Struct Multidisc Optim 23, 63–74 © Springer-Verlag 2001

A genetic algorithm with real-value coding to optimize multimodal continuous functions*

M. Bessaou and P. Siarry

Abstract In this paper a new Genetic Algorithm (GA) to optimize multimodal continuous functions is proposed. It is based on a splitting of the traditional GA into a sequence of three processes. The first process creates several appropriate sub-populations using the information entropy theory. The second process applies the genetic operators (selection, crossover and mutation) on every subpopulation that is so gradually enriched with better individuals. We then determine the best point s^* among the best solutions issued from each of the preceding subpopulations. In the neighbourhood of this point s^* is generated a population used to initialize a traditional GA in the third process. In this last process, the population is entirely renewed after each generation, the new population being generated in the neighborhood of the best point found. The neighborhood size is decreased after each generation. A detailed comparison of performances with several stochastic global search methods is presented, using test functions of which local and global minima are known.

Key words global optimization, genetic algorithms, multimodal continuous functions

1 Introduction

Genetic algorithms are stochastic search methods which derive from a metaphor of the evolution process in nature (Proceedings 7th Int. Conf. on Genetic Algorithms 1997; Laucasius and Kateman 1992; Mitchell 1996). Their domain of utilization is very large, a review of their implementations and some application domains are given by De Jong (1975), Goldberg (1989). Among these applications, the training of concepts, or the recognition of shapes (Lutton and Martinez 1994) can be mentioned.

Experimental results show that through an appropriate choice of representation patterns of the elements of the search space and operators, genetic algorithms allow to solve a lot of "difficult" problems (Mi 1996) with reasonable computational costs. According to John Holland, genetic algorithms encourage the emergence and the maintenance in the population of relatively independent "pieces" of solutions, called building blocks (Goldberg 1989). The juxtaposition of these blocks, achieved by the crossover operator, produces complete solutions.

Up to now, genetic algorithms, like most other "metaheuritics", such as simulated annealing or tabu search, were mainly applied to "difficult" combinatorial optimization problems. Nevertheless, numerous practical applications need to tackle objective functions depending on several continuous functions. In such cases, classical gradient-based methods are quite efficient if both of the following conditions occur:

(i) the analytical expression of the objective function is known; and
(ii) the problem is unimodal, i.e. it admits only one optimal solution.

When condition (i) is not satisfied, the calculus of the gradients is very expensive in CPU time, and may lead to numerical instabilities, for instance if the objective function evaluations require the handling of experimental data (e.g. for identification of models, or for inverse problems). When condition (ii) is unsatisfied, gradient-based algorithms are trapped into one local optimum, which may be very bad in comparison with the best globally optimal solution. When gradient-based techniques are inappropriate, metaheuristics may be successfully used, since they make no use of the objective function derivatives and aim at finding one global optimum.

In the field of product and process improvement, the use of genetic algorithms (GAs) appears particularly at-

Received October 2, 2000

M. Bessaou[1] and P. Siarry[2]

[1] IUT de Cergy-Pontoise, Laboratoire de Modélisation des Systèmes en Electronique, Rue d'Eragny, Neuville-sur-Oise, F-95031 Cergy, France
e-mail: Mourad.Bessaou@iutc.u-cergy.fr
[2] Université de Paris 12, Laboratoire d'Etude et de Recherche en Instrumentation, Signaux et Systèmes, 61 avenue du Général de Gaulle, F-94010 Créteil, France
e-mail: siarry@univ-paris12.fr

* This paper was originally submitted to and accepted by the former journal *Design Optimization*

tractive: for instance, GAs were applied to the design of turbines (Axelsson 1993), the design of engine blocks (Fisher 1993), load balancing (Vavak *et al.* 1995), aerodynamic design (Périaux *et al.* 1995), stiffness maximization of laminated plates (Potgieter and Stander 1998), and optimal reservoir system operation (Wardlaw and Sharif 1999)

In this paper, we propose a new technique for the optimization of multimodal continuous functions, using a genetic algorithm with real-value coding, called RCGA. RCGA uses several existing techniques such as the real coding and the composition of sub-populations based on the entropy theory. The novelty of RCGA lies in the proposal of a careful equilibrium between both tasks usual in heuristic search, namely "intensification" and "diversification". RCGA first performs the diversification by means of subpopulations well dispersed within the whole search space, which prevents GAs from premature convergence. Intensification is then efficiently performed by using a population wholly renewed after each generation and selected within a progressively reduced neighbourhood of the best point found.

Before presenting RCGA in Sect. 3, we recall in Sect. 2 some generalities about genetic algorithms (GAs). In Sect. 4, we compare the performances of our algorithm with those of several global optimization heuristic methods, using a collection of test functions, of which local and global minima are known. In conclusion, we point out the present perspectives of this work.

2
Principle of genetic algorithms

A GA makes a population, that is a set of configurations called chromosomes, evolve. This type of algorithm uses three basic operators: the selection, the crossover and the mutation (see Fig. 1). The selection operator works out a new population starting from the current one, by encouraging the chromosomes having the strongest fitness (degree of adaptation), that is those which are in the neighbourhood of the global minima of the "objective function". The crossover operator uses the information contained in two chromosomes, in order to build two others. The mutation performs a random transformation of a chromosome, in order to bring diversity.

GAs can be viewed like techniques of local search with two levels of neighbourhoods: a local neighbourhood, characterized by the mutation operator (which locally changes a given configuration), and an extensive neighbourhood, constituted, for a given population, by all the configurations that may be obtained by crossover.

- choice of coding.

- randomly generate an initial population.

- *Repeat*

Selection
Crossover
Mutation

- *until* a given stopping criterion.

Fig. 1 Principle of a genetic algorithm

2.1
Representation of chromosomes

The first problem met during the utilization of GAs is the representation of the individuals. This coding is connected to the specifications of the optimization problem. For the optimization of functions of continuous variables, the classic binary coding is not well adapted. Indeed GAs were defined to handle discrete variables of which values vary with a small step (alphabet). In continuous optimization, the representation of continuous variables using a finished alphabet causes difficulties. Our attention was turned to the utilization of real coding. So the x chromosome is coded in the form of a vector of real numbers, every component of x represents a variable of the function f (Mi 1996).

