ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/5589211

Hybrid methods using genetic algorithms for global optimization. IEEE Trans Syst Man Cybern B Cybern 26(2): 243-258

Article *in* IEEE TRANSACTIONS ON CYBERNETICS · February 1996

DOI: 10.1109/3477.485836 · Source: PubMed

CITATIONS
340

READS
301

2 authors:

Jean-Michel Renders
Xerox Corporation
159 PUBLICATIONS **1,937** CITATIONS
SEE PROFILE

Stéphane P Flasse
Independent Researcher
36 PUBLICATIONS **2,643** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Cross-modal Image Processing View project

All content following this page was uploaded by Stéphane P Flasse on 06 April 2017.

The user has requested enhancement of the downloaded file. All in-text references underlined in blue are added to the original document and are linked to publications on ResearchGate, letting you access and read them immediately.

Hybrid Methods Using Genetic Algorithms for Global Optimization

Jean-Michel Renders and Stéphane P. Flasse

Abstract—**This paper discusses the trade-off between accuracy, reliability and computing time in global optimization. Particular compromises provided by traditional methods (Quasi–Newton and Nelder–Mead's Simplex methods) and Genetic Algorithms are addressed and illustrated by a particular application in the field of nonlinear system identification. Subsequently, new hybrid methods are designed, combining principles from Genetic Algorithms and "hill-climbing" methods in order to find a better compromise to the trade-off. Inspired by biology and especially by the manner in which living beings adapt themselves to their environment, these hybrid methods involve two interwoven levels of optimization, namely Evolution (Genetic Algorithms) and Individual Learning (Quasi–Newton), which cooperate in a global process of optimization. One of these hybrid methods appears to join the group of state-of-the-art global optimization methods: it combines the reliability properties of the Genetic Algorithms with the accuracy of Quasi–Newton method, while requiring a computation time only slightly higher than the latter.**

I. Introduction

GENETIC Algorithms have recently emerged as an increasingly popular family of methods for global optimization [1], [2]. These methods perform a search by evolving a population of candidate solutions through the use of nondeterministic operators and by improving incrementally the individuals forming the population by mechanisms inspired from those of genetics (e.g., crossover and mutation). They are known to offer significant advantages over traditional methods by using simultaneously several search principles and heuristics whose most important ones are: a population-wide search, a continuous balance between exploitation (convergence) and exploration (maintained diversity) and the principle of building-block combination. In certain cases, particularly when facing complex optimization problems with numerous local optima, where traditional optimization methods fail to provide efficiently reliable results, Genetic Algorithms can constitute an interesting alternative. Nevertheless, Genetic Algorithms can suffer from excessively slow convergence before providing an accurate solution because of their fundamental requirement of using minimal *a priori* knowledge and not exploiting local information. This kind of "blindness" may prevent them from being really of practical interest for a lot of applications.

Manuscript received June 23, 1993; revised December 10, 1994.

J.-M. Renders is with the Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium.

S. P. Flasse is with the Environmental Sciences Group, Natural Resources Institute, Kent ME4 4TB, UK (e-mail: nri@ukc.ac.uk).

Publisher Item Identifier S 1083-4419(96)02306-0.

On the other hand, classical "hill-climbing" methods, such as the Quasi–Newton method, are well known to exploit all local information in an efficient way, provided that certain conditions are fulfilled and, in particular, that the function to be minimized is "well-conditioned" in the neighborhood of the unique optimum [3]. Such a high level of exploitation requires a lot of local information to be known (gradient and, sometimes, Hessian matrix): the more intensive the exploitation, the stronger the need of specialized information about the function to be minimized. Moreover, if the basic requirements are not satisfied, the reliability of the "hill-climbing" method is greatly jeopardized.

At this stage of consideration, it is useful to keep in mind the fundamental conflict between accuracy, reliability and computation time when searching for the global optimum of complex problems, especially for problems with many local optima: it is generally impossible to reach accurately and reliably the global optimum in a short computation time. This conflict is closely related to the "exploitation-exploration" trade-off. Each optimization method represents a particular compromise or, in other words, a particular way to manage the fundamental conflict. As far as Genetic Algorithms are concerned, despite incontestable advantages and original principles that they implement (especially, the ability to "rough out" a problem reliably by finding the most promising regions of the entire search space), they often represent an unsatisfactory compromise, because they suffer from a certain inefficiency, characterized by a slow convergence and a lack of accuracy when an exact solution is required. Complementary, "hill-climbing" methods appear to realize another "extreme compromise" to solve the conflict: they focus solely on accuracy and computation time (exploitation) to the detriment of reliability. The aim of this work is to investigate how to overcome the limitations inherent to both classes of methods and to design hybrid methods bringing together the benefits of the single ones: the hybrid methods we will propose combine the reliability properties of the Genetic Algorithms and their original search heuristics with the accuracy of "specialized" hill-climbing methods, while requiring a computation time only slightly higher than the latter (particularly when parallel computers are used).

This paper discusses the trade-off between accuracy, reliability and computing time, examines the particular compromises provided by traditional methods (Quasi–Newton [4] and Nelder–Mead's Simplex [5] methods) and Genetic Algorithms, and offers new hybrid methods combining principles from Genetic Algorithms and "hill-climbing" methods in order to

1083–4419/96$05.00 © 1996 IEEE

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

find a better compromise to the trade-off. Inspired by biology and especially by the manner in which living beings adapt themselves to their environment, these hybrid methods involve two interwoven levels of optimization, namely Evolution (Genetic Algorithms) and Individual Learning (Quasi–Newton), which cooperate in a global process of optimization.

To illustrate these theoretical considerations, a particular application is chosen in the field of nonlinear system identification. System identification with Genetic Algorithms alone has already been proposed by [6], focusing on linear dynamic systems in view of their control. Genetic Algorithms have also been developed for structural identification by [7], in combination with specific parameter estimation methods. The system considered in this paper is the Earth's surface as seen by satellite; the identification problem consists of retrieving surface properties from satellite measurements [8], [9]. Such a problem, which can be seen as an "inversion problem," usually requires the minimization of functions with complex landscapes. This system is flexible enough to exhibit a large class of behaviors (local minima, nearly-flat valley, ...) by modifying several parameters (number of measurement data, physical parameters of the system, ...), while being sufficiently representative of a large class of problems in nonlinear identification. In the first part of the paper, we apply traditional methods and Genetic Algorithms to this problem and we compare the methods in terms of accuracy, reliability and computation time. In the second part, we design hybrid methods to offer an optimal compromise between these three criteria in solving the identification problem. Finally, one of the new hybrid methods is compared to the "Multi-Level Single Linkage" method [10], considered as a powerful approach for global optimization, and whose principles have some similarity with those of our hybridized "biologically-inspired" methods.

II. Traditional Optimization Methods and Genetic Algorithms

A. Description of the Optimization Methods

Minima of a function f can be formally defined as follows:

Let M be the set of feasible points (i.e., possible solutions which satisfy the constraints).

$$\begin{aligned} &f(x^*) \text{ is a local minimum} \\ &\qquad \Updownarrow \\ &[\exists \epsilon > 0 | \forall x \in M: ||x - x^*|| < \epsilon \\ &\quad \Rightarrow f(x) \geq f(x^*)] \quad \text{and} \quad x^* \in M \end{aligned} \tag{1}$$

$$\begin{aligned} &f(x^*) \text{ is a global minimum} \\ &\qquad \Updownarrow \\ &\forall x \in M: f(x) \geq f(x^*) \quad \text{and} \quad x^* \in M \end{aligned} \tag{2}$$

in other words, the global minimum is the smallest of the local minima. Henceforth, we shall use the term "local minimum" only to mean a local minimum which is not a global minimum.

At least three main strategies of search can be distinguished, according to their ways of exploring and exploiting the function to be optimized. The first one (e.g., Quasi–Newton) explores the search space using a single point and exploits all the local information (in particular the gradient) to find a better next point. The second (e.g., Simplex) uses a family of points to explore the search space and exploits the relative order of the various candidate solutions to drive the future search in a better direction. Both strategies are usually known as "hill-climbing," because they use only local information to find a better solution. When they converge to a stationary point, there is no guarantee that this is in fact the global optimum. By contrast, the third strategy (e.g., Genetic Algorithms) uses a population of candidate solutions initially distributed over the whole function space and quickly identifies the subdomain in which the global minimum function is located, while maintaining constant exploration of the search space. In accordance with the convention in the literature of Genetic Algorithms, this ability is called the "explorative" property, while the ability to exploit all the local information for refining progressively and efficiently the solution is called the "exploitative" property. Based on the characteristics of each method, our prior expectation would be that the "hill-climbing" methods would show good exploitative properties, and the Genetic Algorithms good explorative properties.

1) Quasi–Newton (QN): The Quasi–Newton method, also called the *variable metric method,* is based on Newton's method, which consists, at each iteration, of the minimization of a quadratic approximation to the function. This requires that the function be twice differentiable, and that both gradient vector and Hessian matrix can be calculated at all points. The Quasi–Newton does not require the Hessian matrix to be calculated, but builds it up iteratively; it therefore only requires the first derivatives of the function [in fact, the routine adopted in this study (routine E04JAF from the NAG library [11]) approximates the first derivatives with finite differences]. The Quasi–Newton method here used carries out line minimizations, their results being used to update the current estimate of the Hessian matrix and to find further promising directions of search [4].

The termination criteria are based on the following principles. The search is halted when the relative change occurring between two successive iterations is less than some prescribed quantity for the sequence of candidate solutions and function values, or when a maximum number of function evaluations is reached.

