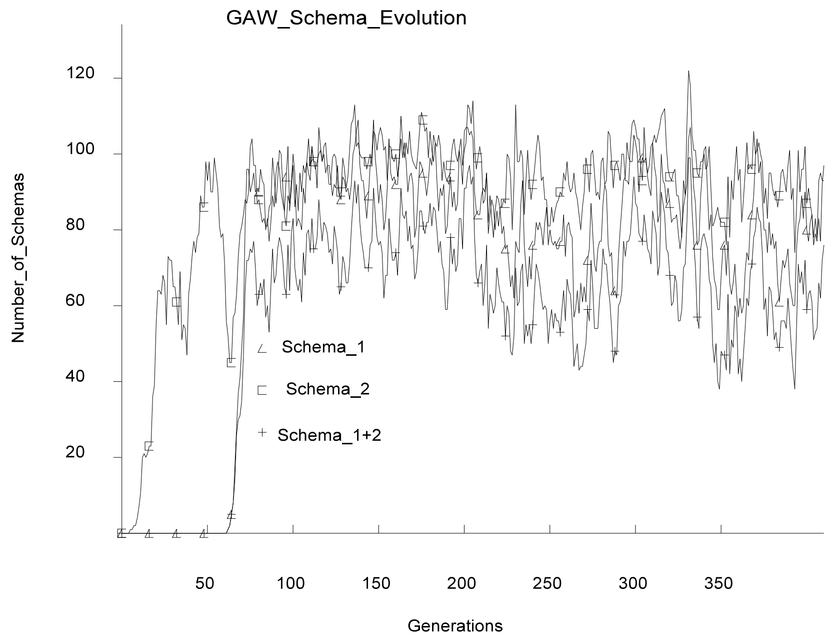

Figure 3: Evolution of schemas 1,2,and 9. The intermediate level schema appears soon after both low-order schemas are found. The number of schemas in the population varies much more than with the SGA (FM '92), indicating less stability.

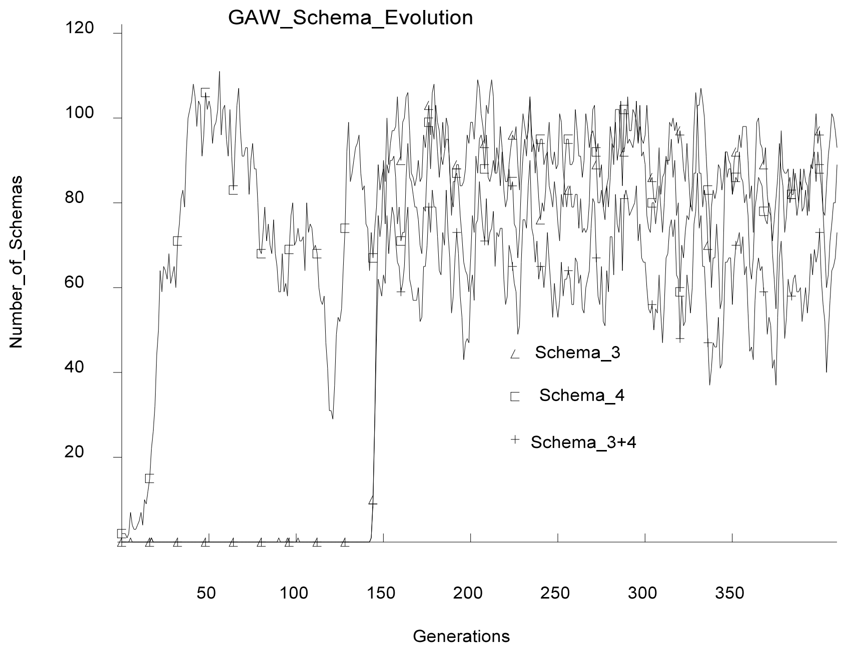

Figure 4: Evolution of schemas 3,4 and 10. The intermediate level schema appears soon after schema 3 is present in sufficient numbers (around 140 generations).

