Analytica Chimica Acta, 268 (1992) 123–134
Elsevier Science Publishers B V, Amsterdam

Optimization of calibration data with the dynamic genetic algorithm

Tong-Hua Li

Department of Chemistry, Tongji University, Shanghai 200092 (China)

C B Lucasius and G Kateman

Laboratory for Analytical Chemistry, University of Nijmegen, Toernooiveld, 6525 ED Nijmegen (Netherlands)

(Received 3rd January 1992)

Abstract

Genetic algorithms constitute a set of powerful search heuristics A modified genetic algorithm was used to optimize calibration data sets In order to construct an ideal genetic procedure, the diversity in a population is crucial The idea proposed is to estimate the diversities along two directions, namely the diversity between the chromosomes in a population and the diversity between the alleles in all chromosomes The newly defined diversity functions are able to describe the procedure of a genetic algorithm in detail and can be used as a feedback for dynamic control of the process in an almost ideal way The optimization results show that for both short and long runs the dynamic genetic algorithm is superior to the "classical" genetic algorithms and that after optimization not only can the data sets be compacted and refined but also the predictive ability of the calibration model can be improved

Keywords Optimization methods, Calibration, Dynamic control, Genetic algorithms, Wavelength selection

Genetic algorithms (GAs) are based on the simulation of natural selection, genetics and evolution and have been intensively studied and applied in recent years [1–10] Much of the interest in genetic algorithms is ascribed to the fact that genetic algorithms belong to the class of efficient domain-independent search strategies that are usually superior in performance to traditional methods without the need to incorporate highly domain-specific knowledge Several applicatons of genetic algorithms in machine learning, adaptive systems and optimization also show that genetic algorithms are more powerful search heuristics in noisy environments (an environment is defined by a given problem) where the space usually is large, discontinuous, complex and poorly understood Recently, genetic algorithms were introduced into chemometrics by Lucasius and Kateman [11–15] They devoted much attention to the applications of genetic algorithms in this discipline, and clearly pointed out that chemometrics is a fertile area for the application of genetic algorithms

In this work, a modified genetic algorithm was employed in a practical chemical problem Consider a calibration data set that is the response measured by an advanced chemical instrument and usually is a multivariate, multi-dimensional and multi-sample set Our purpose is to develop an optimum approach that can automatically select variables, diagnose outliers, retrench redundant data and build a calibration model that has powerful predictive capabilities and is robust in noisy environments This means that the method

Correspondence to Tong-Hua Li, Department of Chemistry, Tongji University, Shanghai 200092 (China)

should construct a subset (subspace) that contains the same calibration information as the original set except for part of the experimental and manual errors In general, this is a procedure of compacting and refining data sets Because a few erroneous data will drastically influence the reliability of the results, any intelligent system, e g , an intelligent instrument or an expert system, should be provided with this recognition capability

This optimization problem has attracted attention for a long time It is well known that the selection of variables (e g , wavelength) can reduce the calibration residuals [15,16] and avoid the spectral areas where no information is contained (essentially baseline or drift) Diagnosing outliers has been an active topic recently [17,18] and it is of benefit for building a calibration model [19] Recently, a new method, simulated annealing, was used as global optimization method with wavelength selection for UV spectrophotometry [20]

However, these traditional methods (except for the simulated annealing algorithm) generally work for one-dimensional cases and usually need highly domain-dependent knowledge The complexity makes the traditional methods useless when a high-dimensional optimization problem is involved Therefore, those methods which are domain-independent, called weak methods, are considered in this field

Apart from discussing the aforementioned analytical optimization problem, in this paper much attention is paid to genetic algorithms themselves and some important improvements According to the ideas of analysis of variance, new diversity functions are proposed and defined to describe the genetic process In addition, the defined functions are used as a feedback to control efficiently the GA's process in an ideal way

The principles of genetic algorithms are presented in the next section, followed by the diversity functions, their definitions and properties, and then the results and a discussion, including the description of a GA procedure an ideal GA procedure, dynamic control and the optimized results of near-infrared reflectance (NIR) data and liquid chromatographic–diode-array detection (LC–DAD) data

GENETIC ALGORITHMS

Genetic algorithms are adaptive generate-and-test procedures derived from the principles of natural population genetics There are numerous variants of GAs, and this section describes one of the most common Because most chemists are not familiar with this field, we shall first discuss the GA procedure briefly The main idea behind genetic algorithms is simple Figure 1 shows the skeleton of a genetic algorithm

During iteration t, the genetic algorithm keeps a population $P(t)$ which consists of a number of competing candidate solutions in which better performing individuals have a higher probability of surviving and generating later generations The wealth of information in the individuals is saved and propagated in a highly parallel fashion This intrinsic parallelism, for which Holland [1] laid the mathematical foundations, has the capability of making structural changes over time, discovering better and better individuals and/or improving the consistency of internal knowledge

The architecture of a genetic algorithm can be divided into five components representation, initiation, evaluation, genetic operations and genetic parameters

First, a genetic algorithm requires each candidate solution to be represented as a fixed-length string, or "chromosome" The chromosome, analogous to natural chromosomes, consists of "genes" for which usually a binary alphabet {0,1} is used In general, each gene on this abstract

```
genetic algorithm()
{
   t=0,
   initialize population P(0);
   evaluate P(0),
   while ( termination conditions )
     {
         t=t+1,
         select P(t) from P(t-1),
         recombine P(t),
         evaluate P(t),
     }
}
```