2) Simplex or Flexible Polyhedron (S): The Simplex method [5] is a robust nonlinear multi-dimensional optimization technique. The method does not require the derivatives of the function to be optimized. A simplex is a geometrical figure consisting, in N dimensions, of $(N + 1)$ vertices. The Simplex method starts, not with a single point, but with an initial simplex ($N + 1$ points); then, through a sequence of elementary geometric transformations (reflection, contraction and extension), the initial simplex moves, expands and contracts, in such a way that it adapts itself to the function landscape and finally surrounds the optimum. For determining the appropriate transformation, the method uses only the relative order between the performances (values of the function to be optimized) of the point considered. After each transformation, the current worst point is replaced by a better

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

one; in this way, providing that certain precautions are taken, the algorithm always forces convergence of the sequence of iterates.

The termination criterion generally depends on the magnitude of the difference between the performances of the best point and the worst one: as soon as this difference is less than a pre-defined threshold, the search is halted. When the termination criterion is reached, it is customary to restart the entire procedure with a new simplex "exploded" (the point claimed to be the optimum is kept as one of the vertices of the new simplex, while the others are chosen in arbitrary orthogonal directions). Such explosions are carried out until the simplex repeatedly collapses onto the same solution. This iterative procedure allows certain anomalous terminations to be avoided and can increase the reliability of finding the global optimum.

3) Genetic Algorithms (GA): The Genetic Algorithms method [1] is an iterative search algorithm based on an analogy with the process of natural selection (Darwinism) and evolutionary genetics. The search aims to optimize a user-defined function (the function to be optimized) called the fitness function. To perform this task, GA maintains a "population" of candidate points, called "individuals," over the entire search space. At each iteration, called a "generation," a new population is created. This new generation generally consists of individuals which fit better than the previous ones into the external environment as represented by the fitness function. As the population iterates through successive generations, the individuals will in general tend toward the optimum of the fitness function. To generate a new population on the basis of a previous one, GA performs three steps: a) it evaluates the fitness score of each individual of the old population, b) it selects individuals on the basis of their fitness score, and c) it recombines these selected individuals using "genetic operators" such as mutation and crossover, which, from an algorithmic point of view, can be considered respectively as means to change locally the current solutions and to combine them.

What makes GA attractive is its ability to accumulate information about an initially unknown search space and to exploit this knowledge to guide subsequent search into useful sub-spaces. The fundamental implicit mechanism underlying this search consists of the combination of high-performance "building blocks" discovered during past trials.

Three important features distinguish the GA approach: a) GA works in parallel on a number of search points (potential solutions) and not on a unique solution, which means that the search method is not local in scope but rather global over the search space; b) GA requires from the environment only an objective function measuring the fitness score of each individual and no other information nor assumptions such as derivatives and differentiability; and c) both selection and recombination steps are performed by using probability rules rather than deterministic ones; this aims to maintain the global explorative properties of the search.

In classical GA, a fourth difference can be distinguished: GA encodes the parameters to be optimized and operates on the codes and not on the parameters themselves (cf., the genotype/phenotype distinction of genetics). The aim of coding the parameters is to transform the original optimization problem into a combinatorial one, because GA is essentially a mechanism for combinatorial search. For example, in the case of continuous numerical optimization problems, the coding of the parameter value is usually chosen as a binary coding into a chain of limited length. In this way, continuous problems are handled as discrete ones. Coding the parameters implies the design of decoders and repair algorithms, in order to ensure that the solution found by GA can be translated into a feasible solution of the original problem. In several cases, this design is not an easy task, as described in [12].

Recently, a modified version of the classical GA ("discrete GA") has been developed [2], [13] to meet the needs of industrial GA practitioners in solving real-world problems; this new version is called "real-coded GA" or "continuous GA" and is characterized by a direct floating-point representation of the parameters to be optimized, whereas a binary representation is usually adopted with classical GA, resulting in a loss of precision. This real-coded GA generally offers the advantages of being better adapted to numerical optimization for continuous problems, of speeding up the search and of making easier the development of approaches "hybridized" with other methods; but it requires the development of new "genetics-inspired" operators, and to date it lacks strong mathematical foundations, currently established only for discrete GA (the "Schema theorem" [1], deception problem related to Walsh analysis [14], and convergence theorems [15]). In our particular case study, we have adopted a real-coded GA. Appendix B provides a comparison between the two kinds of GA, describes the genetic operators used for real-coded GA (mutation and crossover), and shows that the continuous version is more convenient and more efficient for solving the inversion problem.

Several termination criteria for the search have been proposed. One simple criterion is to stop the procedure when almost all individuals are identical or nearly so; another possible criterion is to test the improvement in the best fitness score over successive generations. However, the first criterion can lead to excessive search time (too many generations before halting), while the second one is not satisfactory for functions characterized by a landscape presenting "plateau-type" regions. In this work, we decided simply to stop the search after a fixed number of generations; this number was experimentally chosen as representing a reasonable compromise between the constraints of population convergence, computing time and accuracy.

B. Case Study Description

In order to test the applicability and the accuracy of the different methods, a particular application is chosen in the field of nonlinear system identification.

The system envisaged is the Earth's surface as seen by satellite: satellite sensors measure the surface reflectance. This reflectance is directly related to surface properties that influence the scattering of the light, and to the geometry of illumination and observation (since the Earth's surface is anisotropic). In order to describe the surface by parameters

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

that correspond to the physics used in the surface description of climate models, this study case uses the physically-based model of [8], which can be formally represented as follows:

$$\rho = \rho(\alpha;\, \mathbf{x}^*) \tag{3}$$

where ρ represents the modeled measurement (reflectance), α the characteristics of the measurements, and $\mathbf{x}^*$ the properties of the surface (independent variables of the model)—bold letters represent set of parameters (vectors). In the model used in this study (see description in Appendix A), $\mathbf{x}^* \in R^4$ and $\alpha \in R^3$.

The interpretation of satellite data then consists in the retrieval of intrinsic parameters of the system ($\mathbf{x}^*$) from a set of measurements and a pre-determined structure of the model. This implies the "inversion" of (3) against the measurements (problem of "model inversion" or "parameter identification"). Because of the complexity of the model, no analytical solution is generally known for the inversion of models, and in particular for the inversion of the model adopted in this study. Therefore, some form of numerical search technique is needed. When transforming the inversion problem in an optimization problem, the choice of the criterion to be optimized depends on the characteristics of the measurement noise, as well as the modeling errors. As it may be difficult to know accurately these characteristics and, in particular, the probability distributions of these errors, choosing the correct criterion can be a delicate problem in practice. For the sake of simplicity, we suppose, in this work, that real measurements and output of the model are related by

$$\rho_k = \rho(\alpha_k;\, \mathbf{x}^*) + \epsilon_k \tag{4}$$

where ρ_k corresponds to the measurement k, ρ to the modeled value of that measurement, using its characteristics α_k and the true surface properties $\mathbf{x}^*$; ϵ_k is a Gaussian white noise with zero mean and constant variance, uncorrelated with the other ϵ, and representing measurements and modeling errors. The criterion to be minimized is then (least-square criterion)

$$\delta^2(\mathbf{x}) = \sum_{k=1}^{n} [\rho_k - \rho(\alpha_k;\, \mathbf{x})]^2 \tag{5}$$

where n is the number of measurements; $\alpha_{i,\,k}$ the characteristics of the kth measurement ($k = 1, \cdots, n$); ρ_k is the measurement k in itself and ρ is the modeled measurement calculated according to (3) using estimates $\mathbf{x}$ of the surface parameters. The domain of this function is bounded and included in R^4.

By varying the surface parameters ($\mathbf{x}^*$) and the number of measurements, it turns out that the corresponding landscape of the function δ^2 exhibits different degrees of complexity (local minima, nearly flat valley, ...).

Several inversion experiments were performed using measurements generated by (3) over three different surfaces (A, B, C). The results of the inversions will be compared to the values of the parameters representing the surfaces used to generate these measurements. These sets of parameters were chosen after study of the partial derivatives of the bidirectional function, to represent between them most of the significant features of the entire function. Of course the properties of a four-dimension domain cannot be fully expressed with 3 sets of surfaces; however, the conclusions drawn from these three surfaces are believed to hold for a wide range of surfaces.

For each surface, three types (1, 2, 3) of data sets were built, with n = 4, 16, and 64 measurement values respectively, in order to test the sensitivity of the inversion to the number of measurements. The angles were chosen in order to insure the observability of the parameters.

The number N of parameters to retrieve being 4, the minimum number of measurements required is also 4. When only 4 measurement values are available, the problem is not, strictly speaking, an optimization problem; it is rather the search for the solution of 4 nonlinear simultaneous equations in 4 unknowns, treated as an optimization problem.

These experiments were performed 50 times, with "initial guess" values taken randomly in the whole domain of the function.

The values of the parameters representing the surfaces, the angles used to create the data sets and the technical characteristics of each method are described in Appendix C.

C. Results and Discussions

1) Description of the Function δ^2: It is interesting to try to represent the different shapes of the function to be minimized and to visualize its global and local minima. Since δ^2 is a 4-dimensional function, it is not possible to have a global view of the function in a single graph. We attempt to represent the most interesting facts within 2-dimensional figures, keeping in mind that two of the four parameters are fixed and that the full 4-dimensional function could exhibit unexpected features, difficult to visualize.