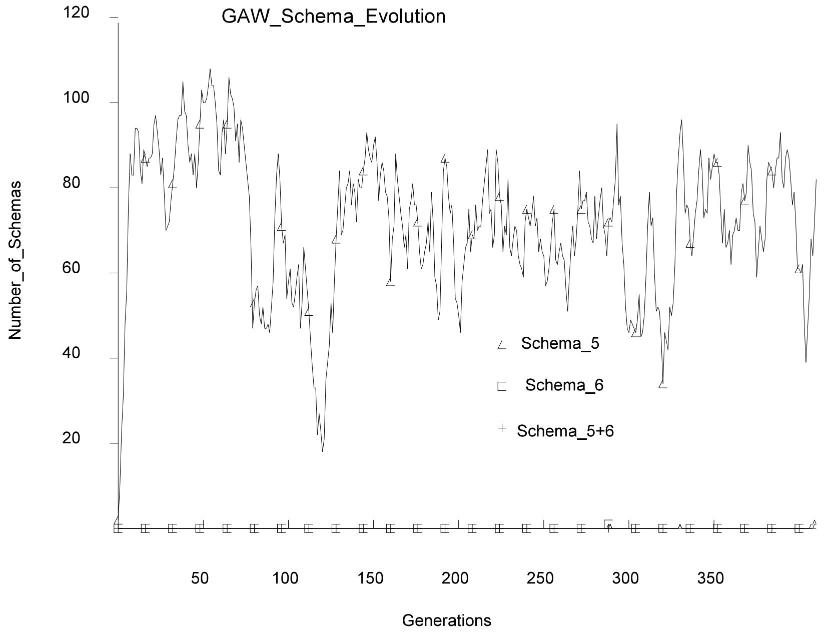

Figure 5: Evolution of schemas 5,6 and 11. Schema 6 is found late in the run (288 generation) and lost until rediscovered at the end of the run. This is the cause of prolonged search for the optimum.

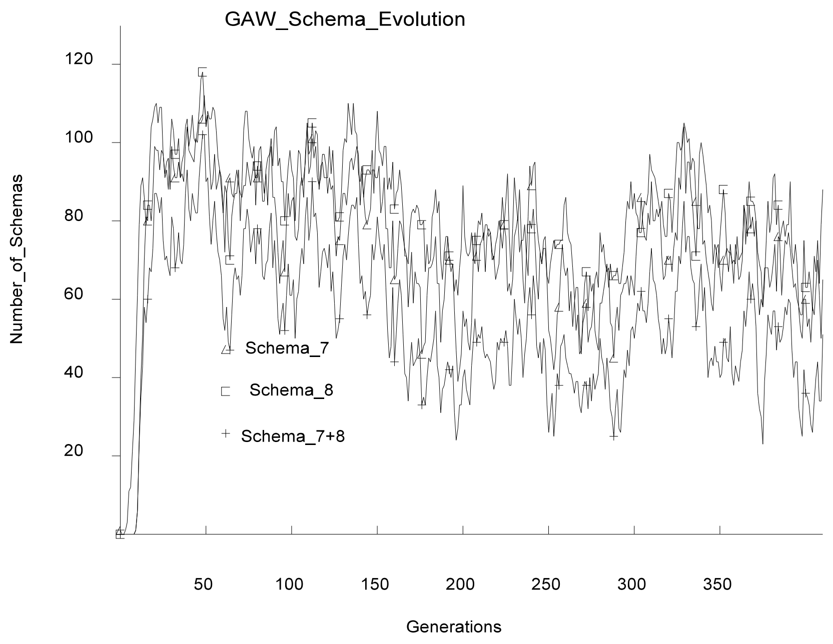

Figure 6: Evolution of schemas 7,8 and 12. All three schemas appear very early and maintain presence, even though with high variability.

The above figures illustrate that GAW displays more variability in the numbers of schemas it preserves relative to the SGA (FM '92, 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 GAW was prolonged by its inability to find one low-level schema. The time to find intermediate level schemas is typically very short once low-order schemas are present. We conclude that more variability brought about by the GAW structure does not prevent "hitchhiking" (FM '92), but it may help find schemas faster due to potentially larger mutation[8] applied on some strings.

