J. Phys. A: Math. Gen. **30** (1997) 7849–7861. Printed in the UK PII: S0305-4470(97)84657-4

Genetic algorithms for continuous optimization problems—a concept of parameter-space size adjustment

Aleksandra B Djurišić†, Jovan M Elazar† and A D Rakić‡

Faculty of Electrical Engineering, University of Belgrade, PO Box 35–54, Belgrade, Yugoslavia

‡ University of Queensland, Department of Electrical and Computer Engineering, St Lucia QLD 4072, Brisbane, Australia

Received 27 May 1997, in final form 2 September 1997

Abstract. The concept of parameter-space size adjustment is proposed in order to enable successful application of genetic algorithms to continuous optimization problems. Performance of genetic algorithms with six different combinations of selection and reproduction mechanisms, with and without parameter-space size adjustment, were severely tested on eleven multiminima test functions. An algorithm with the best performance was employed for the determination of the model parameters of the optical constants of Pt, Ni and Cr.

1. Introduction

Optimization algorithms based on the analogy natural phenomena, such as simulated annealing [1] and genetic algorithms [2], find many applications in various fields. These algorithms have in common a high probability of locating the global optimum regardless of initial estimates, but the prices paid are large memory and CPU time requirements.

Genetic algorithms (GAs) have been recently proposed for solving complex optimization problems. Their application, due to the nature of the algorithm, is mostly restricted to optimization problems whose solution could be easily represented in binary form. In the search for optimal solutions GAs use the mechanisms of natural survival: selection, mating and mutation. These mechanisms are applied to the set of vectors called ‘strings’, that represents a population. Strings, referred to as chromosomes in the context of GAs, are possible solutions of the problem, which are, in our case, vectors of model parameter values. They are characterized by their ‘fitness’—performance with respect to some objective function. Strings with high fitness may enter the mating population, i.e. they may survive or give offspring in the next generation, while the strings with low fitness are killed off. The new population is formed by applying the crossover and mutation mechanisms to the chromosomes in the mating population. After a specified number of generations an optimal solution should be obtained.

Traditional GAs which employ binary-string representation of the solution space are not convenient for solving continuous optimization problems. By using the floating-point number coding, as suggested in [3, 4], with appropriate mechanisms of crossover and mutation, the length of the chromosome is given by the number of model parameters. Conversion of decimal equivalents into binary numbers, and vice versa, is avoided and hence the computational time is reduced. Moreover, the parameter values will not be altered or destroyed during the crossover operation, so that uncontrolled mutations introduced during the crossover in the case of binary coded strings are avoided.

The model-parameter estimation problem, which frequently arises in many areas, consists of determination of parameter values of the theoretical model on the basis of known experimental results. An algorithm capable of locating the global optimum without providing initial estimates should be employed for solving this problem [5–7]. However, even with floating-point number coding, GAs are not suitable for determination of model parameters due to their continuous nature. The main reason for this is the discrete sampling of the solution space. GAs are capable of locating roughly the global optimum, but a huge number of chromosomes in the population is necessary for any refinements. A method for solving that problem is proposed here. The concept of genetic algorithms with parameter-space size adjustment (GAPSSAs) is based on the idea of concentrating the search in the area where optimum is expected, by adaptively narrowing the boundaries for each parameter around the average value obtained for that parameter in the previous outer loop iteration. In such a manner more reliable and more precise location of the global optimum is made possible for considerably fewer chromosomes in the population. Due to the achieved improvements in precision of locating the global minimum, continuous optimization problems can be successfully solved using GAPSSAs, as shown here for the problem of model parameter determination of the optical constants of three metals.

The paper is organized as follows. In section 2 we describe the GAs and GAPSSAs proposed in this paper. Section 3 is devoted to the comparison of performance of these algorithms applied on eleven multiminima test functions. In section 4 a model for the optical constants of metals is described and an algorithm with the best performance is used to estimate its parameters for platinum, nickel and chromium.

2. Description of the algorithm

In order to implement a genetic algorithm we need to define the procedure of how to generate the population, and how to perform the operations of selection, mating, crossover and mutation.

2.1. Representation of chromosomes and population generation

A chromosome is represented by a string of finite length. To each element in the string, i.e. a gene, a value of the corresponding model parameter is assigned according to a floating-point number coding scheme [3, 4]. Therefore, operations of selection, reproduction and mutation suitable for real numbers are employed. Parameter values $p(k)$ in strings of the initial population are generated inside the initially set boundaries $p_{\mathrm{l}}(k)$ and $p_{\mathrm{u}}(k)$, according to the formula

$$p(k) = p_{\mathrm{l}}(k) + (p_{\mathrm{u}}(k) - p_{\mathrm{l}}(k))r \tag{1}$$

where r is a random number $r \in [0, 1]$.

2.2. Selection and mating

We investigated two different types of selection and mating. In the first case, we used the selection scheme of binary tournament [8]. The tournament population consists of two copies of every string in the current population. Strings of the tournament population are randomly grouped in pairs and the one with better fitness survives in the mating population, while the other is eliminated. If two strings in the tournament have the same fitness value, the surviving one is randomly chosen. Both copies of the best performing string survive in the mating population, while the worst is completely eliminated. The crossing over between two randomly chosen strings in the mating population is performed with the given crossover probability P_{cross}. If crossover does not take place, strings survive in the next generation. After performing the mutation, all strings are duplicated to form the new tournament population.