A careful analysis of the inversion results and of the graphs representing the function δ^2 for two parameters—the others being fixed—leads to the observation of three phenomena: 1) The global minimum is always located in a kind of quasilinear gutter or valley, nearly parallel to one of the parameter axes. This kind of gutter presents a bottom nearly "flat" in the axial direction. If no special precautions are taken, the flatter the gutter, the more significant the effects of roundoff and truncation errors. Fig. 1(a) shows this phenomenon for data set A1, where x_3 and x_4 are displayed, x_1 and x_2 being fixed at the value of the global minimum. A close-up view near the global minimum [Fig. 1(b)], shows that a minimum really exists, but that the gradient along the gutter length is very weak. The intensity of the slope generally depends on the number of data values. 2) In the neighborhood of the global minimum the search for the solution is very sensitive to small variations in the parameters; in other words, there is a kind of "cross-sensitivity" of the function in the neighborhood of the global minimum. This is illustrated by Fig. 2, where $\widetilde{x_3}$ is the x_3 value which minimizes δ^2 for each x_4 value, x_1 and x_2 being fixed, x_3^* and x_4^* are the values of x_3 and x_4, respectively, which correspond to the global minimum of δ^2. This graph shows that the "cross-sensitivity" is characterized

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

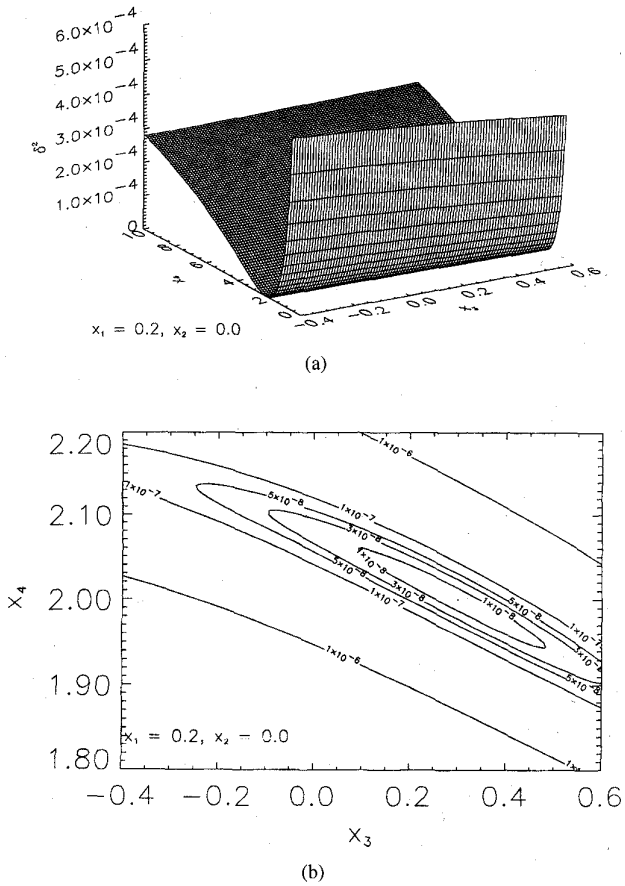


Fig. 1. Representation of the function δ^2 for data set A1. (a) Landscape with x_1 and x_2 fixed. (b) Contour representation around the global minimum.

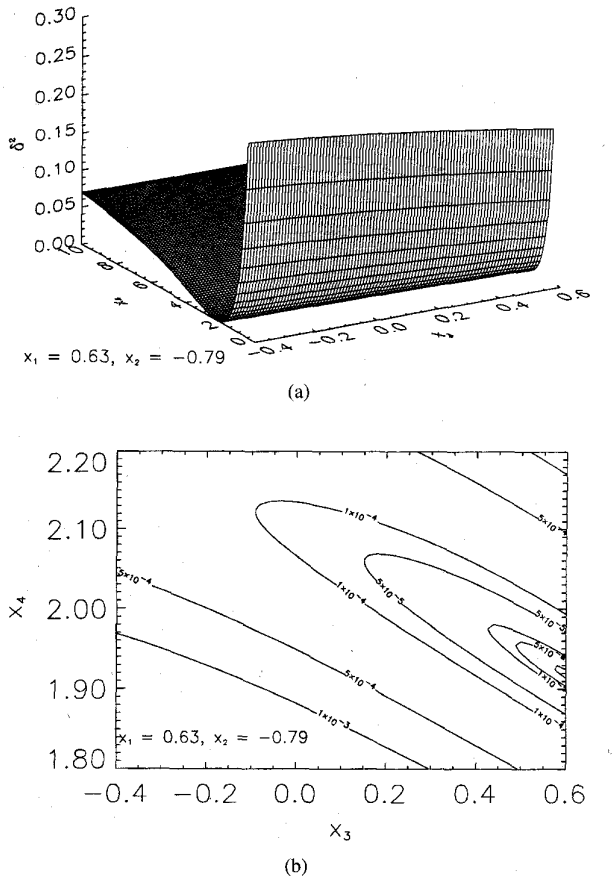


Fig. 3. Representation of the function δ^2 for data set C1. (a) Landscape with x_1 and x_2 fixed. (b) Contour representation around the global minimum.

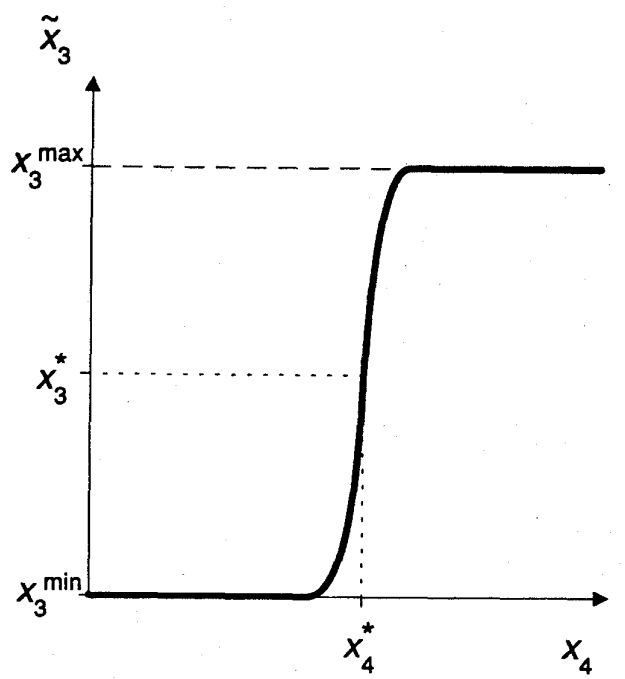


Fig. 2. Illustration of the "cross-sensitivity" phenomenon.

by a very high variability around the global minimum, such that a small variation in x_4 induces a large variation in $\widehat{x_3}$. This "cross-sensitivity" occurs for the other parameters as well: Fig. 3, for example, shows that the function δ^2, for data set C1 and with x_1 and x_2 fixed at a value near (but not exactly on) the optimal value, presents a gutter [Fig. 3(a)] where the bottom slightly slopes toward the boundary [Fig. 3(b)]. 3) Local minima can occur. They either form a "horizontal gutter" along a bound of the function domain, or are isolated points inside it.

Since the function δ^2 appears to be complex and "ill-behaved," the optimal solution will not be easy to find and, with standard termination criteria, optimization procedures can result in solutions either stuck far from the real solution for one or two parameters [phenomena (1) and (2)], or located a long way from it [phenomenon (3)]. Therefore, it is necessary to use a strong and powerful method in order to ensure the correct retrieval of the parameters.

We observed that the number of data values influences very much the shape of the function δ^2. A larger number of values always increases the gradient in all directions around the global minimum and decreases the "cross-sensitivity" in its neighborhood; therefore the effects of phenomena (1) and (2) are less significant. With a higher number of values, the function δ^2 "curls up," placing the minimum in a kind of

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

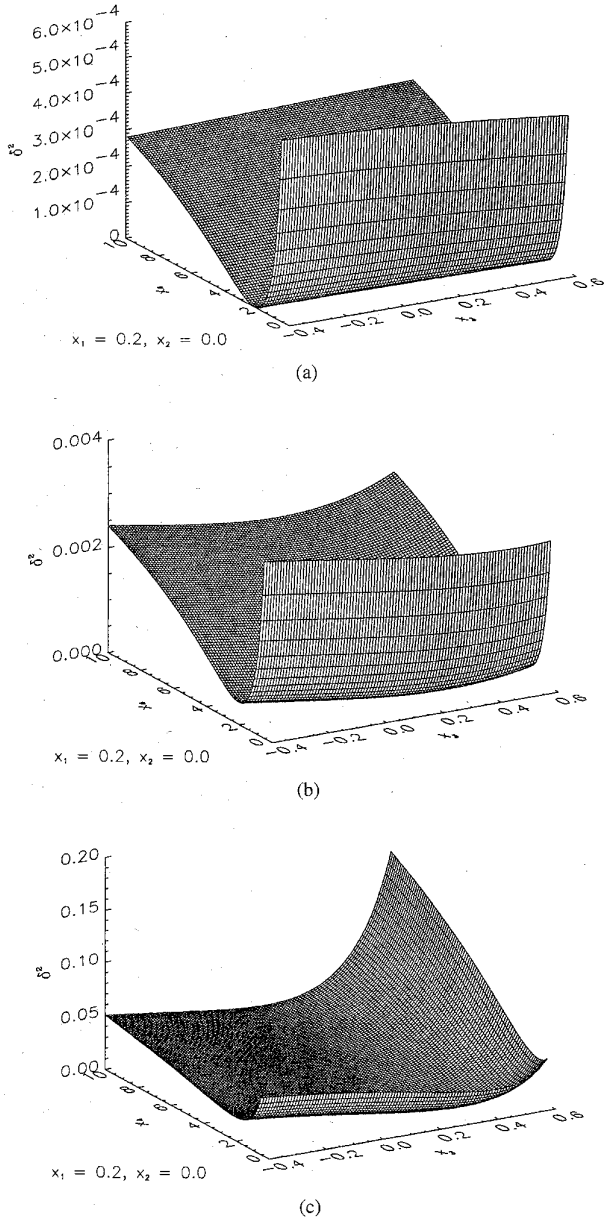


Fig. 4. Representation of the function δ^2 for data set A1 (A), A2 (B), A3 (C).

"bowl" rather than in a gutter, as seen in Fig. 4 for data sets A. As far as phenomenon (3) is concerned, we observed that with a larger number of values local minima present less of a problem.

Different types of surface generally produce a function δ^2 of the same global shape. However, they may display different sensitivity to the above phenomena. For example, the gutter, as seen in case A1, becomes larger and flatter in width in case B1, while case C1 shows more "cross-sensitivity."

2) Comparison Criteria: We compare the methods under four different headings: convergence, reliability, accuracy, and computation time.