[8]Even though mutation may be the same on average, with GAW some strings will be exposed to large mutation, while other to low, rather than all to equal (average) rate, thereby producing different mating pairs in consecutive generations. For example, two strings, one with pmut=0, and the other with pmut=1 will not produce the same mates as two strings with pmut=1/2.

3.2. Non coding segments

Non coding segments are applied next, as in FM '92, Mitchell, Holland and Forrest, 1994 and WL '95.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 confirm their results in Table 3, while Table 4 reports the results when non coding segments are added to the GAW version of the algorithm, also demonstrating no significant change. We may need to explore the combination of non coding segments and diversity provided by GW further, before any conclusive results can be reported. If the intuition that non coding segments restrain the disruption of crossover is correct (FM '92), then the combination of this effect with our potentially fairly disruptive operator (GW) should be more effective than introns combined with the SGA. Even though the combination of GAW 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 NCS		R2-SGA with NCS	
	eval.	gen.	eval.	gen.
Average	90279	704	84852	662
	(3443)	(27)	(3175)	(25)
Std. Dev.	48702	380	44906	350
Median	79618	621	77100	601

Table 3:SGA with non-coding segments

	R1-GAW with NCS		R2-GAW with NCS	
	eval.	gen.	eval.	gen.
Average	56196	438	58299	454
	(2452)	(19)	(2482)	(19)
Std. Dev.	34679	270	35095	274
Median	48320	377	48438	378

Table 4:GAW with non-coding segments

4. 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, the most often probably 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. In problems of different nature we combined the GAW with elite selection, and improved GA performance (see Novkovic and Šverko 1998), as this operator preserves useful information which may be lost due to potentially disruptive nature of random crossover and mutation rates. Others use elite selection for similar results. We want to see how elite selection affects GAW here, given that it proved to smooth the approach to the optimum and decrease diversity of the population, offsetting added population variance caused by mutation in our previous studies.

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-GAW with elite selection		R2-GAW with elite selection	
	Eva.	gen.	eval.	gen.
Average	43143	339	45134	354
	(1551)	(12)	(1621)	(13)
Std. Dev.	21878	172	22864	180
Median	39086	307	40217	316

Table 5:GAW with elite selection, preserving the string with maximum fitness.

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 GAW, as illustrated in Section 3. Yet, one can make a case that the approximate 20% improvement in performance is worth applying the elite selection[9]. Still, a more appropriate form of elite selection 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

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

create the IGA ("idealized genetic algorithm") making use of a similar elitist selection, with 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 GAW and the SGA, to conclude that this "idealized" variant benefits the GAW more. Tables 6 and 7 illustrate.

	R1-SGA with idealized elite selection		R2-SGA with idealized elite selection	
	Eva.	gen.	eval.	gen.
Average	58615	457	67305	525
	(2525)	(19)	(2976)	(23)
Std. Dev.	35718	279	42089	328
Median	51572	402	56747	443

Table 6:SGA with "idealized"elite selection, preserving each low-order schema once it appears in the population

	R1-GAW with idealized elite selection		R2-GAW with idealized elite selection	
	eval.	gen.	eval.	gen.
Average	17887	139	24941	194
	(996)	(8)	(1042)	(8)
Std. Dev.	14080	109	14738	115
Median	12545	97	20973	163

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

An observation can be made that elite selection adds to GA efficiency, but fixed control parameters of the SGA, which were extremely good for the original version, are no longer appropriate. Assuming that the parameters used in FM'92 were optimal (aside from the population size), with addition of the elitist selection, another set of parameters is required to improve the algorithm performance. This is exactly what can be avoided with the use of random parameters in GAW, and the point we wish to make with this presentation.

The form of elite selection presented above was motivated by loss of low-order schemas from the population. Although unusable in general, its inclusion 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. In the next section we analyse possible advantages of variable length representation. Let us just reiterate that the elitist selection one can combine with GAW may be of different types. With Royal Road functions, fitness is assigned to parts of the string, and we use that information. Clearly, 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, elite selection with preservation of strings with maximum fitness does not hinder the performance.

5.Variable length representation

Unless elite selection is used (Section 4), 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 long time, prolonging 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 Wu and Lindsay's, but we believe it suits well the Royal Road function 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[10]. See Figure 7.