2.2
Selection

During the selection, the "parent" individuals aimed at producing the "child" chromosomes are chosen. The first step consists in calculating the objective function value at every individual. Each individual will then be reproduced with a probability which, in the case of the minimization of functions, is inversely proportional to the value of the objective function of the individual at hand. Several methods exist in the the literature to achieve this task. In our work we chose to use the "roulette wheel selection" algorithm (called also "stochastic sampling with replacement"), which is characterized by its simplicity. The individuals are arranged in the form of adjoining segments, the length of every chromosome's segment is proportional to its selection probability. A random number, comprised between 0 and 1, is then generated. The individual of which segment contains the drawn number is selected. This operation is repeated as many times as required by the size of the population.

2.3
Crossover

The crossover operator is acknowledged as one of the main causes of the efficiency of genetic algorithms: it

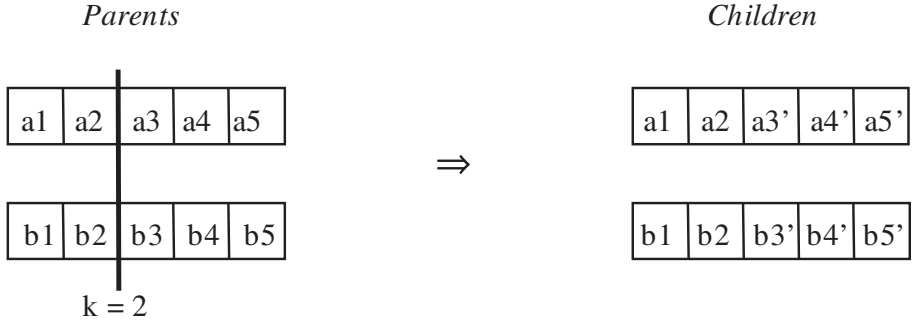


Fig. 2 Description of the cross-over operator $a'_i = b'_i = (a_i + b_i)/2$ for $i > k$

allows us to combine some hopeful schemata and thus quickly progresses towards the optimal regions of the search space. The crossover produces new individuals by combining the information contained in the parent chromosomes. Good results can be obtained with a random matching of the individuals (Goldberg 1989). Each pair generates two children who replace their parents inside the population. Single-point crossover is the simplest form of this operator: one crossover position is randomly selected and the variables situated after this point are exchanged between the individuals, thus producing two offspring. In Sect. 3.3 we describe the crossover used by us, which is a particular single-point crossover, shown in Fig. 2. Other forms of crossover are available, especially the following ones.

- *Multi-point-crossover*: m crossover positions are chosen, then the variables between successive crossover points are exchanged among the two parents to produce new offspring.
- *Uniform crossover*: a crossover mask is created at random and the parity of the genes (bits) in the mask indicate which parents will supply the offspring with which bits.
- *Intermediate recombination*: that method is usable for real variables. The values of the offspring variables are chosen from the values of the parents variables according some rule.

3 The proposed algorithm

The problem under consideration is the minimization of a function f

$$f : D \to \mathcal{R}\,, \quad D \subset \mathcal{R}^n\,, \tag{1}$$

where n represents the dimension of the function f; D represents the search space

$$D = \{x \in \mathcal{R}^n / A_i \le x_i \le B_i, i = 1, \ldots, n\}\,. \tag{2}$$

3.1 Structure of the population

Generally a GA starts with a single population, randomly generated inside the domain D. One of the difficulties of GAs is that they often converge too quickly and tend to make quickly uniform the population of the chromosomes. Consequently, they are easily trapped into local minima of the objective function. This difficulty is mainly due to the premature loss of diversity of the population during the search. To solve in part this problem, we use another organization of the population at the beginning of the algorithm. The overall idea is the following one: instead of organizing the population in the form of

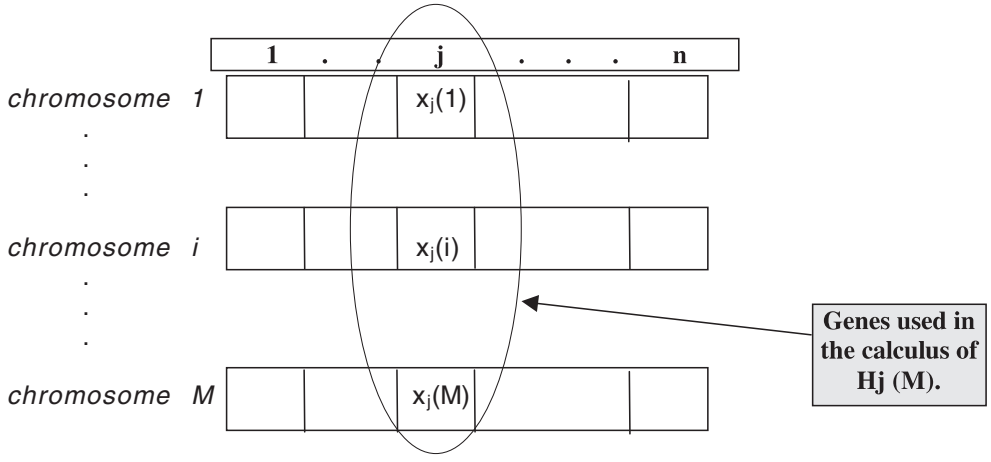


Fig. 3 Determination of entropy of gene j

a "flat set" (one single population), we split the population into subpopulations. The advantage of this technique is that it permits an exploration of several minima of the function at the same time, what allows to keep, for the second step of the algorithm, only the best point found.

3.2 Initialization

The composition of the subpopulations greatly influences the performances of a GA. More varied are these sub-populations, better is the convergence, in respect to both the time of convergence and the accuracy of the solution.

With this aim in view, we used the entropy originating from the information theory, as suggested by Chun *et al.* (1997). Let be M the size of every subpopulation (see Fig. 3), the entropy of the jth gene is

$$H_j(M) = \sum_{i=1}^{M} \sum_{k=i+1}^{M} -P_{ik} \log P_{ik} \,, \tag{3}$$

where P_{ik} represents the probability that the value of the jth gene ofthe ith chromosome is different from the one of the jth gene of the kth chromosome. It is determined in the following way:

$$P_{ik} = 1 - \frac{\mid x_j(i) - x_j(k) \mid}{B_j - A_j} \,, \tag{4}$$

where $[A_j, B_j]$ is the variation domain of the jth gene.