Convergence: Convergence is defined in theory as the existence of a finite limit for the sequence of solutions (or iterates). In practice, this is assessed by observing the magnitude of the change over the last few terms of the sequence. This does not guarantee convergence to a stationary point. The problem of insuring real convergence on an optimum before stopping a numerical search is very tricky, particularly in the problem discussed here. The criterion of stopping the search when the difference between successive values of the function is sufficiently small is not always satisfactory, because the search halts prematurely for "gently-sloping" functions, such as the function δ^2 [phenomenon (1)]. On the other hand, the alternative criterion proposed—stopping the search when the distance moved in the parameter space during the last few iterations is less than the location accuracy required—can give rise to similar difficulties [3]. We mentioned above the usual termination criteria associated with each of the methods (QN, S, and GA). In this study, we decided to adopt these usual criteria without modification or special precaution.

Reliability: The reliability of a method is its ability to find the global minimum and therefore to avoid local minima. In our study, only the solutions found outside the gutter including the global minimum [phenomenon (3)] were considered as failure for the method. This is easily assessed using the error on the first two parameters, since these errors are very small when the solution is in the gutter, compared to those for solutions outside the gutter. The reliability will be characterized by the percentage of searches successful out of the 50 experiments.

Accuracy: The accuracy corresponds to the distance from the solution to the global minimum, for cases not considered as failure. In particular, this accuracy involves the effects of phenomena (1) and (2). It is assessed by the error defined as:

$$\text{ERROR} = \sqrt{\sum_{i=1}^{4} \left(\frac{x_i - x_i^*}{w_i} \right)^2} \tag{6}$$

where x_i^* and x_i, $i = 1, \cdots, 4$ are the values of the four parameters for the real minimum and the inversion solution respectively, w_i is the weight given for each parameter, corresponding to the range of the domain of each parameter (w_i = 1, 2, 1, 10 for parameters x_1, x_2, x_3, x_4, respectively), in such a way that an identical relative error on each parameter gives an identical absolute contribution to the global ERROR. The accuracy of the method will be characterized by the mean of the error over the successful experiments.

Computation Time: Evaluating the function to be minimized constitutes the greatest part of the computation time. Therefore, we assess the computation time by the number of calls of the function ρ (3). Since we perform the search over several experiments which differ only in their "initial guess," we consider the mean number of calls over the number of successful experiments. Moreover, in order to make the results comparable between the different data sets (n = 4, 16, and 64), we have normalized the mean computation time by the number of measurements. In this way, the normalized computation time corresponds to the average number of calls of the function δ^2.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

3) Methods Comparison: We discuss here only the experiments with "clean data sets" and initial guesses taken at random over the whole domain of the function.

Convergence: Although it is clear that convergence is not a very good indicator for comparing methods, it is important to ensure that the methods converge. In the implementation adopted, GA, S, and QN guarantee that the value of the function δ^2 will not increase because, in the iterations, the new "best point" is always chosen to be strictly better than or equal to the previous one. As a consequence the methods necessarily converge. However, premature termination is not necessarily avoided.

The usual termination criteria turn out to be unsatisfactory for the inversion with small data sets (4 values), i.e., when phenomena (1) and (2) are the most significant. For these cases, premature termination in the gutter brings poor accuracy. Refinements in the criteria, for example by re-scaling the function on the gutter region ("zoom"), should be studied in order to ensure a satisfactory behavior of the search for the problematic cases. This point is currently under investigation.

Reliability: Fig. 5(a) shows that, as expected, GA offers a very good exploration of the search space and is 100% secure in finding the global minimum. The "hill-climbers" (QN and S) have more difficulties in the search for the global minimum. For example, the reliability for data sets 1 varies between 45% and 95%. However, the larger the number of data values, the more reliable the result. Moreover, it seems that S is more reliable than QN for a small number of data values (4 values), whereas QN is better than S with a larger number (16 and 64).

Accuracy: Fig. 5(b) shows that, whatever the method, the more data values available the higher the accuracy. The error never reaches a value better than 0.12 with data sets 1 but can reach an error of 10^{-6} with data sets 3. For a small number of data values, no method shows systematic advantages over the others. However, when the number of data values increases, GA never reaches a high level of accuracy, whereas QN and S do.

We observed that most of the error is carried by the parameter x_3. This is the gutter phenomenon. When the number of data values increases, the error on χ_3 diminishes and the parameter x_4 becomes the least well defined. This behavior is equally visible in all methods, the error on a single parameter being inherent in the function landscape [cf., phenomenon (1) in 5.1].

Computation Time: QN is generally at least twice as fast as S, while GA takes far more computation time [Fig. 5(c)]; as discussed above, the computation time allowed for GA was chosen as a reasonable compromise between the constraints of population convergence, computing time and accuracy. The differences between the computation time of the different methods are basically related to the use of the information acquired during the search. QN uses quantitative information about the neighborhood of the current (single) point, particularly by computing the derivatives, whereas S only uses the order relations between the points of the current simplex. GA uses also the order relations between the points of individuals of the current population; however, its search is performed

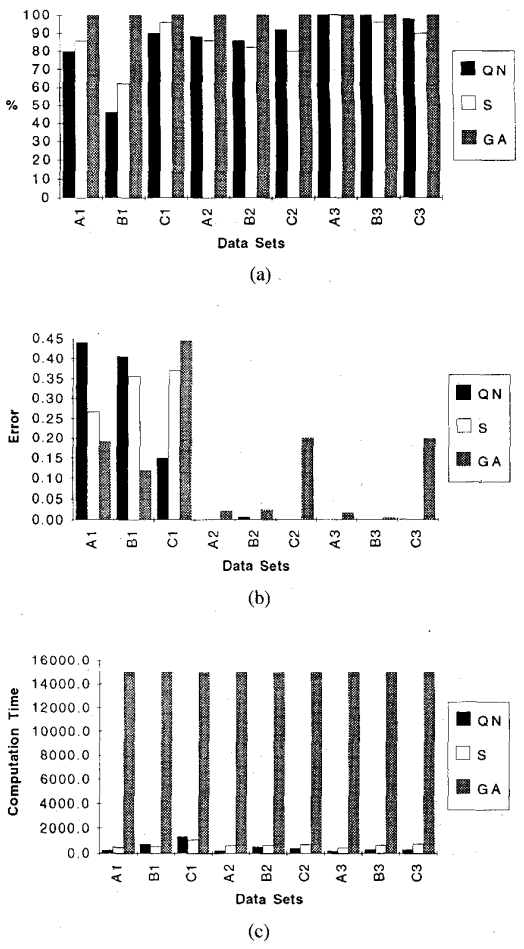


Fig. 5. Comparison of the methods QN, S, and GA. (a) Reliability. (b) Accuracy. (c) Normalized computation time.

partially randomly and may include bad intermediate solutions that a "hill-climbing" method would not have explored.

The convergence of GA occurs during the first hundred generations. Afterwards, the δ^2 value (fitness) of the solution continues to decrease slightly, but this does not necessarily imply that the accuracy of the solution improves; this is related to the shape of the function landscape.

Conclusion: GA requires a lot of computation time and yields poor accuracy, but seems to be absolutely reliable. As expected, GA turns out to be very good for exploration. QN on the other hand has the advantage of being very accurate provided it is not stuck in a trap, and this accuracy is reached in a relatively short time. QN shows good exploitative properties. S falls in between GA and QN, but its performances remain nearer those of QN because both use the "hill-climbing" principle. All in all, it turns out that the results of the optimization procedures agree with our prior expectations, taking into account the function landscape.

Finally, tests have shown that adding noise to the data only moves the global minimum from the expected one, and does not interfere with the search for a minimum in itself. Since measurements are contaminated by noise due to the physical characteristics of instruments and to the conditions of

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

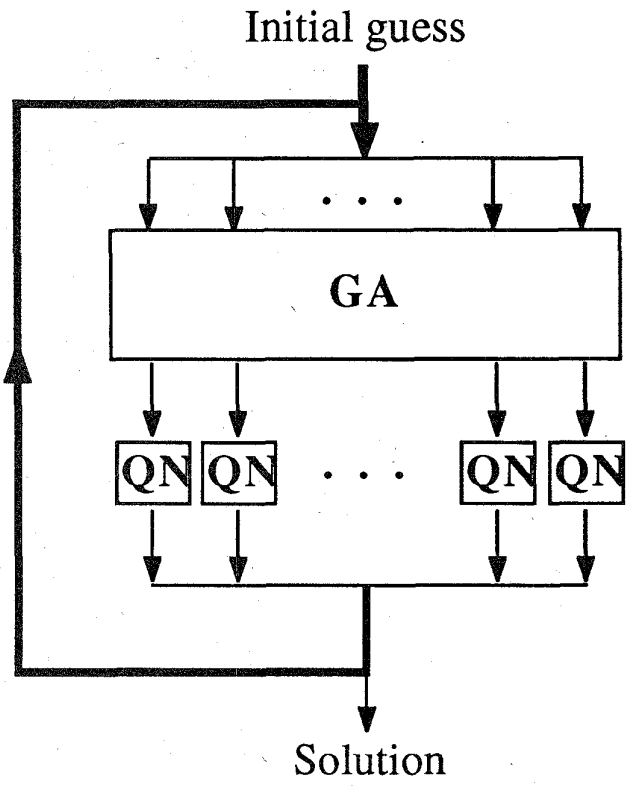


Fig. 6. Schematic representation of hybrid methods principles.