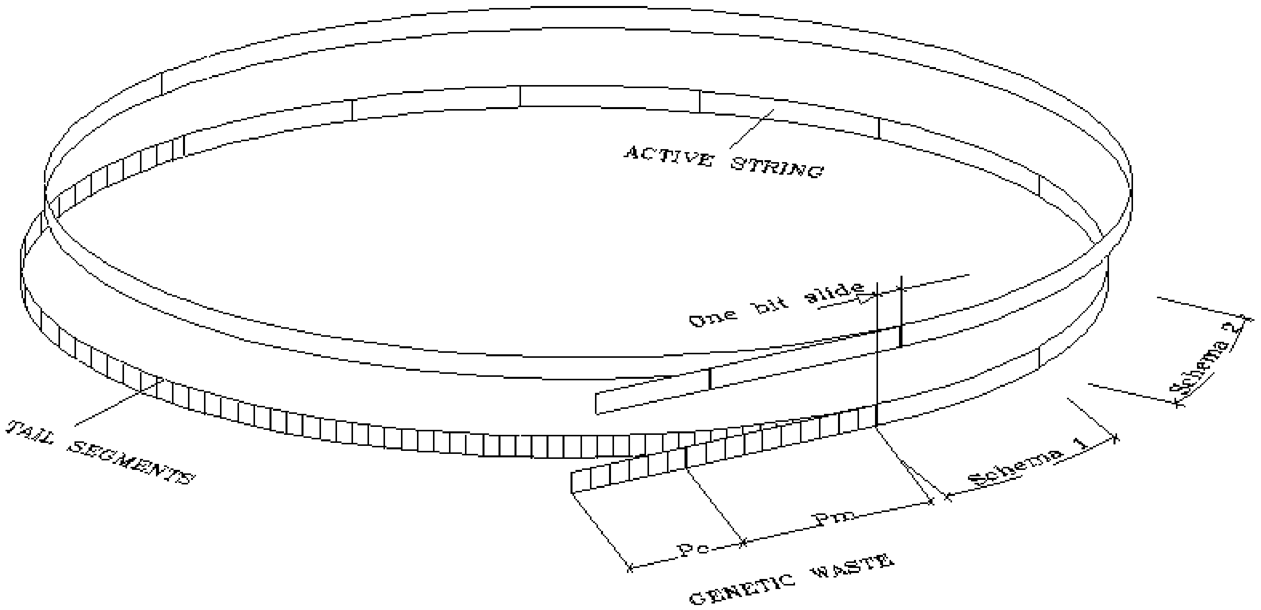

Figure 7: A ring representation of variable length. The tail segment increases the string length by some integer multiple of 8 alleles. The GA checks the string for fitness of all 8-tuple locations, and then it slides down the ring one bit at a time, repeating the process.

We first look at a zero-length tail segment, i.e. we close the original string (64 alleles) in a circle, and witness a change from the original mean of 60185 evaluations for R2 down to 47297

[10]We also applied the 8-tuple (schema)slide, but bit by bit explores the space more efficiently. Sliding down the ring by one schema at a time was on average 30% less efficient than the one bit-slide.

(369 generations). The improvement could be expected, as more information is contained in variable representation of building blocks, even this simple - the GA explores overlapping bits seven more times than before[11].

	R1- 72 bit string		R2- 72 bit string	
	eval.	gen.	eval.	gen.
Average	17717	139	25342	199
	(652)	(5)	(1055)	(8)
Std. Dev.	9203	72	14885	117
Median	15537	122	20463	161

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

	R1- 128 bit string		R2- 128 bit string	
	eval.	gen.	eval.	gen.
Average	34310	269	51234	402
	(1361)	(11)	(3632)	(16)
Std. Dev.	19198	151	29006	228
Median	29142	229	45828	360

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

Tables 8 and 9 illustrate the GAW with variable representation when tail with 8 bits is added and when 64 bits are added to the 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

[11]A note on reporting the results - we report one evaluation no matter how many fitness calculations were performed, as long as no crossover, mutation and selection were applied. In this case, 8 calculations of fitness were needed for each evaluation.

building block representation, as the low order schemas still have to be together in a block for best performance. As string length increases, the role of crossover operator decreases, since 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 fixed number of crossing sites are selected, or if each mating pair is exposed to randomized[12] selection of the number of sites. The first three rows in Tables 10 and 11 illustrate this for R1 and R2, respectively.