Fig 1 The skeleton of a genetic algorithm

chromosome is a group of bits Both theoretical and empirical results have shown that a bitstring notation is a powerful representation to encode a wide variety of information that is not necessarily bound to a restricted domain Other representations are available, although they have not been intensively studied In the present context, we use a pseudo-high-dimensional chromosome to represent a candidate subset The chromosome is divided into several segments according to the number of dimensions in the calibration problem Each segment represents a sampling manner in one dimension, {0,1} expresses a selection, that is, if the ith gene of a segment is "1", the ith column (or row) is selected to construct a subset which is a candidate for the solution The length of the whole chromosome has the fixed value m, n chromosomes construct a population In this way the population may be regarded as a matrix with n rows and m columns The sampling strategy is optimized and the global or near-global optimum subset is expected to be found at the end of the evolution process

The second component is the method of creating an initial population For practical reasons, a random initial population is frequently used In the present analytical application, special knowledge is available to put into the initial population For instance, some special wavelengths can be chosen in every subset of the initial population in order to gain a higher sensitivity Hence in this population, some alleles (columns in the population matrix) always take 1 s This should be done carefully, however, as genetic algorithms are notoriously opportunistic and may rapidly converge to a local optimum if the population contains a few chromosomes that are far superior to the remainder of the population [21]

The next component is called evaluation or fitness The evaluation function plays the role of the environment It provides a measure of the subset's fitness for the optimization problem It governs the extent to which chromosomes can or cannot survive in a population and influence the next generation The evaluation usually is expensive Typically it requires more than several hundred samples before genetic algorithms have sufficient information to bias strongly subsequent samples into successful subspaces [3] The highest price to be paid for searching a large space is the calculation of evaluation functions In the present context, the evaluation function is PRESS (predictive residual error sum of squares) of cross-validation For a sampled subset, a powerful calibration method, the modified principal component regression, is used to decompose the competitive subsets Because this method exploits both calibration and predictive measured data to build its model, only one decomposition is needed for every sampling The price therefore is evidently reduced After decomposition, cross-validation is used, i e , each time the "leave-on-out" method is used for predicting the left sample This is repeated until every sample has been left out once and only once PRESS is an average over all prediction residuals of squares in a subset

Another component, the most important in GAs, is the following set of variation operators the crossover operator, the mutation operator and the inversion operator The chief among these operators is crossover, through which the selected parents can pass their genetic material to their offspring Figure 2 illustrates the way in which the crossover operator works When two parents are selected, two breakpoints i and j are created at random, where $0 < j, j < m$ Two new offspring members are generated by exchanging the par-

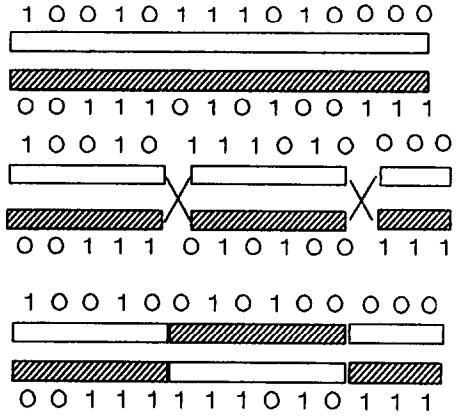

Fig 2 The crossover operation Two parent chromosomes exchange part of their genetic material and two offspring have new constructions

ents' segments and both contain part of the genetic material of their parents Crossover is an important operator in GAs, despite the fact that it is unable to create new information that does not exist in the parent structures

The way in which one can "discover" new or retrieve lost information is by using the mutation operator Mutation randomly changes the genes in a chromosome according to the mutation probability For example, in a randomly chosen chromosome a position is selected at random, if this gene is 0, it mutates to 1, and vice versa In the present context only crossover and mutation are used, the inversion operator was not used in this study

The last component, the genetic parameters, includes the size of a population, the probability of crossover, the probability of mutation, the number of generations, etc It is recognized that when a genetic algorithm is used to optimize a practical problem, the setting parameters themselves must usually also be optimized with regard to convergence speed Although the optimization of genetic setting parameters has been intensively studied [4,22], there is still a great freedom of choice for these parameters, and varying these parameters in run-time can improve the performance of GAs [21] In the present context, we shall introduce some new parameters that enable the genetic procedure to be performed more efficiently

DIVERSITY FUNCTIONS

Although genetic algorithms have been applied successfully to optimization problems in gas pipeline control [23], semiconductor layout [24] and the design of computer networks [25], there are still several performance issues that are not very clear It is our feeling that there is still much to learn about how to implement GAs efficiently with respect to time

The first issue is an inherent trade-off between exploitation and exploration which has been discussed in many publications [26,27] GAs could be thought of as search strategies between the random search and the hill-climbing strategies For the random search strategy, historical knowledge about estimates of the optimum solution is ignored and every region is sampled blindly Usually it is intolerably inefficient For the hill-climbing strategy, the best-so-far is exploited to limit the search space, other latent opportunities being ignored It is sensitive to pointing to an incorrect direction when the problem is complex

A genetic algorithm takes a balance between exploiting what already works best and exploring possibilities that might eventually evolve into something even better However, in practice, the optimum balance cannot be achieved automatically It is not clear whether the balance is optimum and how to control it as well as possible