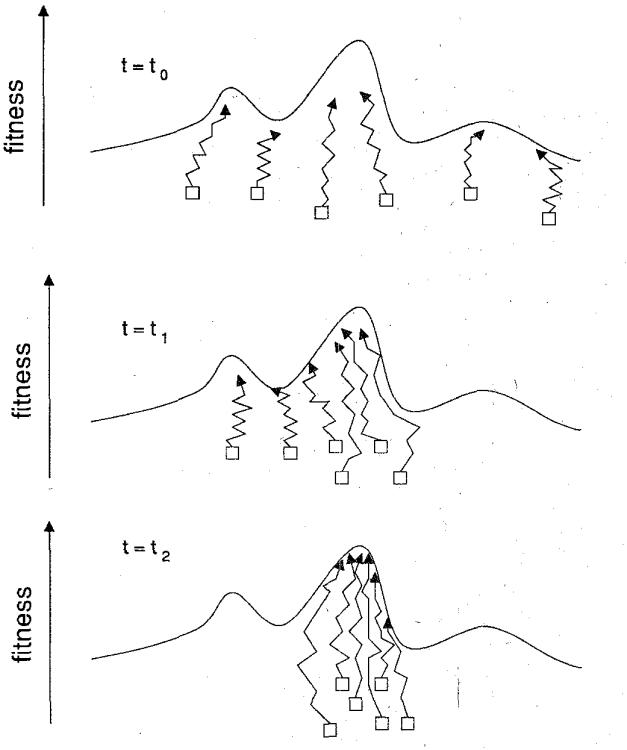


Fig. 7. Typical evolution to the population on the fitness landscape for the Darwin-inspired combination of real-coded Genetic Algorithms and Quasi–Newton (GQD). Symbol "X" indicates the nondeveloped individual (at birth), while the winding array represents the life of this individual (optimization with QN). Individuals selected for reproduction are marked by a dashed circle.

observation, further investigation should address the question of what is the maximum amount of noise that can be tolerated in the data before the global minimum becomes unacceptably different from the expected one, rendering the inversion procedure meaningless.

III. Hybrid Methods

The idea is to combine two of the previous methods in order to merge their advantages and reduce their disadvantages. Similar ideas have already been introduced in neural network learning [16] and in engineering design optimization; for this last application, Powell has developed an "interdigitized" technique combining Expert Systems, numerical optimization and Genetic Algorithms to provide an answer to engineering design problems [17].

We investigated a coupling of GA and QN, using both the exploration capabilities, parallelism and combination properties of GA, and the exploitation power of QN. For this purpose, we "interweaved" the two methods, relating the use of GA to the concept of "evolution" of a population of individuals and that of QN to the concept of "life" for each individual. In general, GA takes a population and makes it evolve in such a way that most of the population reaches the global minimum. With pure GA, the transition from one generation to another is based on the selection of individuals evaluated at their "birth" by instantaneous fitness. With hybrid methods, the principle is to let GA make its selections based on the fitness at the end of the individual life, the life being performed by QN (Fig. 6). Other couplings would have been possible (e.g., GA and S), but we chose to restrict our study to the most promising coupling, i.e., the one consisting of 2 methods with very different principles and complementary properties.

We tried two hybrid methods inspired by the evolutionary theories [18] of Darwin and Lamarck, respectively.

A. Darwin-Inspired Combination of Real-Coded Genetic Algorithms and Quasi–Newton (GQD)

Genetics-Inspired Description (Fig. 7): The natural selection of individuals depends on their fitness after living, maturing and adapting to their environment. The genetic material used during reproduction is the genetic code of the individual which is inborn and never changes during its life. The population gradually comes to be composed of genetic material producing optimal adult individuals.

Algorithmic Description: Iteratively, GA gradually produces better initial guesses for QN algorithms (these could work in parallel), in the sense that the QN starting point comes to be one which, being located squarely within the basin of attraction of the global minimum, will easily and reliably evolve to this optimum. When evaluating the fitness of each individual, GA uses the results of QN methods working with an initial guess corresponding to this individual. During reproduction and genetic transformations (cross-over, mutation) for the production of the individuals of the next generation, GA works on the initial guesses previously given to QN and not on the solutions provided by it.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

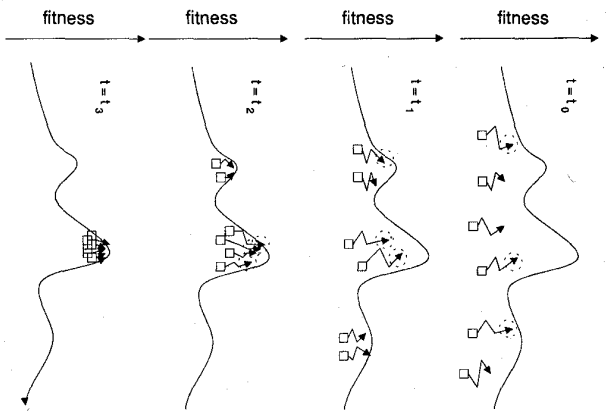


Fig. 8. Typical evolution to the population on the fitness landscape for the Lamarck-inspired combination of real-coded Genetic Algorithms and Quasi–Newton (GQL). Symbol "X" indicates the nondeveloped individual (at birth), while the winding array represents the life of this individual (optimization with QN). Individuals selected for reproduction are marked by a dashed circle.

B. Lamarck-Inspired Combination of Real-Coded Genetic Algorithms and Quasi–Newton (GQL)

Genetics-Inspired Description (Fig. 8): Here again, the natural selection of individuals depends on their fitness after living, maturing and adapting in their environment. But in this case, the features acquired by the organism during its life can be hereditary transmitted by sexual reproduction (Lamarckism). This is as if the genetic code of the individual changed during its life, with the genetic material used in reproduction being the modified genetic code of the mature organism.

Algorithmic Description: Iteratively, GA gradually produces better solutions using QN as an "accelerator" mechanism thanks to its exploitative properties. When evaluating the fitness of each individual, GA uses the results of QN methods working with an initial guess corresponding to this individual. During reproduction and genetic transformation (cross-over, mutation) for the production of the individuals of the next generation, GA works on the new solution (termination point) found by QN (in contrast to the GQD method, in which GA works on the initial points provided to QN).

Recently, other researchers have investigated the benefits of adding "Lamarckian" principles to GA for solving graph partitioning problems [19] and competitive, multi-agent control problems [20]. They have found that, in some cases, it can result in a significant enhancement of performance.

C. Termination of QN "Life"

For both GQD and GQL methods, when evaluating the individuals with QN, it is not necessary to reach one of the usual termination criteria of the method: individual optimization (life) can be performed over a limited number of steps, because the main part of the information given by the search with QN is acquired during the first few steps and because the search is pursued and refined over the next generations anyway, especially for the GQL method. GQD, however, needs a longer "QN-life" time than GQL because, at each generation, the information acquired during the individual's life is not directly used when generating new individuals (lower level of exploitation); in other words, GQD will require a more accurate fitness evaluation and consequently a longer "QN-life" time so that the selection mechanism remains meaningful.

In practice, the combined methods terminated with an "extended-life," in which the best individual of the last GA generation was exploited by QN using the normal termination criteria in order to give the best accuracy for the final solution. We found experimentally that limiting the maximum number of function evaluations of the QN procedure to 20 for GQL and 115 for GQD (instead of 1615 under the normal criterion) gave good results for the function δ^2.[1]

D. Comparison of Interweaving Methods with Darwin and Lamarck Principles

Technical details concerning the implementation of both methods are given in Appendix C. The interweaving methods always converge because they use a combination of methods that converge separately. The reliability is generally excellent for both methods, even if GQL presents a 6% failure rate in the most difficult case [Fig. 9(a)]. The accuracy of both methods is of the same order of magnitude, linked to the QN accuracy [Fig. 9(b)]. As expected because of the construction of the methods, GQL works faster than GQD by a factor of 6 [Fig. 9(c)].

In the GQD method, a shorter QN-life would not preserve the reliability, even though the computing time would be globally reduced. This can be explained by the fact that the evaluation of an individual with a shorter QN-life is no longer completely meaningful. Indeed, if a too short life time is adopted, individuals which offer a faster decrease of the δ^2 function during their life are favored for further selection, regardless of the minimum value that they would have reached if a complete local convergence was allowed. As GA works on the initial points provided to QN, and not on the termination point, no further refinement or correction of the information used for the individual evaluation can be expected and successive selection are more likely to produce results corresponding to some nonglobal minimum. In the GQL method, a longer QN-life would of course improve the quality of the function evaluation and therefore increase the reliability, but to the detriment of the computation time. We believe that the current choice of GA and QN parameters in the GQD and GQL methods, for these case studies, seems to result in the more satisfactory compromise.

All in all, GQD and GQL give nearly the same results, the only significant difference being the computation time. The principal advantage of GQL is that the convergence can be made very fast, because each generation considers individuals which have already evolved toward the solution.

[1] These numbers include an initialization cost of 15 function evaluations involved in starting up the QN method with a finite difference approximation to the initial Hessian. Since a QN-step requires a minimum of $N + 1$ function evaluations, the 20 function evaluations of GQL correspond to the initialization stage and one QN-step, while the 115 function evaluations of GQD correspond to the initialization stage and at most 20 QN-steps.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

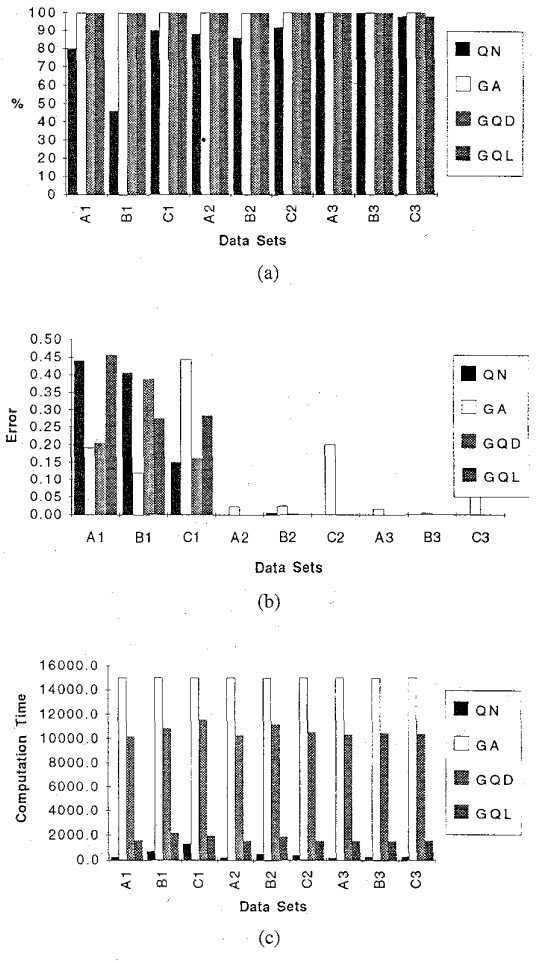


Fig. 9. Comparison of the methods QN, GA, GQD, and GQL. (a) Reliability. (b) Accuracy. (c) Normalized computation time.