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

[12]A different number of crossing points (between 1 and the number of 8-tuples in the string) is selected for each pair of mates.

		R1-Number of crossing sites							
		2		4		8		Random	
Avg	No	39823	310	34604	269	36349	283	40725	317
	NCS	(1664)	(12)	(1800)	(14)	(1466)	(11)	(1626)	(12)
St.dev		23535	183	25455	198	20743	162	22996	179
Median		35385	275	27024	210	31378	245	33655	262
Avg	With	39576	308	42331	330	51011	398	48932	381
	NCS	(1670)	(13)	(2025)	(15)	(2081)	(16)	(2127)	(16)
St.dev		23623	184	28644	223	29433	229	30085	235
Median		33240	259	33707	262	42711	333	43455	339

Table 10: A multiple point crossover on R1. Mean, standard deviation and median for GAW 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.

		R2-Number of crossing sites							
		2		4		8		Random	
Avg	No	43831	341	43011	335	42091	328	44929	350
	NCS	(1815)	(14)	(1854)	(14)	(1640)	(12)	(1782)	(13)
St.dev		25678	200	26230	204	23204	181	25215	196
Median		37975	296	34766	271	36517	285	38434	300
Avg	With	42930	334	43381	338	57615	449	53093	414
	NCS	(2023)	(15)	(1810)	(14)	(2496)	(19)	(2348)	(18)
St.dev		28623	223	25609	200	35300	275	33218	259
Median		36801	287	36152	281	48259	376	45422	354

Table 11: A multiple point crossover on R2. Mean, standard deviation and median for GAW without non coding segments (first 3 rows, 64 bits) and with non coding segments (last 3 rows, 128 bits). Number of crossing sites 1,2,4,8, or randomly selected.

		R1-Number of crossing sites			
		1		Random	
Avg	No NCS	149862	1170	24106	187
		(17349)	(135)	(3129)	(24)
St dev		122680	958	22129	172

Median		115767	904	19397	151
Avg	128- NCS			26751	208
				(2647)	(20)
St dev				18723	146
Median				24918	194

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

		R2-Number of crossing sites							
		1		16		128		Random	
Avg	No NCS	109892	858	31939	248	24814	193	23558	183
		(11225)	(87)	(2526)	(19)	(1802)	(10)	(1802)	(14)
St.dev		79379	620	17866	139	12746	76	12746	99
Median		94596	738	28162	219	22553	175	19655	153
Avg	128 - NCS							39176	305
								(3738)	(29)
St dev								26435	206
Median.								33098	258

Table 13: Impact of a multiple point crossover on R2 with string length 1024 bits (first 3 rows). Addition of non coding segments (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. Addition of non coding segments to a long string does not significantly hamper the results.

6. Concluding remarks

In this paper we present the GAW version of a genetic algorithm for function optimization (Novkovic and Šverko 1997,1998), where non translated portions of the DNA provide random control parameters for the algorithm and we apply it to Royal Road functions. The GAW varies the probabilities of mutation and crossover, rather than use fixed parameters or invest in search for optimal parameter setting, which can be costly and uncertain. The GW part of the genetic makeup ensures that the performance of the GA does not depend entirely on the programmer's choice of initial control parameters (probabilities of crossover and mutation, in particular). We find the performance of GAW fairly reliable in most cases, confirming the result of our previous research where it proved robust with minimal deceptive problem (Novkovic and Šverko, 1998). Due to potentially high mutation, GAW proves most efficient when combined with elitist selection.

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

It is clear that population diversity brought about by varying the control parameters throughout the runs is likely to improve 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 GAW produces excellent results and can be safely used for optimization problems. One can combine other helpful alterations to increase the speed of search of the algorithm. Further research should illuminate most successful combinations, as well as their shortcomings.