The second issue is premature convergence, i e , the search becomes trapped in a local optimum before the global optimum is found In this situation, the crossover operator becomes less effective because some loci are fixed on alleles and the search space is reduced At the same time, lower mutation rates can hardly help to prevent the search from becoming trapped in local optima, whereas higher mutation rates will disrupt the proliferation of high-performance alleles in addition to poor ones A robust search strategy should anticipate the premature convergence and effectively avoid it In general, this is a balance between a broad search and a sufficient refinement

In order to gain a deep insight into the issues mentioned above, some tools or functions to monitor a genetic process and, if possible, to control this process are needed These tools should be domain-independent There are several functions that have been proposed to describe the GA behaviour In earlier work by DeJong [22], the allele loss was used as one of the criteria to choose the setting parameters Mauldin [28] proposed the minimum Hamming distance to enforce the needed diversity to avoid premature convergence He defined a uniqueness value that is an allowed distance between any offspring and all existing chromosomes in the population Whenever a new individual was generated and it contained an existing structure, the alleles were randomly changed in the offspring until the required distance was achieved Grefenstette [21]

measured the population entropy for his travelling salesman problem The population entropy is a good measure, that is, as the population converges, the entropy approaches zero However, his definition of entropy makes it suitable only for his particular problem Baker [29] observed that rapid convergence often occurs after an individual or a group of individuals has passed a large number of offspring to the next generation He and Booker [30] measured the percentage involvement, i e , the percentage of the current population producing offspring In this way they could anticipate rapid convergence and had a chance to prevent it

All of these efforts have in common that they want to make an ideal genetic algorithm The key to an ideal genetic algorithm is to maintain a high degree of diversity within a population Population diversity is crucial to a GA's ability to continue the fruitful exploration of the search space In contrast to other GA researchers, we think that the diversity should contain two contexts, one being the diversity between the chromosomes in a population and the other the diversity between the alleles in all chromosomes The former is concerned with the efficiency of genetic operators and the latter mainly expresses the state of convergence

For these reasons, we propose new diversity functions that can describe the behaviour of a genetic algorithm and can be used to control the genetic process to work in an ideal manner The ideal of these diversity functions originates from chemometrics, and is called ANOVA (analysis of variance)

Again, consider a population at time t as a matrix, in which a row expresses a chromosome and a column in an allele The statistical analysis is carried out along two directions, rows and columns Then the diversity functions, named the "between chromosome" diversity (BC) and the "between allele" diversity (BA), are defined as follows

$$BC = \left(\sum_i S_i^2/m - S^2/nm \right)/(n-1) \quad (1)$$

$$BA = \left(\sum_j S_j^2/n - S^2/nm \right)/(m-1) \quad (2)$$

where S is the sum of genes "1" in a population, S_i expresses the sum over a row, i is the row index, S_j expresses the sum over a column and j is the column index

As the genes are members of a binary alphabet here, the diversity functions have some important properties (the proof of these properties is presented in the Appendix), as follows

The diversity functions are indifferent to mutual exchange of two chromosomes in a population, i e , the diversity functions are independent of the permutations of the chromosomes

If a population is homogeneous in either 0 or 1, the diversity functions are zero This implies that there is no information in this population when the diversity functions all are zero

When all the chromosomes in a population are identical, the BC function is zero and the BA function holds a constant value that is dependent on how many genes "1" are in a chromosome Hence, if BC is low, convergence is expected to be achieved

On crossover, the BA function is unchanged In other words, BA is independent of the crossover operator

If a population "loses" a chromosome, i e , all the genes in this chromosome are zero, the maximum value of BC is m/n Moreover, if k chromosomes are lost, the maximum value of BC is $mk(n-k)/n(n-1)$ Because this expression is limited, the BC function is always less than m/n

When a population "loses" an allele, i e , if a gene "0" occupies the same position on all chromosomes, the maximum value of BA is n/m Moreover, when p alleles are lost, the maximum value of BA is $np(m-p)/m(m-1)$

The diversity functions are independent of the problem as long as the representation is binary They are able to describe comprehensively a genetic process We shall use them to describe a genetic process

RESULTS AND DISCUSSION

Description of genetic processes

Let us consider a key loop of a genetic algorithm which is illustrated in Fig 3 This loop is

slightly different from common genetic algorithms In addition to the crossover probability P_c and the mutation probability P_m, two new parameters, P_d and P_r, are added into this loop

P_d is a duplication parameter Duplication or multiple duplication can improve the performance of the process There are several reasons for using P_d the best-so-far or a group of them should survive in the next generation, so there is less risk of losing a better structure, when degeneracy or near degeneracy exists in problem space, multiple duplication can find most solutions [11], and when the number of duplications, P_d, is large, it can prevent an individual from generating a large number of offspring members in the next generation Consequently, the performance of the population is effectively improved Figure