The average entropy $H(M)$ of the subpopulation is equal to the average of the entropies of the different genes

$$H(M) = \frac{1}{n} \times \sum_{j=1}^{n} H_j(M) \,. \tag{5}$$

The diversity of the population can be derived from (5). If, for example, the subpopulation is made up of only

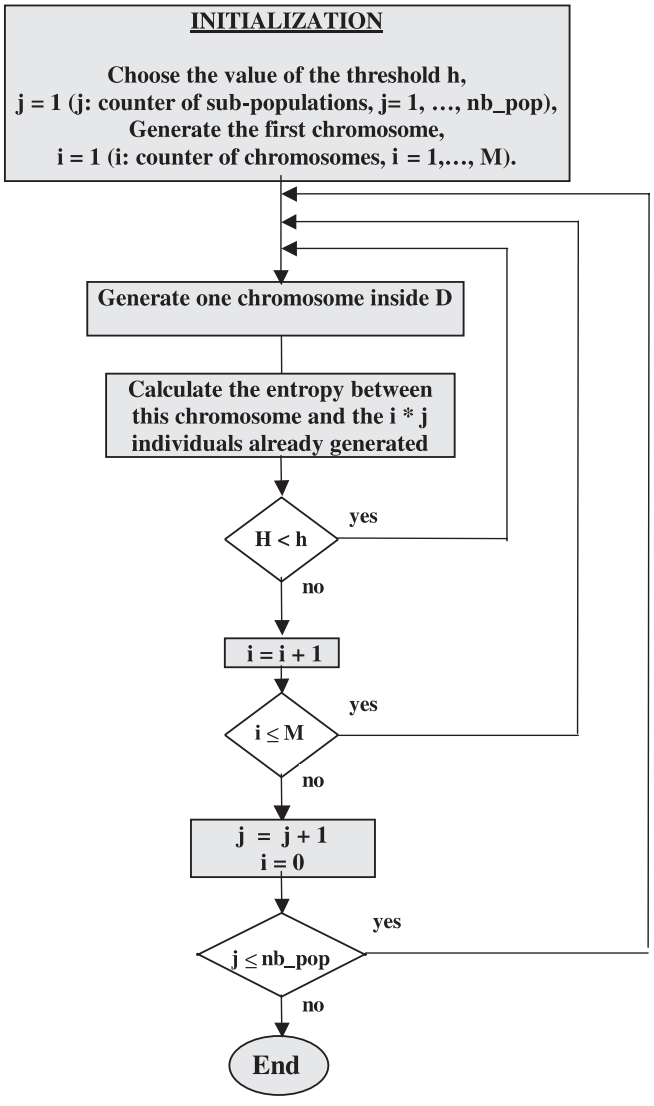
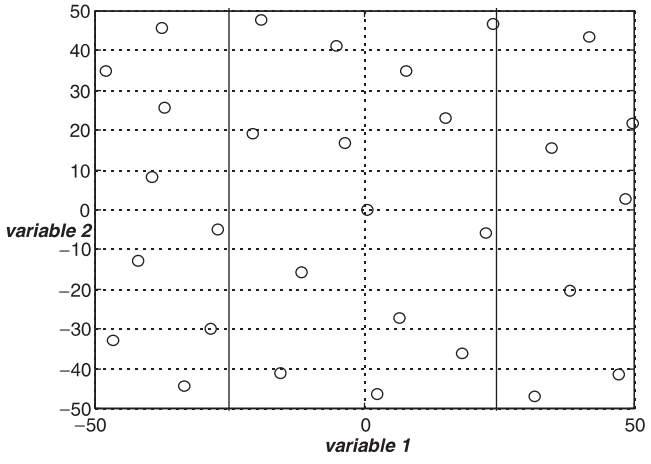
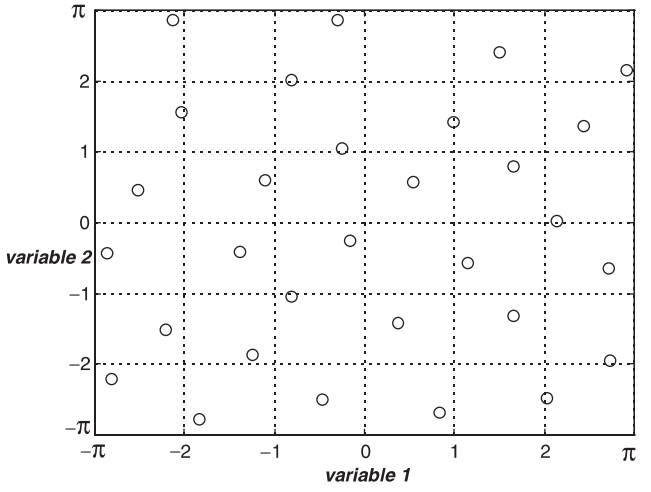


Fig. 4 Generation of the subpopulations



(a) *Bohachevsky function*



(b) *Michalewsicz function*

Fig. 5 Distribution of the initial subpopulations (30 individuals) for two functions of two variables (with a threshold $h = 0.2$.) (a) Bohachevsky function, (b) Michalewicz function

copies of a same chromosome, the average entropy value is equal to zero. In this case, we can say that it is the same for the diversity. Therefore the more varied the chromosomes, higher the entropy of the population, and the better is its "quality".

Every time a new chromosome is generated, the entropy between this one and the previously generated individuals is calculated. If this information quantity is higher than a threshold h, fixed at the beginning, the current chromosome is accepted. Otherwise, it is rejected and will be replaced by another one. This process is repeated until the M chromosomes of every subpopulation are generated (see Fig. 4). This method induces a good distribution of the initial individuals within D. Besides, it allows us to obtain disconnected subpopulations, that is to say the chromosomes of each subpopulation are not generated in the same neighbourhood, whatever its dimension may be. Our results for two functions of two variables (Bohachevsky' and Michalewicz'), for which the dimensions of D are respectively "large" and "small" (respectively -50 to 50 and $-\pi$ to π for each variable), are represented in Fig. 5.