In the second case [9], P_{s} per cent of the new generation is produced by selection, and P_{c} per cent is produced by crossover. $N_{\mathrm{s}} = N * P_{\mathrm{s}}$ strings with the best fitness, where N is the number of strings in the population, enter directly the population of the next generation. The remaining number of strings in the new population is generated by crossover among the parent strings which are randomly chosen between all the strings in the current population. The probability of a string having offspring in the next generation depends on its normalized fitness

$$F(i) = \frac{f(i)}{\sum_{i=1}^{N} f(i)} \tag{2}$$

where $f(i)$ is the fitness value of the ith string. The strings with higher fitness have higher probability of being the parent strings. We obtained the best results with values $P_{\mathrm{s}} = 0.2$ and $P_{\mathrm{c}} = 0.8$.

2.3. *Crossover*

Operation of crossover exchanges subsets of elements between two chromosomes. If the subset consists of adjacent elements, it is an ‘ordered combination’ crossover, while in ‘uniform combination’ crossovers each element is randomly chosen, as stated in [10]. We investigated two types of crossover: two-point crossover, as a representative of the ‘ordered combination’ crossover, and crossover of randomly chosen genes, i.e. ‘uniform combination’ crossover. In the first case two points are randomly chosen, and elements of strings between the two points are swapped [3]. Figure 1(a) shows the two-point crossover, which is more efficient than the one-point crossover [3]. In the second case, as depicted in figure 1(b), we determine the number of elements to be swapped by generating a random integer $N_1 \in [n_{\mathrm{min}}, n_{\mathrm{par}}]$, where n_{par} is a number of model parameters, i.e. number of elements in strings, and n_{min} is the minimal number of elements exchanged in the crossover. Then we generate random integers $n_i \in [1, n_{\mathrm{par}}]$, $i = 1, N_1$ and swap elements at positions n_i.

2.4. *Mutation*

When a new generation is formed by means of selection and mating, we perform random mutations of parameter values in strings, referred to as genes, with probability of mutation P_{mute} for each gene. If the random number generated between 0 and 1 is less than the value of the mutation probability, the gene in the chromosome will be mutated. Mutation is necessary for maintaining a certain diversity in the population, i.e. to prevent the quick convergence to a local minimum. If the value of P_{mute} is to large, mutation no longer has the function of improving the performance of the population, because it causes the loss of some important genetic information, causing poor convergence. In the case of binary representation of the chromosomes, mutation is performed by simply inverting the value of the gene, while in the case of floating-point representation there must be adopted a method of mutation suitable for dealing with real numbers. In this paper, mutation is performed by generating the new value of the parameter in the same way as in the process of population generation. In our calculations, the best results were obtained with the value of the $P_{\mathrm{mute}} = 0.01$.

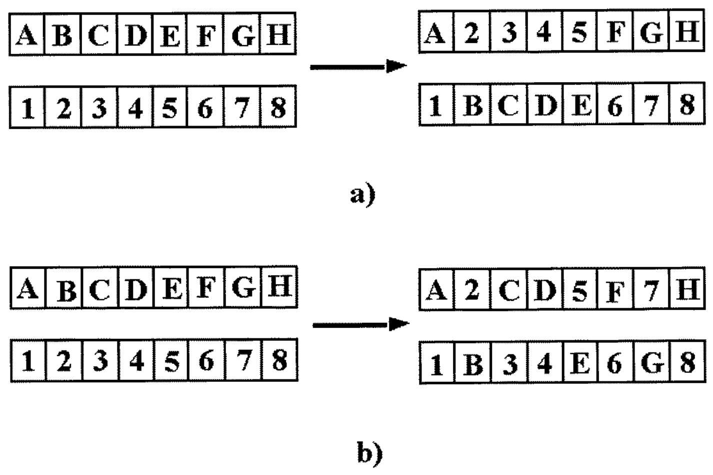

Figure 1. (*a*) Scheme of the two-point crossover, (*b*) scheme of the crossover of randomly chosen genes.

2.5. *Realization of the adaptive parameter-space size*

The main problem in using genetic algorithms for continuous optimization problems is the necessity of employing a very large number of strings in the population, which demands extensive computer resources. Because of the discrete sampling of the solution space, even rerunning the algorithm a number of times does not guarantee that the global optimum is located precisely, if it is found at all. The phenomenon that real-coded GAs can be blocked from further progress in certain situations has already been recognized and discussed [11]. Improving the performance of the real-coded GAs by introducing more appropriate crossover mechanism was proposed [12–14]. All suggested crossover operators (intermediate or line crossover) have one feature in common—they introduce new parameter values in between two parent ones. However, these new values are in a certain way bound to existing ones, so that the presented problem is not always solved successfully in this manner. Therefore, we introduce a concept of the adaptive parameter-space size, based on the idea that more accurate parameter values could be obtained by refining our guess about the dimensions of the solution space. The pseudocode of the GAPSSA algorithm is shown in figure 2. Within the initially set boundaries we generate a population and employ one of the classical GAs. Then we reduce the parameter-space size according to the average value of each parameter in the final population at the end of the inner loop. The new initial population at the beginning of the next inner loop is generated within the new boundaries and the genetic algorithm is performed again. The new boundaries are determined according to

$$p_{\mathrm{u}}(k) = p_{\mathrm{u}}(k) - c(p_{\mathrm{u}}(k) - \hat{\mu}(k)) \tag{3}$$

$$p_{\mathrm{l}}(k) = p_{\mathrm{l}}(k) + c(\hat{\mu}(k) - p_{\mathrm{l}}(k)) \tag{4}$$

where $\hat{\mu}(k)$ is the average value of the parameter $p(k)$ in the inner loop final population, and c is a predetermined positive number less than 1. At the beginning of the outer loop, we generate a new population that includes the string with the best-fitness in the inner-loop final population. The inner loop terminates if the best-fitness value remains unchanged in three consecutive iterations, or if the initially set of maximal number of generations is reached. The outer loop is performed $n_{\max}$ times, where $n_{\max}$ is initially set at maximal number of iterations.

```
do i=1, nmax
   generate population of N strings
   do while (stopping criterion is satisfied
   or
   max. number of generations is reached)
      evaluate fitness of each string
      perform selection
      perform mating and crossover
      perform mutation
   end do
   do j=1,npar
      compute
      update boundaries pl(k) and pu(k)
   end do
end do
```

Figure 2. Pseudocode of the GAPSSA algorithm.