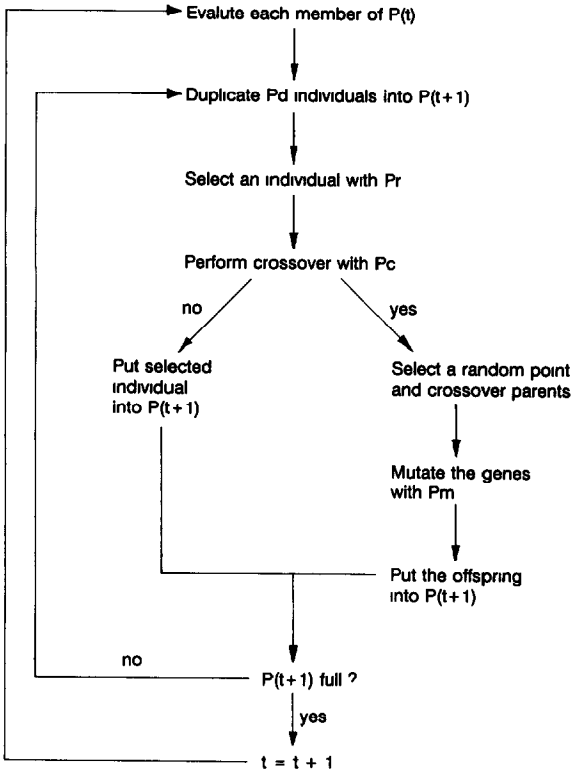

Fig 3 A key loop of the dynamic genetic algorithm The parameters P_r, P_c and P_m are defined in the text An ideal GA will be formed by dynamically by varying these parameters according to ANOVA

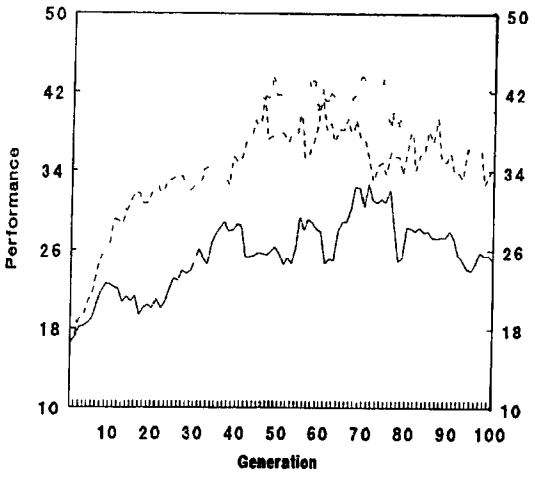

Fig 4 Influence of the duplication function (P_d) on performance The population performances of the first 100 generations are illustrated with different P_d (solid line, $P_d = 2$, dashed line, $P_d = 6$, dotted line, $P_d = 10$), with $P_c = 0.1$, $P_m = 0.001$, $P_r = 3$ The higher is P_r, the better is the performance

4 illustrates the relationship between P_d and the population performance, where the population performance is defined as the sum of the reciprocals of the evaluation functions (since in our problem lower variances correspond to better solutions, this definition gives the better individuals higher weights), other parameters are $P_c = 0.1$, $P_m = 0.01$, $P_r = 3$ Clearly, a higher P_d indicates a higher population performance

P_r is a selective or mating parameter In most applications, a Gauss weighed probability is assigned to every individual according to its evaluation In order to study the effects of different mating fashions, we construct a schedule as follows After evaluation of each member of $P(t)$, the individuals are sorted according to their evaluations (according to property "*a*", the diversity functions are unaffected), then each half part is assigned a different selection probability P_r is the ratio of the two probabilities When $P_r = 1$, i e , completely random selection, every individual has an equal opportunity of being selected as a parent When P_r is very large, the under-average-performing individuals have almost no opportunity of being selected Figure 5 shows that the *BC* and *BA* functions are both affected by P_r values, where $P_c = 0.6$, $P_m = 0.001$, $P_d = 2$ Varia-

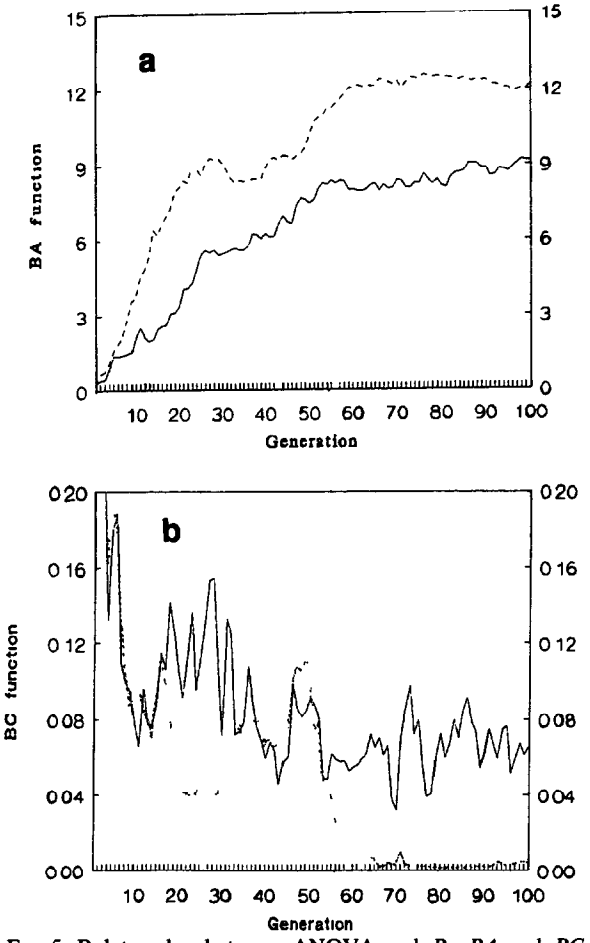

Fig 5 Relationship between ANOVA and P_r *BA* and *BC* functions of the first 100 generations with different P_r (solid lines, $P_r = 1$, dashed lines, $P_r = 3$) (a) *BA* function vs P_r, (b) *BC* function vs P_r, $P_c = 0.6$, $P_m = 0.001$, $P_d = 2$

tion of P_r can affect the exploitation of what is known so far

The P_c and P_m parameters are defined as usual They are the most important parameters in a genetic process Their effects can be described by *BC* and *BA*, respectively In Fig 6, where $P_m = 0$, $P_d = 2$, $P_r = 2$, the *BC* value clearly reflects the behaviour of different P_c values (property "*d*", *BA* is unaffected) When P_c is low, the diversity between the chromosomes disappears rapidly The genetic operators become ineffective In contrast, high P_c values enable the search to continue for a relatively long time before con-