3.3 Crossover

Crossover allows us to generate new individuals (children) starting from existing ones (parents). In our algorithm, we use a single-point crossover, the children are built inheriting from their parents the values of their variables. This operator is performed in the following way: an integer k, representing a component of the $\mathbf{x}$ vector, is randomly generated between 1 and $n-1$ (n being the dimension of the function to be optimized). Two new individuals may, for example, be created by calculating the half sum of parental genes, for all the genes comprised between k and n (see Fig. 2).

This operator is applied in a random way, with a P_c probability typically comprised between 0.6 and 0.9 (Goldberg 1989).

3.4 Mutation

This operator changes the value of a gene x_i, what brings the diversity among the population. Similarly to the temperature of simulated annealing, the mutation allows the exploration of the search space so that the algorithm "visits" several minima of the function. This exploration depends on two parameters, the mutation mode of the genes and the probability P_m of applying this operator.

– If a variable x_i is selected to be mutated (practically, this is done by generating a random number in the interval [0,1]: if this number is lower than P_m, the gene is mutated, otherwise it stays unchanged), the mutation operator randomly changes the value of this variable by generating a random number within the variation domain of x_i. This domain must not be kept constant throughout the algorithm, since the regions not likely to contain the global minimum must be eliminated.
– Like the crossover operator, the mutation is applied with a probability P_m, which is classically fixed to a "small" value throughout the algorithm. However, the theoretical convergence towards the global optimum of a GA, operating with a constant probability of crossover (P_c), is ensured, if the probability of mutation $P_m(k)$ follows a given decreasing law, in function of the generation number k (Davis and Principe 1991). From a practical viewpoint, like in the case of simulated annealing, we use a fast decreasing rate (otherwise, a prohibitive number of generations would be necessary to ensure convergence of the GA towards the global optimum)

$$P_m(k) = P_m(0) \times \exp(-k/\alpha)\,, \tag{6}$$

where $P_m(0)$ is the initial value of the mutation probability [typically $P_m(0) = 1$], a is calculated so that

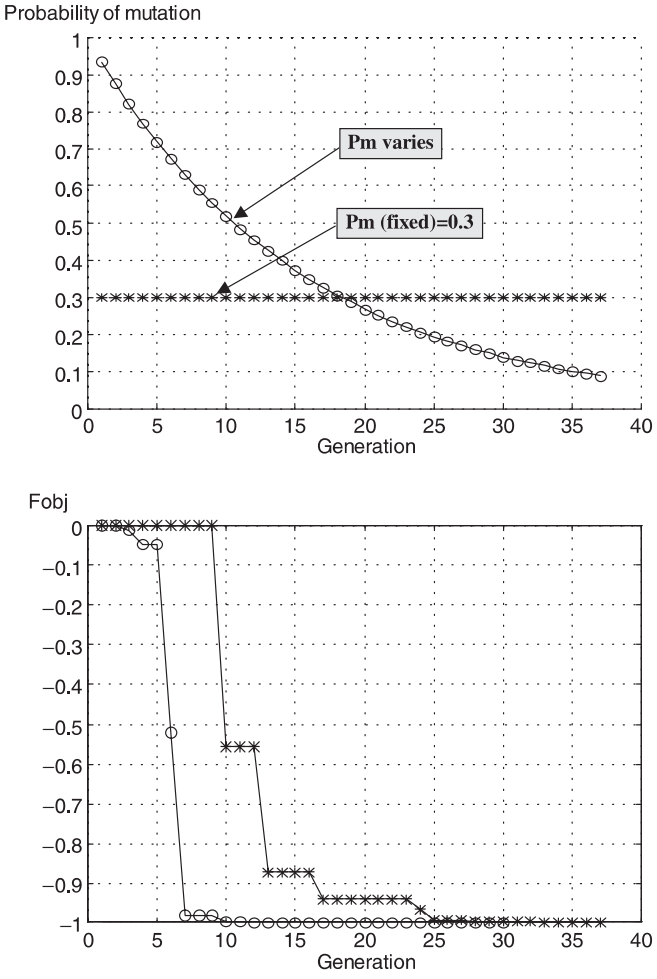


Fig. 6 Convergence of the algorithm, in function of the probability of mutation for the *Easom* function

a very low final rate of mutation (10^{-3}) is achieved, after a maximum number of generations,

$$\alpha = MaxGen / \log \left[\frac{P_m(0)}{10^{-3}} \right] , \quad (7)$$

where $MaxGen$ represents the maximum number of generations.
The decreasing rate used allows a wide exploration of the search space at the beginning of the algorithm, and a faster convergence at the end of the GA.
The resulting convergence of the algorithm, represented in Fig. 6, shows that the performances, relating both to the speed and to the accuracy of the final solution, are better in the case where the probability of mutation is variable throughout the successive generations than if it is kept constant.

- **Generate 'nb_pop' sub-populations**
- **Evaluate**
- ***Repeat***
- **for each sub-population**

  1. **Selection**
  2. **Crossover**
  3. **Mutation**
  4. **Evaluation**
  5. **Replacement**

- ***As long as* generation < MaxGen**

Fig. 7 Second step of the algorithm

3.5 Structure of the algorithm

After having generated the initial subpopulations, as described above, the algorithm enters its second step. During this process, the algorithm applies the genetic operators in order to determine the portion of the search space D which is likely to contain the global minimum. The stopping criterion used is the maximum number of generations $MaxGen$, as stated in Fig. 7.

At the end of this step, the best individual of each subpopulation is determined. Then the best chromosome s^* is extracted among all the best individuals previously found. The point s^* is used to initialize the genetic algorithm involved in the third step of the algorithm.

Once the promising zone thus determined, we perform an *"intensification"* inside this zone to refine the solution s^*. For that purpose, we use a GA with the same operators that previously, but handling only one population. At the beginning of the computation, this population is generated around s^* inside a neighbourhood V, which is progressively reduced throughout the evolution of the GA. So, at each generation, a new population is generated around the best point found (see Fig. 8). The decreasing law relating to the neighbourhood is the same that the one used for the mutation probability

$$V(k) = V(0) \times \exp(-k/\alpha) , \quad (8)$$

where $V(0)$ stands for the initial neighbourhood; α is calculated so that the final neighbourhood is small enough (10^{-2} in our case),

$$\alpha = MaxGen / \log \left[\frac{V(0)}{10^{-2}} \right] \quad (9)$$

($MaxGen$ is the maximum number of generations during the last step of the algorithm).