3. Test of the GAs and GAPSSAs

To investigate the performance of the GAs and GAPSSAs we performed experiments on four families of test functions for different number of variables, i.e. algorithms were tested on eleven multiminima test functions. We also investigated the influence of the choice of selection and reproduction mechanisms to the quality of the final solution. Therefore, performance of the following algorithms was tested: GA1a, GAPSSA1a, GA1b, GAPSSA1b, GA1c, GAPSSA1c, GA2a, GAPSSA2a, GA2b, GAPSSA2b, GA2c and GAPSSA2c, where the numbers 1 and 2 denote the employed method of selection and letters a, b and c denote the employed method of crossover:

(1) tournament selection;

(2) selection of the N_s best performing strings;

(a) two-point crossover;

(b) crossover of random elements with minimal number of elements participating in the crossover equal 1;

(c) crossover of random elements with minimal number of elements participating in the crossover equal $n_{par}/2$.

The first investigated family of functions is given by

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} ax_i^2 + bx_i^2 \sin cx_i. \tag{5}$$

Section of $f(\boldsymbol{x})$ along an axis for values of $a = 0.2$, $b = 0.1$ and $c = 2$ is shown in figure 3. The obtained final objective-function values for 20, 50 and 100 variables where $x_i \in [-10, 10], i = 1, n$, each for $N = 2000$, $N = 1000$ and $N = 500$ strings in the population, are presented in table 1.

The second family of multiminima functions was investigated by Aluffi-Pentini *et al*

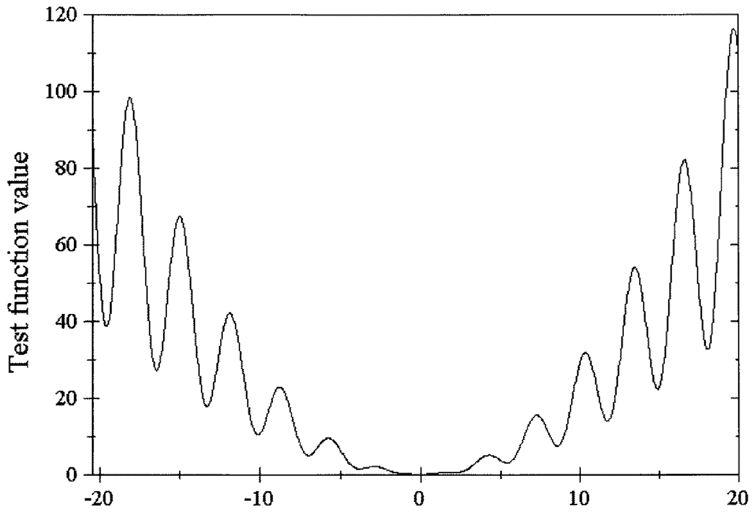

Figure 3. Section of f along one axis for $a = 0.2$, $b = 0.1$ and $c = 2$.

Table 1. Final values of objective function f for n_{par} variables and N chromosomes.

Algorithm	GA1a	GAPSSA1a	GA1b	GAPSSA1b	GA1c	GAPSSA1c
$n_{\mathrm{par}} = 20, N = 2000$	3.961E-4	2.223E-7	4.190E-3	8.762E-7	1.987E-4	8.773E-7
$n_{\mathrm{par}} = 20, N = 1000$	0.639	3.356E-6	5.532	3.722E-5	0.463	1.616E-6
$n_{\mathrm{par}} = 20, N = 500$	36.80	9.280E-3	80.42	1.779E-2	22.97	1.007E-2
$n_{\mathrm{par}} = 50, N = 2000$	0.402	8.244E-6	1.912	3.646E-5	0.188	1.586E-7
$n_{\mathrm{par}} = 50, N = 1000$	8.292	7.610E-3	24.333	1.169E-2	7.548	9.520E-3
$n_{\mathrm{par}} = 50, N = 500$	3.610E-3	3.210E-6	3.156E-2	1.927E-5	3.550E-2	1.116E-5
$n_{\mathrm{par}} = 100, N = 2000$	1.457E-2	1.787E-5	0.171	4.990E-6	3.617E-1	2.527E-5
$n_{\mathrm{par}} = 100, N = 1000$	10.98	8.990E-3	26.89	2.170E-2	8.139	9.680E-3
$n_{\mathrm{par}} = 100, N = 500$	4.812	1.902E-4	9.997	4.414E-4	2.769	3.142E-4
Algorithm	GA2a	GAPSSA2a	GA2b	GAPSSA2b	GA2c	GAPSSA2c
$n_{\mathrm{par}} = 20, N = 2000$	6.765E-2	1.906E-4	3.280E-2	5.267E-5	4.023E-2	6.901E-5
$n_{\mathrm{par}} = 20, N = 1000$	4.020	3.110E-3	0.933	9.592E-4	0.612	4.713E-4
$n_{\mathrm{par}} = 20, N = 500$	34.671	2.712E-2	3.039	6.420E-3	1.941	4.419E-3
$n_{\mathrm{par}} = 50, N = 2000$	3.429	7.790E-3	0.660	2.450E-3	0.557	2.040E-3
$n_{\mathrm{par}} = 50, N = 1000$	20.874	7.283E-2	3.908	1.120E-2	2.408	4.020E-3
$n_{\mathrm{par}} = 50, N = 500$	0.812	1.152E-4	0.334	1.087E-4	0.286	1.001E-4
$n_{\mathrm{par}} = 100, N = 2000$	5.967	6.080E-3	6.075	2.350E-3	6.147	3.130E-3
$n_{\mathrm{par}} = 100, N = 1000$	41.67	0.103	8.652	2.983E-2	6.097	2.220E-2
$n_{\mathrm{par}} = 100, N = 500$	49.42	4.384E-2	27.41	1.782E-2	25.90	1.482E-2