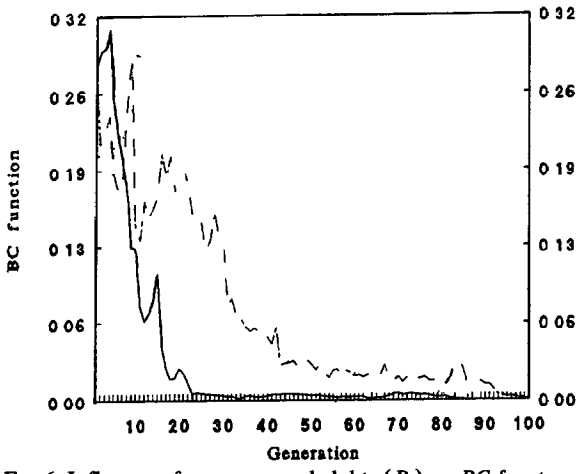

Fig 6 Influence of crossover probability (P_c) on *BC* function *BC* function of the first 100 generations with different P_c (solid line $P_c = 0.2$, dotted line, $P_c = 0.6$, dashed line, $P_c = 1.0$), with $P_m = 0$, $P_d = 2$, $P_r = 2$

vergence In most practical applications, a value, 0.6, is frequently used Clearly, this is not the best way of setting parameters It has been shown that a significant improvement can be gained by varying P_c [21] We shall vary P_c in a different way (see below)

Similar results can be found in Fig 7, where $P_c = 0.2$, $P_d = 2$, $P_r = 2$ *BA* describes the be-

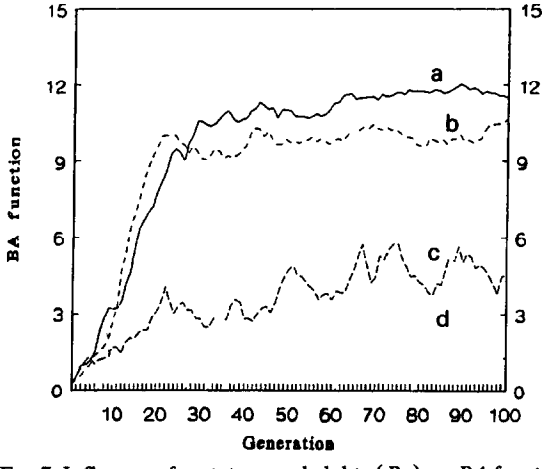

Fig 7 Influence of mutation probability (P_m) on *BA* function *BA* function of the first 100 generations with with different P_m (a, $P_m = 0.001$, b, $P_m = 0.01$, c, $P_m = 0.1$, d, $P_m = 0.2$) with $P_c = 0.2$, $P_d = 2$, $P_r = 2$

haviour for different P_m values When P_m is low, there is an evidently rapid convergence taking place (property "c") Usually, this indicates a premature convergence On the other hand, when P_m is high, BA varies around a relatively low value, and no convergence would be expected to be achieved in this situation This is the reason why most GAs use a lower P_m (usually P_m = 0 001) However, in this instance, there is great risk of convergence to local optimum points such as illustrated in Fig 7

In summary, the variances in a population can be expressed by the diversity functions They are useful for describing a genetic procedure and the effects of the genetic setting parameters As the efficiency and the convergence are the main problems in a genetic algorithm, we shall focus our attention on the construction of the optimum balance between them

Ideal genetic algorithm

So far we have discussed genetic algorithms, new diversity functions and a genetic procedure Now we propose a dynamic genetic algorithm that can approximate an ideal genetic procedure

An ideal genetic procedure is an approach such that the balance between exploitation and exploration can be automatically adjusted and the genetic operators will be most efficient It is an approach such that both a broad search through the problem space and a local refinement are given sufficient consideration It is also an approach such that high performance and convergence are reached simultaneously within a limited CPU time

A GA's strategy can be described by the diversity functions Figure 8 shows the different behaviours of the BC and BA functions with different strategies in a limited evolution time Strategy I quickly converges before it does a broad search In this situation, the premature converegence is frequently expected to take place, and this strategy often finds a local optimum instead of the global optimum In contrast, strategy III does too much random searching to converge, i e , there is a great risk of losing better structures of the solution and it is usually inefficient Strategy II is considered as an ideal genetic procedure It keeps