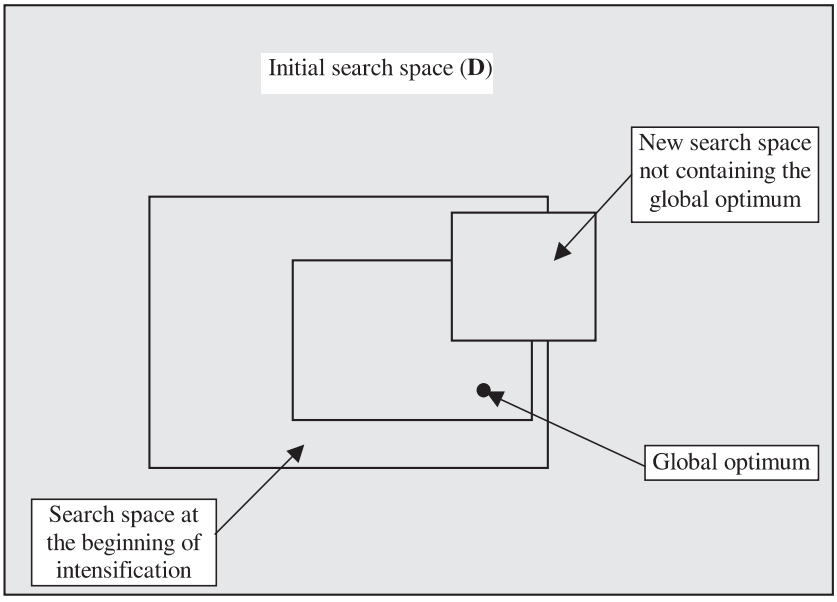
This decreasing rate depends on the shape of the function to optimize. Indeed, for the functions of which shape is "complicated", that is to say comprising several local minima, a too fast decrease (Fig. 9a) of the neighbourhood may eliminate the global optimum from the search

- ***Repeat***

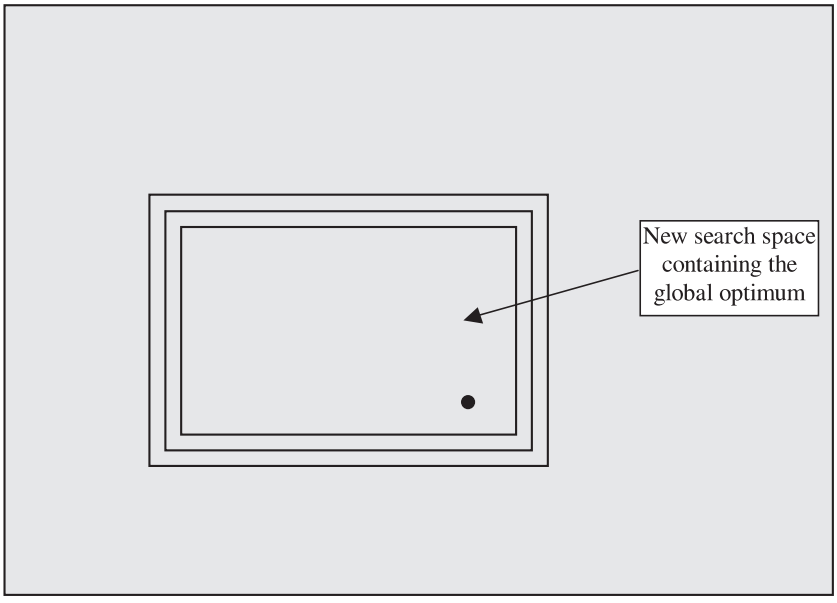
1. Generate one population around s* in a neighborhood V(k)
2. Evaluation of the function for each chromosome of the population
3. Selection
4. Crossover
5. Mutation
6. Evaluation
7. Update of s* by the best point found
8. Reduction of the neighborhood

- ***Until* a given stopping criterion.**

Fig. 8 Last step of the algorithm



(a) Fast reduction of the neighborhood : no convergence.



(b) Slow reduction of the neighborhood : convergence of the algorithm.

Fig. 9 Influence of the reduction speed of the neighbourhood on the convergence of the algorithm. (a) Fast reduction of the neighbourhood: no convergence; (b) slow reduction of the neighbourhood: convergence of the algorithm

domain, preventing the convergence of the algorithm towards the wished solution. For instance, in Fig. 9a, the global optimum was "lost" after three generations. In this case, it is necessary to have a slower decrease of the neighbourhood (Fig. 9b), allowing a better convergence, but to the detriment of the CPU time (number of evaluations of the objective function). For instance, in Fig. 9b, the global optimum is present after three generations; nevertheless, several generations are still necessary to achieve the convergence around that optimum.

3.5.1 Stopping criteria of the algorithm

The stopping criterion generally used is the maximum number of generations, set at the beginning of the algorithm. We retained this idea to control the GA used at the second stage of the algorithm. But, for the last stage, as the number of generations necessary for the convergence of the algorithm depends on the shape of the previously determined promising zone, this criterion is

not well adapted. Indeed, the search for the optimum of a "simple" function, having no or few local minima, requires less generations than a "complicated" function, for which a meticulous exploration of the search domain is necessary. In consequence, we adopted the following alternative criteria.

- Improvement of the solution: after a given number of generations $NbAm$, which depends on the dimension of the objective function, if the algorithm does not progress, that is to say if the solution s^* is not "improved" (from the viewpoint of the value of the objective function), intensification isstopped.
- Accuracy of the solution: like for the first criterion, if, after a given number of generations $NbPr$, the norm of the difference between the solution s^* at the gth generation and the solution got at the generation $g - NbPr$ is smaller than a given threshold (typically 10^{-4}), the last step of the algorithm (intensification) is stopped.

These various parameters are controlled in an empirical way; the quality of the final solution depends on this tuning. Typically the maximum numbers of successive iterations without any improvement of the objective function value, $NbAm$ and $NbPr$, are set respectively to four and two times the dimension of the objective function. Intensification being the last stage of the algorithm, its end corresponds to the end of the algorithm.