[15] and Dekkers and Aarts [16]. It is given by

$$g(\boldsymbol{x}) = \frac{\pi}{n}\left[k_1 \sin^2 \pi y_1 + \sum_{i=1}^{n-1}(y_i - k_2)^2(1 + k_1 \sin^2 \pi y_{i+1}) + (y_n - k_2)^2\right] \quad (6)$$

where $y_i = 1 + 0.25(x_i + 1)$, $k_1 = 10$, $k_2 = 1$, and $x_i \in [-10, 10], i = 1, n$. Figure 4 shows this function for two variables. Function g has roughly 5^n local minima. In cited references, this function was tested for three variables, and the results obtained for 2000

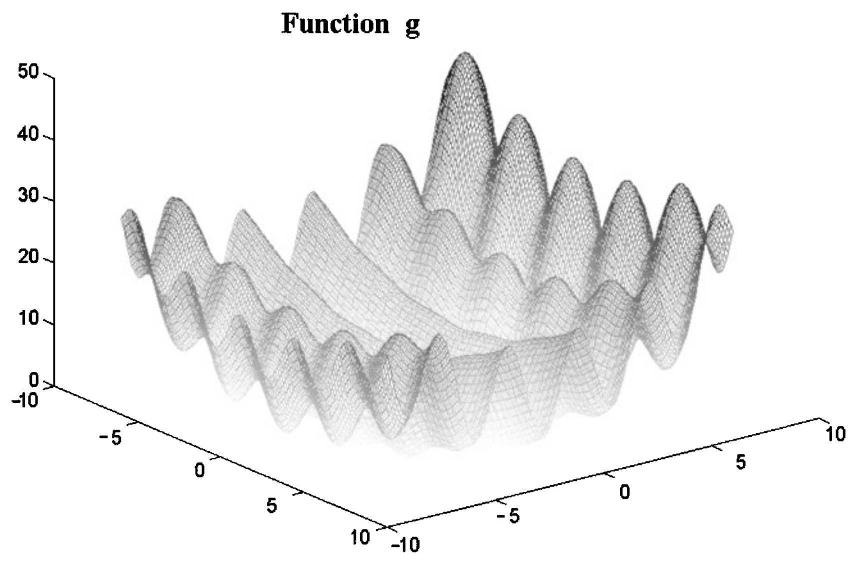

Figure 4. Function g for two variables.

Table 2. Final values of objective function g for n_{par} variables and 2000 chromosomes.

Algorithm	GA1a	GAPSSA1a	GA1b	GAPSSA1b	GA1c	GAPSSA1c
$n_{\text{par}} = 20$	1.707E-5	1.503E-7	3.701E-4	1.851E-8	1.163E-5	1.329E-8
$n_{\text{par}} = 50$	3.376E-2	6.165E-4	0.366	3.304E-7	6.470E-3	9.915E-7
$n_{\text{par}} = 100$	0.272	1.780E-4	0.787	1.916E-4	0.056	1.230E-4
Algorithm	GA2a	GAPSSA2a	GA2b	GAPSSA2b	GA2c	GAPSSA2c
$n_{\text{par}} = 20$	3.620E-2	4.360E-6	1.101E-3	1.660E-6	1.920E-3	7.010E-7
$n_{\text{par}} = 50$	1.826	2.470E-4	0.155	3.192E-5	0.136	3.347E-5
$n_{\text{par}} = 100$	1.314	9.584E-4	0.220	1.370E-4	9.310E-2	1.390E-4

strings in the population are presented in table 2.

The third family of multiminima functions, also investigated by Aluffi-Pentini *et al* [15] and Dekkers and Aarts [16] is shown in figure 5. This family of test functions is given by

$$h(\boldsymbol{x}) = k_3\Bigg\{\sin^2(\pi k_4 x_1) + \sum_{i=1}^{n-1}(x_i - k_5)^2[1 + k_6\sin^2(\pi k_4 x_{i+1})] \\ +(x_n - k_5)^2[1 + k_6\sin^2(\pi k_7 x_n)]\Bigg\} \tag{7}$$

where $k_3 = 0.1$, $k_4 = 3$, $k_5 = 1$, $k_6 = 1$ and $k_7 = 2$. In cited references, this function was tested for five variables $x_i \in [-5, 5]$, $i = 1, 5$, i.e. in the area where the function has roughly 15^5 minima. We performed tests with 20, 50 and 100 variables for $x_i \in [-10, 10]$, $i = 1, n$. The results obtained, also for $N = 2000$ strings in the population, are presented in table 3.