For the function δ^2, it was found possible to combine this rapid convergence (strong exploitative mechanisms) with a high reliability; such a result might not hold for a different function landscape, e.g., one with many local minima. As noted above GQD needed much longer "QN lives" for its individuals than GQL. It may be that for certain function landscapes, the slower convergence of GQD (considering the total number of function evaluations) would be accompanied by greater reliability. Nevertheless, GQL seems to be quite adequate for the function δ^2.

E. Comparison of Interweaving Methods with Non Hybrid Methods

The comparison of the interweaving methods with nonhybrid methods shows the power of the former. They offer a success rate very close to 100%, and always greater than or equal to that of other methods. The accuracy of the results is generally very similar to that of QN. In terms of computation time, interweaving methods are faster than GA; they are slower than QN and S, but offer the security of finding the absolute minimum. GQL offers such a security using only 4 times more computation time than the "hill-climbing" methods. In most cases, these levels of security and accuracy cannot be attained simultaneously by taking the best results of 4 parallel "hill-climbing" methods. It must be taken into account that with the use of massively-parallel computers for this application, the computation time could easily be reduced by a factor corresponding to the population size of GA (15 in our case); in these conditions, GQL could be nearly as fast as QN.

For the completeness of the comparison, we investigated also a direct combination between GA and QN: in the first stage, GA would explore the search space in order to discover useful subspaces and to provide a solution, not located precisely at the global optimum, but at least within its basin of attraction; in the second stage, QN would use the candidate solution provided by GA as an initial guess, and pursue the search in its own exploitative way. Such a combination was demonstrated to have some merit, in that a finer initial guess provided to QN improves both the reliability and the accuracy of the method. However, in order to guarantee that GA will provide a well-located starting point, a substantial number of generations (comparable with that of GA alone) is necessary and, therefore, a substantial computation time is required. It can be concluded that the direct combination method is feasible, but does not compete with the GQL method for this last reason.

F. Comparison of GQL with a Multi-Level Approach for Global Optimization

Finally, we compared our best interweaving method (GQL) with the "Multi-level single linkage" method (MLSL) [10] (see [21] for a parallel implementation of the algorithm). This method, briefly described hereafter, is considered as a powerful approaches for global optimization, offering excellent computational performance and providing probabilistic guarantees to find the global minimum based on strong theoretical properties.

The MLSL-method is a stochastic iterative algorithm, combining random search with local minimization; it presents therefore some similarities with our hybridized methods. Basically, as in GQL and GQD, each iteration of the MLSL method consists of three steps: generation of new individuals, selection, and local search (cf., QN-life). However, the way to generate and to select new individuals is different. MLSL generates new individuals randomly following a uniform distribution over the search space, without trying to accumulate and to exploit information to guide subsequent search in the most promising subspaces. MLSL then selects individuals (i.e., starting points for local search) using a geometrical criterion: an individual x is chosen for local minimization if it has not been already chosen during a previous iteration and if there is no individual within a critical distance $r(k)$ of x with a lower function value [$r(k)$ is a decreasing function of the iteration number k]. The local minimizations can be performed by any standard "hill-climbing" method and, in particular, by a QN method. The termination criterion is a Bayesian stopping rule, based on an estimate of the total number of local minima and the portion of the search space covered by the regions of attraction of the local minima found so far. Moreover, under conditions usually satisfied, both the accuracy

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

and the efficiency of the algorithm are guaranteed in a strong probabilistic sense.

We applied MLSL as follows: 1) The selection of starting points was performed without sample reduction. 2) The local minimizations were performed with the QN method. For the reasons discussed in Section III-C, we limited the maximum number of function evaluations in order to save computation time. The precise characteristics of the implementation are given in Appendix C.

In some of our case studies, where the number of local minima is too large (possibly infinite), the Bayesian stopping rule may require an unrealistic number of sample points and iterations. Therefore, we chose not to adopt the standard stopping rule, but to stop the search when the number of evaluations of the δ^2 function is approximately the same as for GQL. It must be borne in mind that there is no longer a probabilistic guarantee of finding the global minimum, as it is no longer possible to reach the theoretical number of sample points required. However, this allows us to compare the reliability and the accuracy of both methods for comparable amounts of computing time. Given the difference in the numbers of sample points (individuals) involved, and the need to operate in complete generations, the computation times required are not identical—see Fig. 10(c).

Fig. 10(a) and (b) show that both methods present equivalent performances in terms of reliability as well as accuracy, but their optimal performance is obtained for different situations (surfaces and number of measurements). Our experiments confirm that MLSL may fail in finding the global minimum [Fig. 10(a)], all the more so since the standard stopping rule is not applied. It must be kept in mind that there is no theoretical guarantee that GQL will find the global minimum.

We expected GQL to produce better results than MLSL because GQL offers the advantage to exploit the information accumulated during previous generation to guide subsequent search into promising regions, while MLSL is, to some extent, blind. However, experimental results show that this expected superiority is not systematically obtained. We believe that this may be due to the limited number of generations which does not allow to take full advantage of the "guided search" property of the Genetic Algorithms in GQL, principally obtained by the successive application of the cross-over and selection mechanisms.

MLSL would require many more sample points to improve the reliability. In this case, the selection phase would require a significant amount of computing time, no longer negligible compared to the time needed for performing function evaluations. It would follow that our definition of computing time, based on the number of function evaluations, would have to be refined, in order to reflect this extra computing load. This computing load increases with the square of the total number of points already generated up to iteration k, if a straightforward sequential implementation of the algorithm is used—although it is theoretically possible to render it linear in the number of sample points by certain search techniques. On the other hand, the selection procedure of GQL is independent of the total number of individuals generated up to generation k, and is far faster. These arguments suggest that GQL might show significant advantage over MLSL when a large number of sample points are needed; however, such theoretical considerations need to be confirmed experimentally.

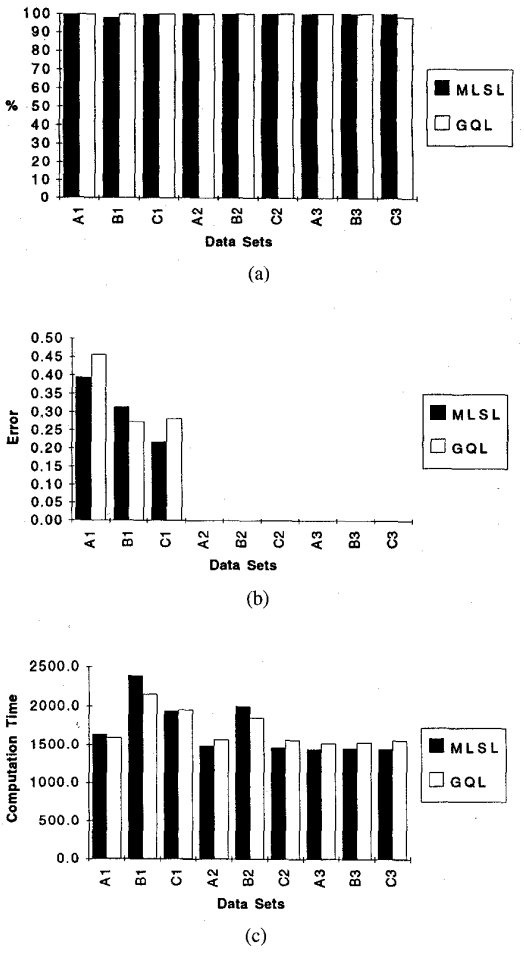


Fig. 10. Comparison of the methods GQL and MLSL. (a) Reliability. (b) Accuracy. (c) Normalized computation time.

IV. Conclusions and Further Investigations

This study emphasized the trade-off between accuracy, reliability, and computation time in global optimization. This can be seen as the impossibility of reaching accurately and reliably the global minimum in a short time. This trade-off is closely related to the balance "exploration-exploitation." Experimental comparisons showed how traditional methods (Quasi–Newton, Simplex) and Genetic Algorithms can be seen as particular instances of this trade-off.

Subsequently, the analysis of results of comparisons gave natural insights on how to produce an optimal compromise by interweaving the methods, taking the best advantage of both exploration and exploitation techniques, with a reasonable computation time. Several interweaving combinations were examined. One of these (the combination between Genetic Algorithms and Quasi–Newton method, with a selection scheme based on Lamarckian theory, GQL) appeared to join the group of state-of-the-art global optimization methods: GQL

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

combines the reliability properties of the Genetic Algorithms with the accuracy of Quasi–Newton method, while requiring a computation time only slightly higher than the latter (particularly when parallel computers are used).

There are still difficulties which are not overcome by the proposed methods. In particular, ways for improving the accuracy of the optimization by avoiding premature termination and by adapting termination criteria must be investigated.

Beyond the particular problem tackled in this study (optimization of the function δ^2), we believe that the structure of the Hybrid Algorithms presented in this paper, designed using a combination of Quasi–Newton methods and Genetic Algorithms, could be applied to solving a wider range of nonlinear least-squares identification problems and to optimizing a broader class of functions, by providing a more efficient tool for finding the global optimum. For the particular function landscape, only the parameters of the Hybrid Algorithms (crossover and mutation rates, population size, "Quasi–Newton life" time, etc.) would of course have to be tuned in order to give optimal performances.