4 Experimentation

4.1 Results from our GA

To test our algorithm, several analytical test functions, listed in the Appendix, were used. This set of functions includes some functions having the following features:

- continuous/discontinuous
- convex/nonconvex
- unimodal/multimodal
- quadratic/non quadratic
- low dimension/high dimension

A good convergence of the algorithm requires the tuning of some of its characteristic parameters (that tuning being valid for all the handled functions).

- The number of subpopulations *nb_pop*: after several tests, this parameter was set to five.
- The number of chromosomes per subpopulation M: by proceeding in the same way, this number was set to ten.
- The maximum number of generations $MaxGen$, used at the second step of the algorithm, was set to ten.
- We fixed the crossover probability at its classical value of 0.6 (Goldberg 1989).

Table 1 Results obtained for 20 test functions

Test function	Rate of success	Average NBEVAL	AVEERROR
MZ	100	452	3.6×10^{-4}
DJ(F2)	100	449	4.7×10^{-3}
GP	100	270	10^{-9}
R2	100	596	10^{-12}
ES	100	642	3.42×10^{-9}
RC	100	490	2.55×10^{-3}
Z_2	100	437	10^{-10}
SH	100	946	6.16×10^{-4}
BH	100	493	4.62×10^{-11}
$H_{3,4}$	100	324	7.1×10^{-3}
DJ(F1)	100	395	5.7×10^{-4}
$S_{4,5}$	62	1158	1.2×10^{-3}
$S_{4,7}$	70	1143	1.4×10^{-4}
$S_{4,10}$	58	1235	4.23×10^{-3}
DJ(F3)	100	540	0
R_5	60	4150	1.1×10^{-1}
Z_5	100	1115	9.3×10^{-4}
$H_{6,4}$	100	937	3×10^{-2}
R_{10}	70	8100	10^{-1}
Z_{10}	100	2190	3.3×10^{-3}

The efficiency of the algorithm is quantified by

- the rate of success, which represents the rate of executions leading to the global optimum of the handled function, among a given number of executions of the algorithm;
- the number of evaluations of the objective function *NbEval* at the end of the execution; and
- the standard deviation *AveError* between the obtained solution and the exact value of the global optimum of the function.

Because of the stochastic nature of genetic algorithms, the discussion of results derived from one single execution of the algorithm is meaningless. So, all the results reported in this paper are obtained by averaging the results from 100 executions per function.

The results given in Table 1 show that the global optimum generally is reached, since the ratio of success is equal to 100% for the majority of the tested functions. The splitting of the initial population into subpopulations and the diversity of these ones contributes a lot in the obtained results. Indeed, the subpopulations well cover the variation domain D at the beginning of the algorithm, when the solution s^* is improved only occasionally. The purpose of this step is not to find the global optimum but mainly to find the promising zone, conditioning the convergence of the GA used at the last step of the algorithm. We also notice in Table 1 a strong correlation between the dimension of the function, the rate of success and the number of evaluations of the objective function. This last one tends to increase when the dimension increases.

4.2 Comparison with other competitive algorithms

We compared the results derived from our GA with the results of some published methods, see Table 2. Among these, simulated annealing and tabu search can be pointed out. Two variants of tabu search are CRTSmin and CRTSave (Battiti and Tecchiolli 1996). They they differ from one another in the value of the objective function used in the neighbourhood considered to generate a new point; for the first method, the algorithm uses the

Table 2 List of the methods used for the comparison

Method	Reference
our algorithm (RCGA)	this paper
continuous genetic algorithm (CGA)	Chelouah and Siarry (2000a)
enhanced continuous tabu search (ECTS)	Chelouah and Siarry (2000b)
continuous reactive tabu search (CRTS min)	Battiti and Tecchiolli (1996)
continuous reactive tabu search (CRTS ave)	Battiti and Tecchiolli (1996)
taboo search (TS)	Cvijovic and Klinowski (1995)
enhanced simulated annealing (ESA)	Siarry *et al.* (1997)
INTEROPT	Bilbro and Snyder (1991)

Table 3 Comparison of average numbers of objective function evaluations, achieved with 8 algorithms to optimize 13 functions

Function/Method	RCGA	CGA	ECTS	CRTSmin	CRTSave	TS	ESA	INTEROPT
GP	**270**	410	231	171	248	486	783	6375
R_2	**596**	960	480	–	–	–	796	–
ES	**642**	1504	–	–	–	–	–	–
RC	**490**	620	245	41	38	492	–	4172
Z_2	**437**	620	195	–	–	–	15 820	–
SH	**946**	575	370	–	–	727	–	–
$H_{3,4}$	**324**	582	548	609	513	508	698	1113
$S_{4,5}$	**1158**	610	825	664	812	–	1137	3700
$S_{4,7}$	**1143**	680	910	871	960	–	1223	2426
$S_{4,10}$	**1235**	650	898	693	921	–	1189	3463
R_5	**4150**	3990	2142	–	–	–	5364	–
Z_5	**1115**	1350	2254	–	–	–	69 799	–
$H_{6,4}$	**937**	976	1520	1245	750	2845	2638	17 262

minimal value, for the second one, it uses the average value.

Table 3 collects the results obtained for 13 functions of which dimension is lower than 6, using 8 different methods. We give, for each function, the average number of evaluations of the function, calculated over 100 tests. We point out that the 13 functions were not used by all the authors.

This comparison shows that the results of our algorithm are similar and sometimes better than those provided by the other methods. When the average number of evaluations of our algorithm is superior to the one obtained with some other method, that additional computational cost is acceptable, because it allows to achieve a better rate of success.

5 Conclusions

The proposed algorithm shows good performances for continuous variable optimization of benchmark functions. Our main contributions lie in the appropriate segmentation of the initial population into subpopulations, obtained by using the notion of entropy to evaluate the "information" contained in each subpopulation. So, at the beginning of the algorithm, several subpopulations are handled in parallel, in order to reduce the search space for the GA used in the last stage of the algorithm. During these two steps, the mutation probability is not kept constant, but varies according to a decreasing law, what allows a better exploration of the search space. The same decreasing law is used for the reduction of the search space during intensification.

The main improvements of our algorithm over the simple genetic algorithm SGA (Goldberg 1989) are as follows.