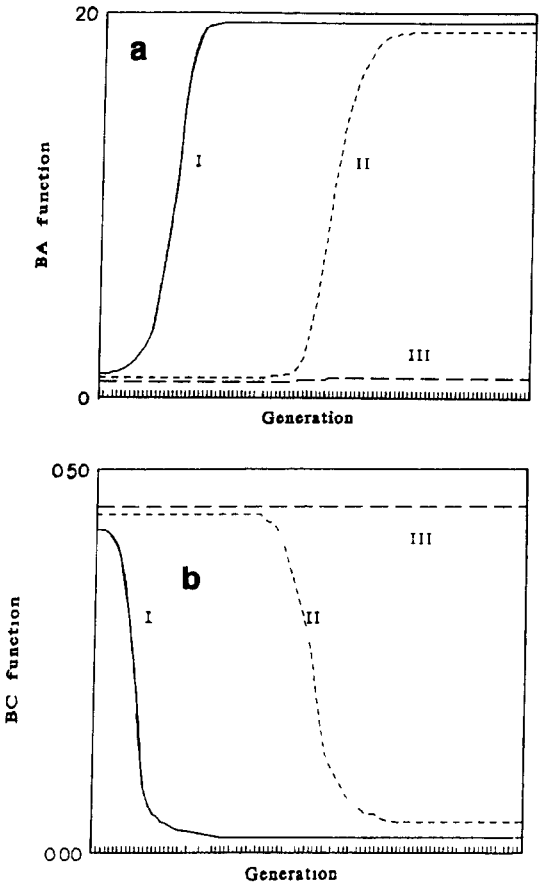

Fig 8 An ideal genetic procedure Strategy I, quick convergence, strategy II, an ideal genetic procedure, strategy III, random search

a high diversity in the search stage and has enough refinement to converge to a global or near-global solution This strategy can be implemented by a dynamic genetic algorithm

To construct a dynamic genetic algorithm, the whole procedure is divided into three stages initiation, search and refinement In the initial stage, the values of the diversity functions are dependent on the initial conditions of a genetic algorithm, so the diversity functions' behaviour is unpredictable The parameters are kept at their initial values in this stage The time of the initial stage is one tenth of the whole generation

In the search stage, the main parameters are designed to vary to guarantee a broad search and efficient exploitation For this purpose the *BC* function should be held at a higher level and the *BA* function at a lower level (properties *c*, *e* and *f*) If the *BC* value is less than a minimum critical limit $BC_{min}(m/n)$, the crossover probability is adjusted upwards slightly If the *BC* value is larger than a maximum critical limit, $BC_{max}(m/n)$, the crossover probability is forced downwards slightly The allowed range of the crossover probability is from 0 2 to 1 The lower limit of crossover guarantees that the genetic operators are not completely ineffective Meanwhile, the *BA* value is held in the range from $BA_{min}(n/m)$ to $BA_{max}(n/m)$ According to the *BA* value, the mutation probability should be varied from 0 001 to 0 2 to have sufficient alleles in a population and in order to avoid premature convergence, i e , when *BA* is higher P_m is adjusted upwards slightly and vice versa In this stage, P_d and P_r are held at low values The search stage is designed as the main part of the genetic procedure, e g , half of the generation time

In the last stage, refinement, the balance is designed toward exploitation and the search is forced in the local regions where there are higher probabilities of finding the optimum solutions In this stage the changes of the parameters are fixed by a linear annealing schedule The mutation probability decreases from the current value to the minimum value (0 001) P_d and P_r are linearly increased to their maxima ($P_d = 10$, $P_r = 5$) Only the crossover probability P_c is held at a constant value (0 6) in order for the refinement to be carried out efficiently The experiments show that the behaviour of a dynamic genetic algorithm is near strategy II

Optimized results

In this section we empirically explore the effect of the dynamic genetic algorithm discussed in the last section and present the optimized results of two calibration data sets

Near-infrared reflectance spectroscopic data Recently, near-infrared analysis has frequently been used for calibration by multiple linear regression and principal component regression However, outliers cause much trouble in many chemical applications The outlier diagnostics for regression are important both for the predictive ability and for interpretation [8] We recognize that the diagnosis of outliers and the optimization of variables in fact are based on the same concepts, as both methods are based on an accuracy and precision model that can be constructed by an optimized subset instead of all experimental data

A genetic algorithm is designed for optimizing published [8] near-infrared reflectance data The data used here consist of 28 samples and 19 variables (wavelengths), the experimental measurements of the samples are percentages of protein The size of the population is 50 and the length of a chromosome is 47 (28 + 19) bits

When diagnosing outliers by GAs, there is a restriction on the selection of the number of samples, i e , a minimum number of samples has to be limited This is because our evaluation function is PRESS, borrowed from cross-validation, for which the value is dependent on the number of samples Here, we let the minimum number of samples be ten

A dynamic genetic algorithm (DGA) and a traditional genetic algorithm (TGA) are programmed for comparison In order to compare their behaviours within an acceptable CPU time, the run time is limited to 30 s for short runs and to 300 s for long runs on a mainframe computer (NAS9600) in C language Table 1 shows a comparison of DGA and TGA in the short-run mode for ten repeats where each time a different random seed is used The initial parameters are $P_d = 2$, $P_r = 2$, $P_c = 0\,6$ and $P_m = 0\,001$ For DGA, BC_{min} is 0 2, BC_{max} is 0 4, BA_{min} is 2 and BA_{max} is 5 P_c and P_m are varied in the search stage In this stage, they are observed to increase simultaneously P_c frequently reaches the maximum value, i e , 1, and P_m is ca 0 1 The diversities are kept at relatively higher levels Except for the fourth and ninth runs, both the best-so-far and the population performances of DGA are better than those of TGA

Table 2 shows the results of the long-run mode For this mode the behaviours of both DGA and TGA are improved For TGA, two experiments