Appendix A
Description of the Bidirectional Reflectance Model

This study case uses the physically-based model of [8] which represents the satellite measurements in terms of surface properties. They have derived an analytical expression for the bidirectional reflectance from physical and geometrical considerations of the transfer of radiation through a porous medium. Their model represents the bidirectional reflectance ρ of a homogeneous and semi-infinite canopy illuminated by the Sun from a direction (θ_1, ϕ_1), observed from a direction (θ_2, ϕ_2). In order to facilitate the retrieval of parameters through an inversion procedure, [9] have developed a simpler and parametric version where the reflectance is normalized with respect to the reflectance of a perfectly reflecting Lambertian surface under the same conditions of illumination and observation, using the significant parameters of the full theoretical model described in [8]. This model is given by

$$\rho(\theta_1, \phi_1; \theta_2, \phi_2) = \frac{\omega}{4} \frac{\kappa_1}{\kappa_1 \mu_2 + \kappa_2 \mu_1} \cdot \left[P_v(G) P(g) + H \frac{\mu_1}{\kappa_1} H \frac{\mu_2}{\kappa_2} - 1 \right] \quad (7)$$

where

$$\mu_1 = \cos \theta_1$$
$$\mu_2 = \cos \theta_2$$
$$\cos g = \cos \theta_1 \cos \theta_2 + \sin \theta_1 \sin \theta_2 \cos(\phi_1 - \phi_2)$$
$$\kappa_i(\mu_i) = \Psi_1 + \Psi_2 \mu_i \quad i = 1, 2$$
$$\Psi_1 = 0.5 - 0.6333\chi_l - 0.33\chi_l^2$$
$$\Psi_2 = 0.877(1 - 2\Psi_1)$$
$$P_v(G) = 1 + \frac{1}{1 + V_p(G)}$$
$$V_p(G) = 4\left(1 - \frac{4}{3\pi}\right) \frac{G}{2r\Lambda} \frac{\mu_2}{\kappa_2}$$
$$G = [\tan^2 \theta_1 + \tan^2 \theta_2 - 2 \tan \theta_1 \cdot \tan \theta_2 \cos(\phi_1 - \phi_2)]^{1/2}$$
$$P(g) = \frac{1 - \Theta^2}{[1 + \Theta^2 - 2\Theta \cos(\pi - g)]^{3/2}}$$
$$H(x) = \frac{1 + x}{1 + x\sqrt{(1 - \omega)}}.$$

In these equations, ω is the average single scattering albedo of the particles making up the surface; κ_1 and κ_2 describe the scatterer orientation distribution for the illumination and viewing angles, respectively; χ_l is a function of scatterer angle distribution; $P_v(G)$ is the parametric phase function that accounts for the "hotspot" effect; r is the radius of the sun flecks on the inclined leaf and Λ is the scatterer area density; $P(g)$ is the average phase function; Θ is the asymmetry factor; and the term $H(\mu_1/\kappa_1)H(\mu_1/\kappa_2) - 1$ approximates the contribution from multiple scattering.

For each pair of angles in the illumination and viewing directions, the model represents the way the incoming radiation is scattered by the surface. Therefore the bidirectional reflectance model defines the surface characteristics only by the characteristics that influence the transmission of light, and the model can formally be represented by the following relation:

$$\rho = \rho(\alpha_1, \alpha_2, \alpha_3, \alpha_4; x_1, x_2, x_3, x_4) \quad (8)$$

where ρ represents the measurement, α_i the characteristics $(\theta_1, \theta_2, \phi_1, \phi_2)$ of the measurements and x_j the properties of the surface $(\omega, \Theta, \chi_l, r\Lambda$, the independent variables of the model).

Appendix B
Comparison of Discrete (Binary-Coded) and Continuous (Real-Coded) Genetic Algorithms

Details about the implementation of both algorithms can be found in [1] and [13]. We illustrate here the mutation and crossover operators used with our "real-coded version" of GA.

Mutation: Let $\mathbf{x} = (x_1, x_2, x_3, \cdots, x_n)$ be the individual about to be mutated; for real-coded GA, each gene represents one particular (un-coded) parameter of the problem and n is the number of parameters to be optimized. Each gene will undergo a modification, that will be significant during the first generations, and then gradually decreasing as the search is going further. For a generation t, two numbers are randomly generated: the first one (p) is either -1 (negative change), or $+1$ (positive change) with equal probability; the second one (r), which determines indirectly the amplitude of the modification, is a number taken randomly in the interval [0; 1] with a uniform distribution. The mutated gene is given by [13]:

$$x'_k = x_k + (x_{\max} - x_k)\{1 - r^{[1-(t/T)]^b}\} \quad \text{if } p = +1 \quad (9)$$
$$x'_k = x_k - (x_k - x_{\min})\{1 - r^{[1-(t/T)]^b}\} \quad \text{if } p = -1 \quad (10)$$

where $x_{\min}$ and $x_{\max}$ designate, respectively, the upper and lower bounds of the value of x_k; b is the decrease rate of mutation ($b = 5$ in all our experiments); T is the index of the generation for which the mutation amplitude goes to zero (the

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

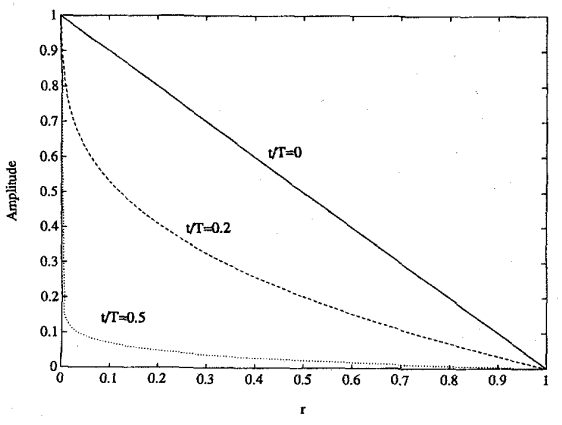


Fig. 11. Distribution of the mutation amplitude (t/T = relative generation index).

later generations will no longer be mutated). Fig. 11 gives the mutation amplitude with respect to the random number r for different values of the ratio (t/T) and for $b = 5$. Of course, the gradual decreasing of the mutation amplitude, which is similar to the temperature decrease schedule in the "Simulated Annealing" method, has a poor biological plausibility.

Crossover: We define here 2 types of crossover operators. The first one, applied with probability p_1, builds a child by randomly taking the gene of one or the other parent: for instance, $(a_1, a_2, a_3, a_4, a_5, a_6)$ and $(b_1, b_2, b_3, b_4, b_5, b_6)$ could give $(a_1, a_2, b_3, a_4, b_5, a_6)$ and $(b_1, b_2, a_3, b_4, a_5, b_6)$ as children. This kind of crossover is called "discrete crossover," because it is similar the operator used in classical discrete GA (uniform crossover). The second type of crossover, called "continuous crossover," is applied to a pair of parents with probability p_2 and performs an "averaging" operation on the genes of both parents; for instance, the above-mentioned parents could give birth to the following two children: $[a_1, a_2, (a_3 + b_3)/2, a_4, (a_5 + b_5)/2, (a_6 + b_6)/2]$ and $[b_1, b_2, (a_3 + b_3)/2, b_4, (a_5 + b_5)/2, (a_6 + b_6)/2]$. Of course, it is possible (and often useful) to design other kinds of crossover operators: for example, a complete continuous crossover, which would produce a child by (convex) linear combination of both parents, or even a "biased" continuous crossover which would result in a child situated preferentially in the neighborhood of the best parent. Fig. 12 illustrates the 4 types of crossover, showing for a given pair of parents the potential children (without taking into account the mutation effect). After all, we could imagine yet other combination mechanisms: for instance, one could take much more than 2 parents to give birth to children following the rules of the Simplex method, therefore using the local relative fitness values to direct the recombination. This could speed up the search efficiently, because operators based on local exploitation are used during the reproduction. But, once again, one faces the exploration/exploitation trade-off and the optimal compromise depends strongly on the particular problem to be solved.

There already are several studies [13] recommending the floating point representation, for the following reasons: 1) the convenience of the correspondence "one gene/one variable," 2) avoidance of problems related to the artificial character of the binary coding (e.g., Hammingclifs), 3) higher accuracy (especially with large domains), and 4) faster convergence.

We have compared the two techniques for solving the inversion problem, with the same "clean" data sets as in the experiments described in the body of this report. Fig. 13 shows that the use of binary-coded GA does not offer the same level of reliability and accuracy as the real-coded GA, while requiring a higher computation time. One explanation of these poor performances is the effect of the discretization of the search space combined with the phenomena described in 5.1; to avoid such a problem, the binary coding would have required prohibitively long representation and therefore excessive computation time.

All things considered, the preference given to the real-coded version of GA seems to be justified in the case of this inversion problem.

APPENDIX C
EXPERIMENTATION CHARACTERISTICS

A. Technical Characteristics of the Optimization Methods

1) Quasi–Newton: The Quasi–Newton algorithm used is the routine E04JAF from the Numerical Algorithm Group (NAG) Library. This routine approximates the first derivatives with finite differences.

The maximum number of function evaluations is fixed at 1615 in the routine E04JAF.

2) Simplex: The meaning of the parameters and details about the implementation are given in

- Number of vertices of the Simplex: 5.
- Simplex termination criterion (before reinitializing search by exploding the Simplex): $|\delta_{\max} - \delta_{\min}| < 10^{-7}$, where $\delta_{\max}$ and $\delta_{\min}$ are, respectively, the minimum and maximum value of the δ function over all vertices of the current Simplex.
- Explosion length (normalized): 0.1 in each direction.
- Explosion termination criterion: $|x_{k+1} - x_k| < 10^{-4}$, where x_k is the best candidate solution (normalized) found after a complete run of the Simplex following the kth explosion.
- Constraint handling: a new objective function is introduced, defined by: $\tilde{f}(x) = f(x) + Kd(x)$, where K is an arbitrary high positive number, $f(x)$ is the function to be optimized and d is the distance from the set of feasible solutions ($d = 0$ if the point is feasible).
- Candidate solution normalized $(0 - 1)$

3) Binary-Coded Genetic Algorithms: The meaning of the parameters and details about the implementation are given in [1].