It should be noted that in this paper we investigated the performance of the algorithms on multiminima test functions for large number of variables (up to 100), while in literature up to date the largest number of variables in test functions was 10. All the above investigated functions have global minima equal zero. In all cases the GAPSSAs obtained a significantly

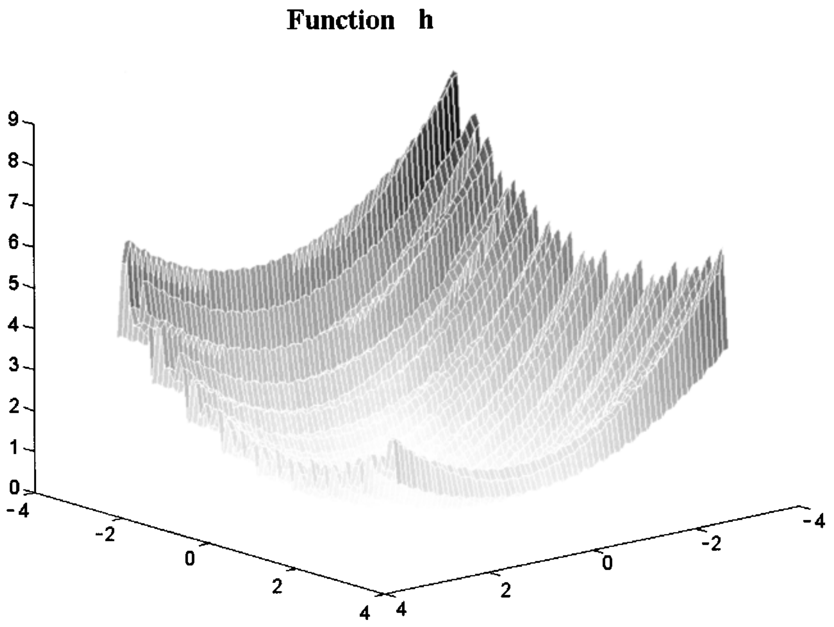

Figure 5. Function h for two variables.

Table 3. Final values of objective function h for n_{par} variables and 2000 chromosomes.

Algorithm	GA1a	GAPSSA1a	GA1b	GAPSSA1b	GA1c	GAPSSA1c
$n_{\mathrm{par}} = 20$	2.037E-4	1.335E-7	1.658E-2	4.134E-7	4.940E-4	1.662E-8
$n_{\mathrm{par}} = 50$	0.258	4.892E-6	1.749	1.262E-4	0.227	6.038E-6
$n_{\mathrm{par}} = 100$	9.891	6.320E-3	22.12	1.109E-2	7.175	6.780E-3
Algorithm	GA2a	GAPSSA2a	GA2b	GAPSSA2b	GA2c	GAPSSA2c
$n_{\mathrm{par}} = 20$	0.149	6.433E-6	1.745E-2	4.532E-5	5.328E-2	1.493E-5
$n_{\mathrm{par}} = 50$	3.015	1.810E-3	0.628	3.697E-4	0.684	4.020E-3
$n_{\mathrm{par}} = 100$	25.99	7.277E-2	8.377	1.020E-2	5.273	7.841E-3

lower objective-function value (for several orders of magnitude). It can be observed that GAs give nearly satisfactory results (about 10^{-2}–10^{-4} while GAPSSA give 10^{-5}–10^{-7}) only when the number of chromosomes in the population is large ($N = 2000$) and for a small number of variables ($n_{\mathrm{par}} = 20$). By reducing the number of chromosomes the performance of the GAs deteriorates significantly. For a larger number of variables ($n_{\mathrm{par}} = 50, 100$) GAs find values far from optimal, while GAPSSAs still obtain near optimal values, though higher than values obtained for ($n_{\mathrm{par}} = 20$). It can also be observed that GAPSSAs which incorporate the method of selection by tournament and the method of 'uniform combination' crossover with minimal number of exchanged elements that equals $n_{\mathrm{par}}/2$ obtained slightly better results than other GAPSSAs.

We also tested our algorithms on the Rosenbrock valleys, given by

$$r(\boldsymbol{x}) = \sum_{i=1}^{n} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2. \tag{8}$$

Table 4. Final values of objective function R for $n_{\rm par}$ variables and 2000 chromosomes.

Algorithm	GA1a	GAPSSA1a	GA1b	GAPSSA1b	GA1c	GAPSSA1c
$n_{\rm par} = 4$	3.453	2.405E-2	133.27	2.551E-2	7.087	1.671E-3
$n_{\rm par} = 10$	4.799	2.863	27.24	4.202E-2	0.290	8.901E-2
Algorithm	GA2a	GAPSSA2a	GA2b	GAPSSA2b	GA2c	GAPSSA2c
$n_{\rm par} = 4$	3.357E+5	9.540E-2	1.366E+4	1.214E-2	7.849E+3	2.933E-2
$n_{\rm par} = 10$	12.67	5.164	7.354	6.011	8.09	5.680

This family of functions, shown in figure 4 for two variables, was investigated by Corana *et al* [17] for two and four variables, and represents a very difficult test for an optimization algorithm. We used four variables $x_i \in [-200, 200], i = 1, 4$, as did Corana, and 10 variables $x_i \in [-10, 10], i = 1, 10$. The results obtained are shown in table 4. Again, GAPSSAs locate the global optimum while GAs fail to do so. For 10 variables only algorithms of GAPSSA1 family with 'uniform combination' crossover obtain optimal values, once again proving the superiority of these algorithms in escaping local minima.