TABLE 1

Comparison of TGA and DGA in the short-run mode

No	TGA		DGA	
	Best-so-far performance	Population performance	Best-so-far performance	Population performance
1	2 0004	49 325	1 5911	60 971
2	1 8875	48 065	1 6341	56 662
3	1 6906	57 702	1 6732	59 642
4	1 2176	72 095	1 3855	63 486
5	2 0284	48 711	1 7830	54 951
6	1 5326	64 463	1 4797	61 206
7	1 7713	50 136	1 2846	76 018
8	2 2255	43 036	1 4970	64 085
9	1 6522	56 628	2 0290	46 639
10	1 2548	77 953	1 1951	80 922
Mean	1 7161	56 811	1 5552	62 458
Variance	0 1155	3 805	0 0815	3 297

are carried out with different P_m values When $P_m = 0\ 001$, the population performance is higher, implying that the search converges but to a local optimum (the best-so-far is the highest) When $P_m = 0\ 1$, a better best-so-far is found, but convergence is not reached It is evident that DGA is superior in both the best-so-far and overall performance

In the NIR data, there are three manual outliers which were diagnosed by Naes [8] with a two-step method that is fairly complex In fact, there are also some data that contain more experimental errors than other data We hold that the optimization must be carried out for samples and wavelengths simultaneously, regardless of manual or experimental errors In other words, a 28×19 matrix, in which some data carry the same useful information and some data contain considerable manual and experimental errors, is redundant for one component (protein) prediction Our purpose is to find an optimized subset that is expected to contain the same useful information and smaller errors compared with the original

In our experiment, the number of variables (wavelengths) and the number of samples are optimized as 8 and 10, respectively Therefore, a compact calibration subset, 10×8 matrix, is obtained by means of minimization of PRESS Consequently, the three manual outliers and some inaccurate samples are out of the subset The PRESS of the optimized subset is near 0 01 when the number of principal components reaches three It is considerably lower than the value of 1 20 for the original set and the value of 0 04 for the set of 25 samples without three manual outliers

LC–DAD data Liquid chromatography coupled with diode-array detection generates a two-dimensional response dependent on time and wavelength At each wavelength there is a chromatogram and at each time there is a spectrum Simultaneously optimizing high-dimensional analytical data is an important issue in chemometrics Although several powerful calibration methods claim that they are able to handle high-dimensional data sets without the need for pretreatment, higher precision can be obtained by the optimized combination of a subset of the original one [20] An experiment is programmed for demonstration of such improvement

The LC–DAD data used here result from the measurement of a two-component mixture, anthracene and phenanthrene They consist of 46 spectra and 40 wavelengths Six samples are calibration standards and five samples are predictions For comparison, the optimization is for wavelengths and spectra (i e , six samples are the calibration standard) The predictive abilities of before and after optimization are shown in Table 3

After optimization, 13 spectra and 20 wavelengths are selected to combine a calibration subset From Table 3 it is seen that the average rates of recovery of before and after optimization are nearly the same The variances are significantly different [$F_{9,9}(0\ 05) = 3\ 137$] by means of statistics ($5\ 29 > 3\ 137$) It is evident that after optimiza-

TABLE 2

Comparison of TGA and DGA in the long-run mode

Performance	TGA		DGA
	$P_m = 0\ 001$	$P_m = 0\ 1$	
Best-so-far performance	1 2734	1 2564	1 0853
Population performance	61 628	25 431	88 974

TABLE 3

Predictive abilities before and after optimization

Sample No	Real value	Before optimization		After optimization	
		Calculation	Recovery (%)	Calculation	Recovery (%)
1	6 46	6 52	100 93	6 37	98 61
	1 63	1 58	96 93	1 57	96 32
2	5 38	5 41	100 56	5 27	97 96
	2 72	2 71	99 63	2 67	98 16
3	1 62	1 50	92 59	1 62	100 00
	6 54	6 65	101 68	6 55	100 15
4	2 69	2 67	99 26	2 67	99 26
	5 45	5 47	100 37	5 42	99 45
5	4 04	4 07	100 74	3 87	98 28
	4 09	4 06	99 27	4 07	99 51
Mean			99 20		98 77
Variance [a]			0 887		0 385

[a] F-test = $0\,887^2/0\,385^2 = 5\,29$

tion the predictive ability of the calibration model is enhanced

Conclusions

Genetic algorithms have been applied to a simple chemometrics problem The diversity functions introduced have the power to describe the internal process of genetic algorithms and controlling it The designed dynamic genetic algorithm significantly improves the behaviour of the GA process within a limited evolution time Using GAs, it is possible to optimize calibration data sets and enhance the predictive ability of a calibration model successfully

However, the application of GAs is a young field, and there are abundant opportunities for both theoretical and practical studies from which there is still much to be learned Also, the power of GAs should be tested on higher level problems such as adaptive systems or the task of optimizing an executable code

The Netherlands Government is thanked for providing financial support to enable Dr T-H Li to do research in the Laboratory for Analytical Chemistry at the University of Nijmegen for 6 months Professor Naes is thanked for his kindness in allowing the authors to use his NIR data C B Lucasius acknowledges the financial support of the Dutch Foundation for Chemical Research (SON), the Dutch Foundation for Informatics Research (SION), and the Dutch Organization for Scientific Research (NWO), Grant 700-344-007 The reviewers are thanked for their critical comments and useful suggestions