- SGA uses binary coding whereas we use real-valued representation.
- The search space is not kept constant throughout the exploration of our algorithm, but is progressively reduced to surround more efficiently the global optimum.

The method is suitable for solving highly multimodal problems. It can be applied in various engineering fields, such as image processing, robotics and circuit design.

References

Axelsson, J. 1993: Turbine preliminary design using genetic algorithms. *Technical Report LiTH-IDA-Ex-9302*, Linköping University, Sweden

Battiti, R.; Tecchiolli, G. 1996: The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. *Annals Oper. Res.* **63**, 53–188

Bilbro, G.L.; Snyder, W.E. 1991: Optimization of functions with many minima. *IEEE Trans. Systems, Man, Cyber.* **21**, 840–849

Chelouah, R.; Siarry, P. 2000a: A continuous genetic algorithm designed for the global optimization of multimodal functions. *J. Heuristics* **6**

Chelouah, R.; Siarry, P. 2000b: Tabu search applied to global optimization. *Eur. J. Oper. Res., Special Issue on Combinatorial Optimization* **123**, 30–44

Chun, J.S.; Kim, M.K.; Jung, H.K. 1997: Shape optimization of electromagnetic devices using immune algorithm. *IEEE Trans. Magnetics* **33**, 1876–1879

Cvijovic, D.; Klinowski, J. 1995: Taboo search. An approach to the multiple minima problem. *Science* **667**, 664–666

Davis, T.E.; Principe, J.C. 1991: A simulated annealing like convergence for the simple genetic algorithm. *Proc. 4th Int. Conf. on Genetic Algorithms ICGA*, pp. 174–182

De Jong, K.A: 1975: *An analysis of the behaviour of a class of genetic adaptive systems*. Ph.D. Thesis, University of Michigan, Ann Arbor, MI

Fisher, K.A. 1993: The application of genetic algorithms to optimising the design of an engine block for low noise. *IEEE Transactions on Vehicular Technology*, **42**, 18–22

Goldberg, D.E. 1989: *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley

Laucasius, C.B.; Kateman, G. 1992: Understanding and using genetic algorithms. Part 1. Concepts, properties and context. In: *Chemometrics and Intelligent Laboratory Systems*, Vol. 19, pp. 1–31. Elsevier

Lutton, E.; Martinez, P. 1994: *Détection de primitives géométriques bidimensionnelles dans des images à l'aide de l'algorithme génétique*. Evolution Artificielle, Ed. Cepadues

Michalewicz, Z. 1996: *Genetic algorithms + data structures = evolution programs*. Berlin, Heidelberg, New York: Springer

Mitchell, M. 1996: *An introduction to genetic algorithms*. Cambridge, MA: MIT Press

Périaux, J.; Sefrioui, M.; Stoufflet, B.; Mantel, B.; Laporte, E. 1995: Robust genetic algorithms for optimizing problems in aerodynamic design. In: Winter, G.; Périaux, J.; Galán, M.; Cuesta, P. (eds.) *Genetic algorithms in engineering and computer science* (Proc. EUROGEN'95, held in Las Palmas, Spain, December) pp. 371–396. New York: John Wiley & Sons

Potgieter, E.; Stander, N. 1998: Genetic algorithm applied to stiffness maximization of laminated plates: review and comparison. *Struct. Optim.* **15**, 221–229

Proceedings of the Seventh International Conference on Genetic Algorithms. Michigan State University, East Lansing, MI, Morgan Kaufmann

Siarry, P.; Berthiau, G.; Durbin, F.; Haussy, J. 1997: Enhanced simulated annealing for globally minimizing functions of many continuous variables. *ACM Trans. Math. Software* **23**, 209–228

Vavak, F.; Fogarty, T.C.; Jukes, K. 1995: Application of the genetic algorithm for load balancing in the process industry. *Proc. 1st Int. Mendelian Conf. on Genetic Algorithms* pp. 159–164. PC-DIR Publishing

Wardlaw, R.; Sharif, M. 1999: Evaluation of genetic algorithms for optimal reservoir system operation. *J. Water Resources Planning Management* **125**, 25–33

Appendix: list of test functions used

Michalewicz (MZ) (n variables):

$$MZ(x_i) = -\sum_{i=1}^{n} \sin(x_i) \cdot \left[\sin(i \cdot (x_i)^2/\pi)\right]^{2m} , \qquad (10)$$

$m = 10$, search domain: $-\pi \le x_i \le \pi$, $j = 1, n$;
$n = 2$, one global minimum: $(x_1, x_2)^* = (2.25,\ 1.57)$;
$MZ[(x_1, x_2)^*] = -1.80$.

De Joung F2 (2 variables):

$$F2(x_1, x_2) = 100 \times (x_2^2 - x_1) + (1 - x_1) , \qquad (11)$$

search domain: $-2.048 \le x_j \le 2.048$, $j = 1, 2$;
1 minimum (local and global): $(x_1, x_2)^* = (0, 0)$;
$F2[(x_1, x_2)^*] = 0$.

Goldstein and Price (GP) (2 variables):

$$\begin{aligned} GP(x_1, x_2) = &\left[1 + (x_1 + x_2 + 1)^2 \times (19 - 14 \times x_1 + \right. \\ &\left. 13 \times x_1^2 - 14 \times x_2 + 6 \times x_1 \times x_2 + 3 \times x_2^2)\right] \times \\ &\left[30 + (2 \times x_1 - 3 \times x_2)^2 \times (18 - 32 \times x_1 + 12 \times \right. \\ &\left. x_1^2 - 48 \times x_2 - 36 \times x_1 \times x_2 + 27 \times x_2^2)\right] \end{aligned} \qquad (12)$$

search domain: $-2 \le x_j \le 2$, $j = 1,\ 2$; 4 local minima;
one global minimum: $(x_1, x_2)^* = (-1, 0)$;
$\mathrm{GP}[(x_1, x_2)^*] = 3$.

Rosenbrock (R_n) (n variables):

$$R_n(x) = \sum_{j=1}^{n-1} \left[100 \left(x_j^2 - x_{j+1}\right)^2 + (x_j - 1)^2\right] . \qquad (13)$$

Three functions are used: R_2, R_5 and R_{10};
search domain: $-5 \le x_j \le 10$, $j = 1, \ldots, n$;
global minimum: $x^* = (1, \ldots, 1)$;
$R_n(x^*) = 0$.