4. Application to platinum, nickel and chromium

We shall briefly discuss the applied model for the optical dielectric function, which was often employed for modelling the optical constants of metals [5, 18]. It was shown [19–22] that the dielectric constant $\epsilon_r(\omega)$ can be expressed in the following form:

$$\hat{\epsilon_r}(\omega) = \hat{\epsilon_r}^{(f)}(\omega) + \hat{\epsilon_r}^{(b)}(\omega) \tag{9}$$

which separates explicitly the intraband effects (usually referred to as free-electron effects) from interband effects (usually referred to as bound-electron effects).

The intraband part $\hat{\epsilon_r}^{(f)}(\omega)$ of the dielectric constant is described by the well known free-electron or Drude model [23]

$$\hat{\epsilon_r}^{(f)}(\omega) = 1 - \frac{\Omega_p^2}{\omega(\omega + \mathrm{i}\Gamma_0)} \tag{10}$$

and the interband part of the dielectric constant $\hat{\epsilon_r}^{(b)}(\omega)$ is described by the simple semiquantum model resembling the Lorentz result for insulators [18]

$$\hat{\epsilon_r}^{(b)}(\omega) = -\sum_{j=1}^{k} \frac{f_j \omega_p^2}{(\omega^2 - \omega_j^2) + \mathrm{i}\omega\Gamma_j} \tag{11}$$

where ω_p is the plasma frequency, k is the number of interband transitions with frequency ω_j, oscillator strength f_j and lifetime $1/\Gamma_j$, while $\Omega_p = \sqrt{f_0}\omega_p$ is the plasma frequency associated with intraband transitions with oscillator strength f_0 and damping constant Γ_0.

The following objective function for determining the fitness of strings was used:

$$E(\boldsymbol{p}) = \sum_{i=1}^{i=N} \left[\left| \frac{\epsilon_{r1}(\omega_i) - \epsilon_{r1}^{\rm exp}(\omega_i)}{\epsilon_{r1}^{\rm exp}(\omega_i)} \right| + \left| \frac{\epsilon_{r2}(\omega_i) - \epsilon_{r2}^{\rm exp}(\omega_i)}{\epsilon_{r2}^{\rm exp}(\omega_i)} \right| \right]^2 \tag{12}$$

where $\epsilon_{r1}(\omega_i)$, $\epsilon_{r2}(\omega_i)$ are calculated values of the real and imaginary parts of the dielectric constant at frequency ω_i, while $\epsilon_{r1}^{\rm exp}(\omega_i)$, $\epsilon_{r2}^{\rm exp}(\omega_i)$ are the corresponding experimental values. In this section we applied the GAPSSA1c algorithm to determine the model-parameter

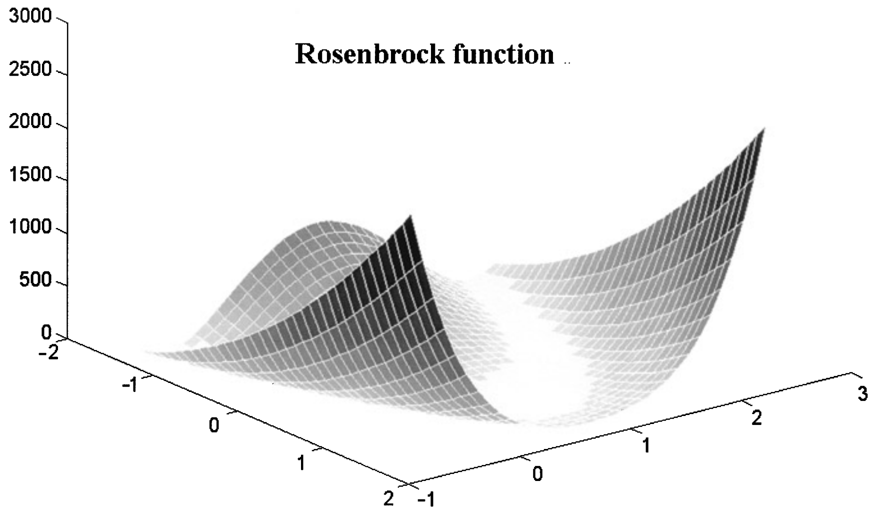

Figure 6. Function r for two variables.

values of the optical constants of the following metals: Pt, Ni and Cr. Values of the plasma frequencies of these metals were determined according to the definition

$$\omega_p = \left(\frac{Ne^2}{m\epsilon_0}\right)^{1/2} \tag{13}$$

where N is the concentration of the valence electrons. For all metals we used four oscillators.

To fit the semiquantum model to the data for platinum we used the experimental results given in [25], based on the work of Weaver *et al* [26]. Weaver [27–29] used reflectance and transmittance [30] data to obtain n and k by the Kramers–Krönig technique. The obtained oscillator-strength values correspond to the plasma frequency $\hbar\omega_p = 9.59$ eV. Figure 7 shows ϵ_{r1}, ϵ_{r2} versus energy for platinum (full curve—semiquantum model, open circles—experimental data). For nickel we used the experimental results given in [25]. The obtained oscillator-strength values correspond to the plasma frequency $\hbar\omega_p = 15.92$ eV. Figure 8 shows ϵ_{r1}, ϵ_{r2} versus energy for nickel (full curve—semiquantum model, open circles—experimental data). To fit the optical constants of chromium we used experimental results given in [31, 32]. The oscillator strength values correspond to the plasma frequency $\hbar\omega_p = 10.75$ eV. Figure 9 shows ϵ_{r1}, ϵ_{r2} versus energy for chromium (full curve—semiquantum model, open circles—experimental data). In all cases a good agreement between calculated and experimental values is obtained.