References:

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

Berg,P. and Singer, M.(1992): *Dealing With Genes*, University Science Books, Mill Valey, CA

Booker, L. (1987): "Improving Search in Genetic Algorithms" in *Genetic Algorithms and Simulated Annealing*, Davis, L. D. (Ed.), Morgan Kauffman, Los Altos, California

Davis, L. (Ed.) (1987): *Genetic Algorithms and Simulated Annealing*, Pitman, London

Davis, L. (Ed.) (1991): *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York

Dawid, H. (1996): *Adaptive Learning by Genetic Algorithms: Analytical Results and Applications to Economic Models; Lecture Notes in Economics and Mathematical Systems 441*, Springer-Verlag Berlin Heidelberg

De Jong, K. (1975): *An Analysis of the Behavior of a Class of Genetic Adaptive Systems* Ph.D. dissertation, University of Michigan

De Jong, K. (1980): "Adaptive System Design: A Genetic Approach" *IEEE Transactions on*

Systems, Man and Cybernetics, 10,9:566-574

De Jong, K. (1992): "Genetic Algorithms are NOT Function Optimizers"in Whitley, D. (editor) *Foundations of Genetic Algorithms 2*, Morgan Kaufmann Publishers, 6-18

Forrest, S. and Mitchell, M. (1992): "Relative Building Block Fitness and the Building Block Hypothesis" in Whitley, D. (editor) *Foundations of Genetic Algorithms 2*, Morgan Kaufmann Publishers, 109-126

Goldberg, D.E. (1987): "Simple Genetic Algorithm and the minimal deceptive problem" in Davis, L., editor, *Genetic Algorithm and Simulated Annealing,* Pitman, London

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

Grefenstette, J.J. (1986): "Optimization of Control Parameters for Genetic Algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, 16, 1:122-128

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

Levenick, J. (1991): "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,123-127

Mitchell M., Forrest, S. and Holland, J. (1991): "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.

Mitchell M., Holland, J.and Forrest, S. (1994): "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

Novkovic, S. and Šverko, D. (1997): " 'Genetic Waste' and the Role of Diversity in Genetic Algorithm Simulations", *Proceedings of the Second Workshop on Economics with Heterogeneous Interacting Agents*, Ancona, May 30-31

Novkovic, S. and Šverko, D. (1998): "The Minimal Deceptive Problem Revisited: The Role of 'Genetic Waste' ", *Computers and Operations Research*, 25,11:895-911

Schaffer, D., editor (1989): *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann, Los Altos, CA

Schaffer, D., Caruana, R.A, Eshelman, L. and Das, R. (1989): "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization" in Schaffer (1989)

Schaffer, D. and Eshelman, L. (1991): " 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,61-68

Song, Y.H., Wang, G.S., Johns, A.T. and Wang, P.Y. (1996): "Improved Genetic Algorithms with Fuzzy Logic Controlled Crossover and Mutation", UKACC International Conference on Control, September 2-5, Conference Publication 427: 140-144

Spears, W. and De Jong, K. (1991): "An Analysis of Multi Point Crossover" in Rawlins, G. (editor) *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, 301-315

Srinivas, M. and Patnaik, L.M. (1994): "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, 24, 4: 656-667

Tanese, R. (1989): *Distributed Genetic Algorithms for Function Optimization*, Doctoral Dissertation, University of Michigan, Ann Arbor, MI.

Tuson, A. and Ross, P. (1998): "Adapting Operator Settings in Genetic Algorithms", *Evolutionary Computation*, 6, 2: 161-184

Wolpert, D. and Macready, W. (1997): "No Free Lunch Theorems for Search", *IEEE Transactions on Evolutionary Computation*, 1,1:67-82.

Wu, A.S. and Lindsay, R.K. (1997): "A Comparison of the Fixed and Floating Building Block Representation in the Genetic Algorithm", *Evolutionary Computation*, 4, 2: 169-193

Wu, A.S. and Lindsay, R.K. (1995): "Empirical Studies of the Genetic Algorithm With Non-coding Segments", *Evolutionary Computation*, 3,2: 121-147

Wu, A.S., Lindsay, R.K. and Smith, M.D. (1994): "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

Wu, Q.H. and Cao, Y.J. (1997): "Stochastic Optimization of Control Parameters in Genetic Algorithms", *IEEE* :77-80