Easom (ES) (2 variables):

$$\begin{aligned} ES(x_1, x_2) = &-\cos(x_1) \times \cos(x_2) \times \\ &\exp\left\{-\left[(x_1 - \pi)^2 + (x_2 - \pi)^2\right]\right\} , \end{aligned} \qquad (14)$$

search domain: $-100 \le x_j \le 100$, $j = 1, 2$;
one global minimum: $(x_1, x_2)^* = (\pi, \pi)$;
$ES[(x_1, x_2)^*] = -1$.

Branin RCOS (RC) (2 variables):

$$\begin{aligned} RC(x_1, x_2) = &\left\{x_2 - \left[5/\left(4 \times \pi^2\right)\right] \cdot x_1^2 + \right. \\ &\left. (5/\pi) \times x_1 - 6\right\}^2 + \\ &10 \times \{1 - [1/(8\pi)]\} \times \cos(x_1) + 10 , \end{aligned} \qquad (15)$$

search domain: $-5 \le x_1 \le 10, 0 \le x_2 \le 15$;
3 global minima: $(x_1, x_2)^* = (-\pi,\ 12.275)$, $(\pi,\ 2.275)$, $(9.42478, 2.475)$;
$RC[(x_1, x_2)^*] = 0.397887$.

Zakharov (Zn) (n variables):

$$\begin{aligned} Z_n(x) = &\left(\sum_{j=1}^{n} x_j^2\right) + \left(\sum_{j=1}^{n} 0.5j \times x_j\right)^2 + \\ &\left(\sum_{j=1}^{n} 0.5j \times x_j\right)^4 , \end{aligned} \qquad (16)$$

three functions are used Z_2, Z_5 and Z_{10};
search domain: $-5 \le x_j \le 10$, $j = 1, \ldots, n$;
global minimum: $x^* = (0, \ldots, 0)$;
$Z_n(x^*) = 0$.

Shubert (SH) (2 variables):

$$\begin{aligned} SH(x_1, x_2) = &\sum_{j=1}^{5} j \times \cos[(j+1) \times x_1 + j] \times \\ &\sum_{j=1}^{5} j \times \cos[(j+1) \times x_2 + j] , \end{aligned} \qquad (17)$$

search domain: $-10 \le x_j \le 10$, $j = 1, 2$;
760 local minima;
18 global minima: $SH[(x_1, x_2)^*] = -186.7309$.

Bohachevsky (BH) (2 variables):

$$\begin{aligned} BH(x_1, x_2) = &x_1^2 + 2 \times x_2^2 - 0.3 \times \\ &\cos(3\pi \times x_1) \times \cos(4\pi \times x_2) + 0.3 \end{aligned} \qquad (18)$$

search domain: $-50 \le x_j \le 50$, $j = 1, 2$;
one global minimum: $(x_1, x_2)^* = (0, 0)$;
$BH[(x_1, x_2)^*] = 0$.

Hartmann ($H_{3,4}$) (3 variables):

$$H_{3,4}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right] , \qquad (19)$$

(see the parameter values in Table 4)

search domain: $0 \le x_j \le 1$, $j = 1, 3$;
4 global minima: $p_i = (p_{i1}, p_{i2}, p_{i3}) = i$th local minimum (approximation);
$f[(p_i)] \cong -c_i$. One global minimum: $x^* = (0.11, 0.555, 0.855)$;
$H_{3,4}(x^*) = -3.86278$.

Table 4 Hartmann ($H_{3,4}$) function parameter values

i	a_{ij}			c_i	p_{ij}		
1	3	10	30	1.0	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3.0	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.0381	0.5743	0.8828

*De Joung F*1 *(DJ)* (3 variables):

$$F1(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2\,, \tag{20}$$

search domain: $-5.12 \le xj \le 5.12$, $j = 1, 3$;
one minimum (local and global): $(x_1, x_2, x_3)^*$=(0, 0, 0);
$DJ((x_1, x_2, x_3)^*) = 0$.

*De Joung F*3 *(DJ)* (5 variables):

$$F3(x_i) = \sum_{i=1}^{5} \text{integer value}\,(x_i)\,, \tag{21}$$

search domain: $-5.12 \le xj \le 5.12$, $j = 1, \ldots\ 5$;
1 minimum (local and global): $-5.12 \le x_i \le -5$;
$\mathrm{DJ}[(x_i)^*] = -30$.

Shekel ($S_{4,n}$) (4 variables):

$$S_{4,n}(x) = -\sum_{i=1}^{n} \left[(x - a_i)^T (x - a_i) + c_i\right]^{-1}\,, \tag{22}$$

(see the parameter values in Table 5)
$x = (x_1, x_2, x_3, x_4)^T$, $a_i = (a_{i1}, a_{i2}, a_{i3}, a_{i4})^T$.

Three functions are used: $S_{4,5}$, $S_{4,7}$ and $S_{4,10}$;
search domain: $0 \le x_j \le 10$, $j = 1, \ldots, 4$;
10 local minima: $a_i^T = i$th local minimum (approximation): $S_{4,n}[(a_i^T)] \cong -1/c_i$. One global minimum: $x^* = (4, 4, 4, 4)$;
$S_{4,n}(x^*) = -10.40294$.

Table 5 Shekel ($S_{4,n}$) function parameter values

i	a_i^T				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Hartmann ($H_{6,4}$) (6 variables):

$$H_{6,4}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]\,, \tag{23}$$

(see the parameter values in Table 6)
search domain: $0 \le x_j \le 1$, $j = 1, 6$;
4 local minima: $p_i = (p_{i1}, \ldots, p_{i6}) = i$th local minimum (approximation);
$f[(p_i)] \cong -c_i$.

Table 6 Hartmann ($H_{6,4}$) function parameter values

i	a_{ij}						c_i	p_{ij}					
1	10.0	3.00	17.0	3.50	1.70	8.00	1.0	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10.0	17.0	0.10	8.00	14.0	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3.00	3.50	1.70	10.0	17.0	8.00	3.0	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	17.0	8.00	0.05	10.0	0.10	14.0	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381