5. Conclusion

The problem of implementation of genetic algorithms to continuous optimization problems is discussed. In doing so, our prime concern was the possibility of applying these algorithms to the very specific and delicate problem: determination of the model-parameter values by minimizing the difference between experimental and calculated data. In this case, only precise and reliable location of global minima can provide the correct solution, since slightly different objective-function values can be obtained for significantly different parameter values. A concept of parameter-space size adjustment was proposed as a method of solving

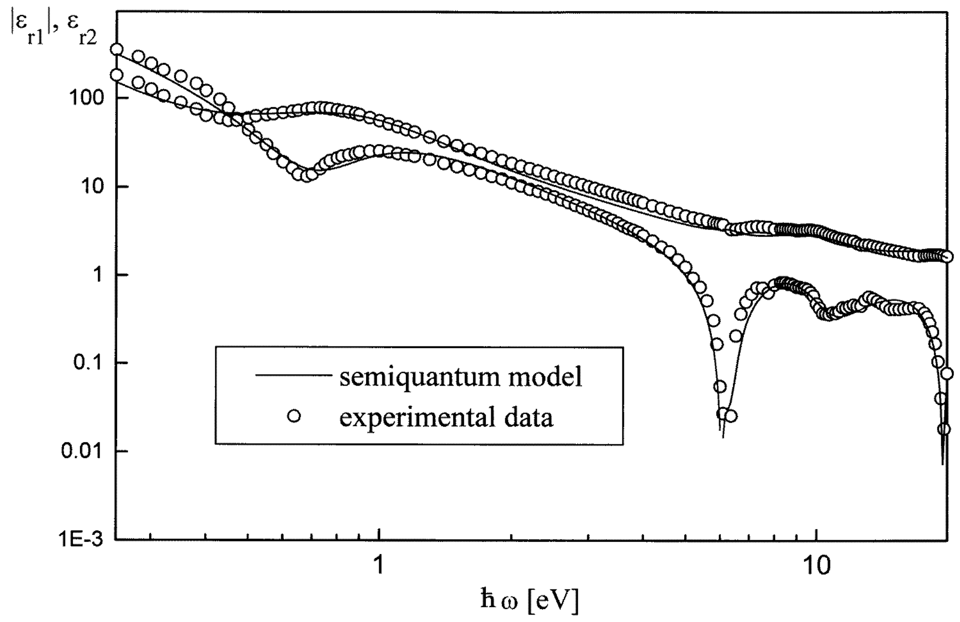

Figure 7. Real and imaginary part of the dielectric constant of Pt versus energy (full curve—model, open circles—tabulated data).

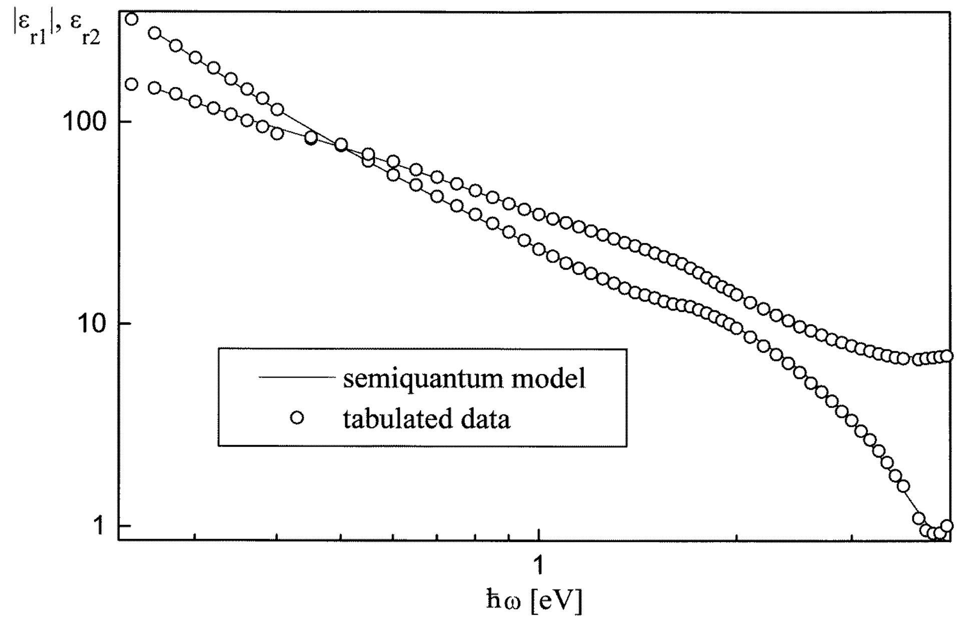

Figure 8. Real and imaginary part of the dielectric constant of Ni versus energy (full curve—model, open circles—experimental data).

the problems originating in discrete sampling of the solution space. We performed tests on four families of multiminima test functions for different numbers of variables (up to