APPENDIX

Analysis of variance (ANOVA) is a powerful statistical technique which can be used to separate and estimate the different causes of variation [21] Here it is exploited to estimate the diversities in a population According to the definition of between-sample estimation, our between-chromosome diversity function is

$$BC = m \sum_i (S_i - S)^2/(n-1) \tag{3}$$

where m is the length of a chromosome, S_i is a chromosome mean of genes and S is a total mean Equation 3 can be rewritten as

$$\begin{aligned} BC &= m \sum_i (S_i/m - S/nm)^2/(n-1) \\ &= \sum_i \left(S_i^2 - 2S_iS/n - S^2/n^2\right)/m(n-1) \\ &= \left(\sum_i S_i^2/m - S^2/nm\right)/(n-1) \end{aligned} \tag{4}$$

This is Eqn 1 in the main text

The proof of the properties of the diversity function is as follows

Properties a and d are easily understood

Property b let all genes be 1 in a population, i e , $S_i = m$, $S = nm$, then

$$\begin{aligned} BC &= (nm^2/m - n^2m^2/nm)/(n-1) \\ &= (nm - nm)/(n-1) = 0 \end{aligned} \tag{5}$$

Property c let each chromosome contain q alleles, i e , $S_i = q$, $S\text{-}nq$, $S_j = n$, then

$$\begin{aligned} BC &= (nq^2/m - n^2q^2/nm)/(n-1) \\ &= 0 \end{aligned} \tag{6}$$

$$\begin{aligned} BA &= (qn^2/n - n^2q^2/nm)/(m-1) \\ &= (qn - q^2n/m)/(m-1) \end{aligned} \tag{7}$$

When $q = m/2$, BA reaches the maximum, $nm/4(m-1)$

Property e assume a population "loses" k chromosomes and the others are full with gene 1 (in this case the diversity is maximum), then

$$S_t = m,\ S = (n-k)m$$

$$BC = \left[(n-k)m^2/m - (n-k)^2m^2/nm\right] /(n-1)$$

$$= mk(n-k)/n(n-1) \quad (8)$$

Property f this is similar to e

REFERENCES

1 J H Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975
2 L Davis and M Steenstrup, in L Davis (Ed), Genetic Algorithms and Simulated Annealing, Pitman, London, 1987, pp 1–11
3 K DeJong, Machine Learning, 3 (1988) 121
4 J J Grefenstette, IEEE Trans Syst Man Cybern, SMC-16 (1986) 122
5 J J Grefenstette (Ed), Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1985
6 J J Grefenstette (Ed), Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1987
7 J D Schaffer (Ed), Proceedings of the Third International Conference on Genetic Algorithms, Morgan–Kaufmann, San Mateo, CA, 1989
8 R K. Belew and L B Booker (Eds), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan–Kaufmann, San Mateo, CA, 1991
9 L Davis (Ed), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, NY, 1991
10 D E Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989
11 Ref 7, p 170
12 C B Lucasius, L M C Buydens and G Kateman, Self-Organizing Systems in Analytical Chemistry, in M Grasserbauer, J K.F Huber and W Wegscheider (Eds), Euro-Analysis VII, ASAC (Austrian Society for Analytical Chemistry), Vienna, 1990, p C2 5 L2
13 C B Lucasius, M J J Blommers, L M C Buydens and G Kateman, in L Davis (Ed), A Genetic Algorithm for Conformational Analysis of DNA, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, NY, 1991, Ch 18, p 251
14 C B Lucasius, S Werten, A H J M van Aert, G Kateman and M J J Blommers, in H-P Schwefel and R Manner (Eds), Conformational Analysis of DNA using Genetic Algorithms, Proceedings of the First Workshop on Parallel Problem Solving from Nature, Springer-Verlag, Berlin, 1991, p 90
15 C B Lucasius and G Kateman, Trends Anal Chem, 10 (1991) 254
16 I Y Bershtein, Fresenius' Z Anal Chem, 332 (1988) 332
17 S C Rutan and P W Carr, Anal Chim Acta, 215 (1988) 131
18 T Naes, Chemometr Intell Lab Syst, 5 (1989) 155
19 W Lindberg, J Ohman and S Wold, Anal Chem, 58 (1986) 299
20 J H Kalivas, N Roberts and M Sutter, Anal Chem, 61 (1989) 2024
21 J J Grefenstette, in L Davis (Ed), Genetic Algorithms and Simulated Annealing, 1987, pp 42–60
22 K.A DeJong, PhD Thesis, University of Michigan, Ann Arbor, MI, 1975
23 D E Goldberg, PhD Thesis, University of Michigan, Ann Arbor, MI, 1983
24 J Kim and J McDermott, in Proceedings of the National Conference on Artificial Intelligence, 1983, pp 197–201
25 L Davis and S Coombs, in J J Grefenstette (Ed), Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1987, p 252
26 L B Booker, D E Goldberg and J H Holland, Classifier Systems and Genetic Algorithms, Technical Report No 8, University of Michigan, Ann Arbor, MI, 1987
27 J M Fitzpatrick and J J Grefenstette, Machine Learning, 3 (1988) 101
28 M L Mauldin, in Proceedings of the National Conference on Artificial Intelligence, Morgan–Kaufmann, San Mateo, CA, 1984, p 247
29 J E Baker, in J J Grefenstette (Ed), Adaptive Selection Methods for Genetic Algorithms, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1985, p 101
30 L Booker, Machine Learning, 3 (1988) 61
31 J C Miller and J N Miller, Statistics for Analytical Chemistry, Wiley, New York, 1984