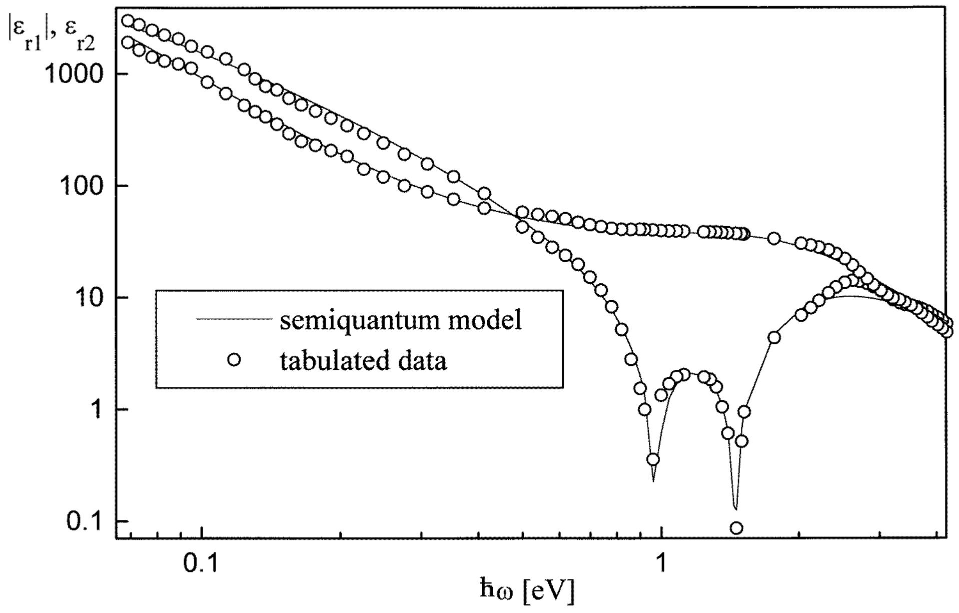

Figure 9. Real and imaginary part of the dielectric constant of Cr versus energy (full curve—model, open circles—experimental data).

100) and for different numbers of chromosomes of the population. It was proved that the performance of GAPSSAs are substantially improved compared with the performance of GAs. In other words, in all cases GAPSSAs obtain lower objective-function values. Also, it can be concluded that a combination of tournament selection and 'uniform combination' crossover gives the best results. In applying the algorithm with the best performance, GAPSSA1c, to parameter estimations of the semiquantum model of optical constants, we obtained good agreement with experimental results for all investigated metals.

References

[1] Kirkpatrick S, Gelatt C D Jr and Vecchi M P 1983 *Science* **220** 671–80
[2] Goldberg D E 1989 *Genetic Algorithms in Search, Optimization and Machine Learning* (Reading, MA: Addison-Wesley)
[3] Wong K P and Wong Y W 1994 *IEEE Proc. Gen. Trans. Distrib.* **141** 507–13
[4] Wong K P and Wong Y W 1993 *Proc. ANZIIS-93 (Perth, Western Australia)* pp 512–16
[5] Rakić A D, Elazar J M and Djurišić A B 1995 *Phys. Rev.* E **52** 6862–7
[6] Djurišić A B, Rakić A D and Elazar J M 1997 *Phys. Rev.* E **55** 4797
[7] Djurišić A B, Elazar J M and Rakić A D 1997 *Opt. Commun.* **134** 407–14
[8] Cienawski S E, Ehart J W and Ranjithan S 1995 *Water Resour. Res.* **31** 399–409
[9] Vemuri R and Vemuri R 1994 *Elec. Lett.* **30** 1270–2
[10] Curatelli F 1995 *Int. J. Electron.* **78** 435–47
[11] Goldberg D 1991 *Complex Syst.* E **5** 139–67
[12] Gutowski M 1994 *J. Phys. A: Math. Gen.* **27** 7893–904
[13] Mühlenbein H and Schlierkamp-Voosen D 1993 *Evolutionary Comput.* **1** 25–41
[14] Mühlenbein M 1995 *Genetic Algorithms in Engineering and Computer Science* (New York: Wiley) pp 59–82
[15] Aluffi-Pentini F, Parisi V and Zirilli F 1985 *J. Opt. Theor. Appl.* **47** 1–16
[16] Dekkers A and Aarts E 1991 *Math. Prog.* **50** 367–93
[17] Corana A, Machesi M, Martini C and Ridella S 1987 *ACM Trans. Math. Soft.* **13** 262–80
[18] Rakić A D 1995 *Appl. Opt.* **34** 4755–67

[19] Ashcroft N W and Sturm K 1971 *Phys. Rev.* B **3** 1898–910
[20] Ehrenreich H, Philipp H R and Segall B 1963 *Phys. Rev.* **132** 1918–28
[21] Ehrenreich H and Philipp H R 1962 *Phys. Rev.* **128** 1622–9
[22] Sturm K and Ashcroft N W 1974 *Phys. Rev.* B **10** 1343–9
[23] Marković M I and Rakić A D 1990 *Appl. Opt.* **29** 3479–83
[24] Powell C J 1970 *J. Opt. Soc. Am.* **60** 78–83
[25] Lynch D W and Hunter W R 1985 *Handbook of Optical Constants of Solids* ed E D Palik (Orlando, FL: Academic) pp 275–367
[26] Weaver J H 1975 *Phys. Rev.* B **11** 1416–25
[27] Yu A Y-C, Spicer W E and Hass G 1968 *Phys. Rev.* **171** 834–5
[28] Seignac A and Robin S 1972 *Solid State Commun.* **11** 217–19
[29] Hass G and Hunter W R 1974 *Space Optics* ed B J Thompson and R R Shanon (Washington, DC: National Academy) pp 525–53
[30] Haensel R, Radler K, Sonntag B and Kunz C 1969 *Solid State Commun.* **7** 1495–7
[31] Foiles C L 1985 *Landolt-Börnstein, Group III: Crystal and Solid State Physics* vol 15b, ed K H Hellwege and O Madelung (Berlin: Springer)
[32] Lynch D W and Hunter W R 1991 *Handbook of Optical Constants of Solids II* ed E D Palik (San Diego, CA: Academic) pp 341–419