- Chromosome length: 40 (bits).
- Population size: 80.
- Probability of mutation: 0.01 (per bit).

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

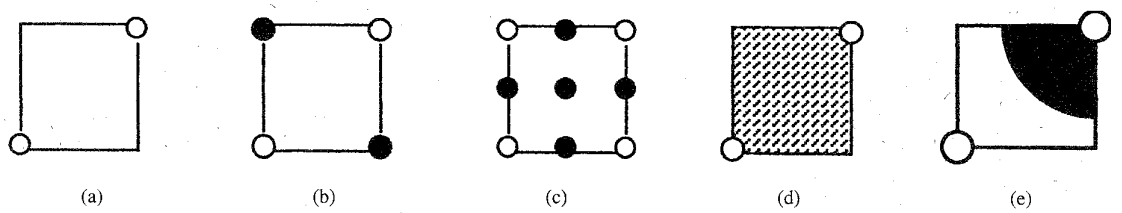


Fig. 12. Types of crossover mechanisms for "real-coded" GA. (a) Original pair of parents (white circles). (b) Children generated by discrete crossover (white and black circles). (c) Children generated by continuous crossover (white and black circles). (d) Children generated by complete continuous crossover (hatched area). (e) Children generated by biased continuous crossover (area in the neighborhood of the best parent).

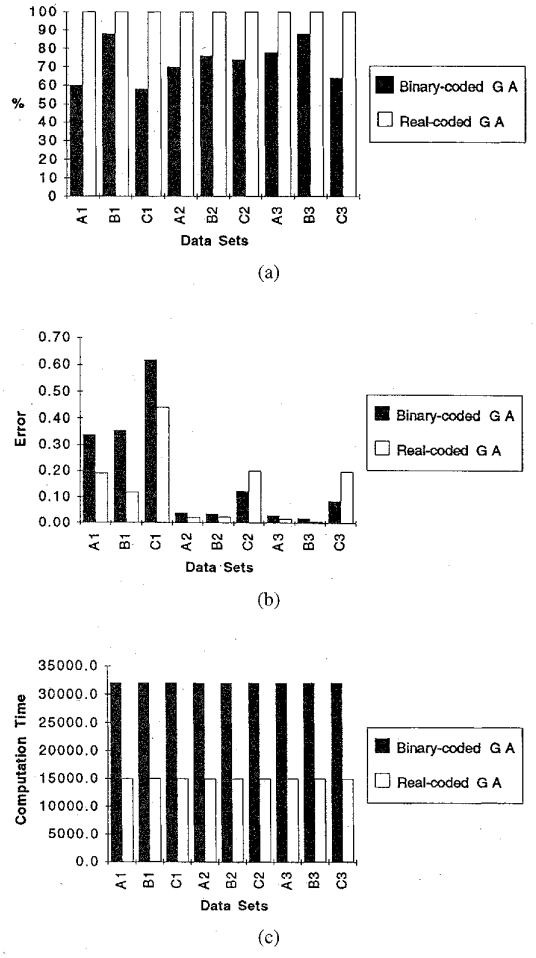


Fig. 13. Comparison of binary-coded and real-coded GA. (a) Reliability. (b) Accuracy. (c) Normalized computation time.

- Probability of cross-over (1 point cross-over): 0.6 (per individual).
- Strategy: elitist.
- Selection: by ranking (selection pressure = 1.4).
- Termination criterion: 400 generations.
- Candidate solution normalized $(0 - 1)$

4) Real-Coded Genetic Algorithms: The meaning of the parameters and details about the implementation are given in [13].

- Chromosome length: 4 (floating point numbers).
- Population size: 15.

TABLE I
SURFACE PARAMETERS

Surface	Surface parameters			
	x_1	x_2	x_3	x_4
A	0.2	0.0	0.3	2.0
B	0.1	0.3	0.0	0.2
C	0.6	−0.8	−0.3	4.0

- Decrease rate of mutation: 5.
- Probability of cross-over (1 point cross-over): 0.8 (per individual).
- Type of cross-over:
  a) continuous crossover: 12%,
  b) discrete crossover: 88%.
- Strategy: elitist.
- Selection: by ranking (selection pressure = 1.5).
- Termination criterion: 1000 generations.
- Constraints handling: by penalty function.
- Candidate solution normalized $(0 - 1)$

5) Darwin-Inspired Combination of Real-Coded Genetic Algorithms and Quasi–Newton (GQD):

- GA Characteristics:
  a) Chromosome length: 4 (floating point numbers).
  b) Population size: 15.
  c) Decrease rate of mutation: 5.
  d) Probability of cross-over (1 point cross-over): 0.7 (per individual).
  e) Type of cross-over:
     - continuous crossover: 29%,
     - discrete crossover: 71%.
  f) Strategy: elitist.
  g) Selection: by ranking (selection pressure = 1.8).
  h) Number of generations: 5
  i) Constraints handling: by penalty function.
  j) Candidate solution normalized $(0 - 1)$

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

TABLE II
ANGLES OF MEASUREMENTS

Data set	n	Angles								
		α_1			α_2			$\alpha_4 - \alpha_3$		
		min	max	step	min	max	step	min	max	step
1	4	50	50	0	50	50	0	0	60	20
2	16	50	60	10	50	60	10	0	60	20
3	64	50	80	10	50	80	10	0	60	20

- QN Characteristics:
  a) Maximum number of function evaluations: 115.
  b) Maximum number of function evaluations for the last generation: 1615

6) Lamarck-Inspired Combination of Real-Coded Genetic Algorithms and Quasi–Newton (GQL):

- GA Characteristics:
  a) Chromosome length: 4 (floating point numbers).
  b) Population size: 15.
  c) Decrease rate of mutation: 5.
  d) Probability of cross-over (1 point cross-over): 0.9 (per individual).
  e) Type of cross-over:
     - continuous crossover: 78%,
     - discrete crossover: 22%.
  f) Strategy: elitist.
  g) Selection: by ranking (selection pressure = 1.2).
  h) Number of generations: 4
  i) Candidate solution normalized $(0 - 1)$
- QN Characteristics:
  a) Maximum number of function evaluations: 20.
  b) Maximum number of function evaluations for the last generation: 1615

7) Multi Level Single Linkage Method (MLSL):

- Selection for local minimization: no sample reduction
- Number of sample points: 1000
- Local minimization: Quasi–Newton (QN):
  a) Maximum number of function evaluations: 65
  b) Maximum number of function evaluations for the last candidate: 1615
- Maximum number of function evaluation (TOTAL) before stopping: 1250
- $\sigma = 4$
- Candidate solution normalized $(0 - 1)$

(See also Tables I and II).

ACKNOWLEDGMENT

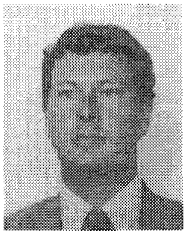
The authors are grateful to M. M. Verstraete and M. N. Mitchison for extensive review of the paper and very helpful discussions.

REFERENCES

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* New York: Addison-Wesley, 1989.
[2] L. Davis, *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991.
[3] G. S. Beveridge and R. S. Schechter, *Optimization: Theory and Practice.* New York: McGraw-Hill, 1970.
[4] P. Gill and W. Murray, "Quasi–Newton methods for unconstrained optimization," *J. Inst. Math. Applicat.,* vol. 9, pp. 91–108, 1972.
[5] A. Nelder and R. Mead, "A simplex method for function optimization," *Computation J.,* vol. 7, pp. 308–313, 1965.
[6] K. Kristinsson and G. A. Dumont, "System identification and control using genetic algorithm," *IEEE Trans. Syst., Man, Cybern.,* vol. 22, no. 5, pp. 1033–1046, 1992.
[7] J.-M. Renders, J.-P. Nordvik, and R. Hanus, "Biological learning metaphors for adaptive process control: A general strategy," in *Proc. IEEE Int. Symp. Intelligent Control,* Glasgow, Aug. 1992.
[8] M. M. Verstraete, B. Pinty, and R. E. Dickinson, "Bidirectional reflectance of vegetation canopies. Part I: Theory," *J. Geophys. Res.,* vol. 95, pp. 11,755–11,765, 1990.
[9] B. Pinty, M. M. Verstraete, and R. E. Dickinson, "Bidirectional reflectance of vegetation canopies. Part II: Inversion and validation," *J. Geophys. Res.,* vol. 95, pp. 11,767–11,775, 1990.
[10] A. H. G. Rinnooy Kan and G. T. Timmer, "Stochastic methods for global optimization," *Amer. J. Mathemat. Manag. Sci.,* vol. 4, pp. 7–40, 1984.
[11] N.A.G., *N.A.G Fortran Library Manual, Mark 14.* Oxford: The Numerical Algorithm Group, 1990.
[12] Z. Michalewicz and C. Z. Janikow, "Ahandling constraints in genetic algorithms," in *Proc. Fourth Int. Conf. Genetic Algorithms,* R. K. Belew and L. B. Booker, Eds. San Francisco: Morgan Kaufman, 1991, pp. 151–157.
[13] C. Z. Janikow and Z. Michalewicz, "An experimental comparison of binary and floating point representation in genetic algorithms," in *Proc. Fourth Int. Conf. Genetic Algorithms.* San Francisco: Morgan Kaufman, 1991, pp. 31–36.
[14] D. E. Goldberg, "Genetic algorithms and Walsh functions: Part II, deception and its analysis," *Complex Syst.,* vol. 3, pp. 153–171, 1989.
[15] T. E. Davis and J. C. Principe, "A simulated annealing like convergence theory for the simple genetic algorithm," in *Proc. Fourth Int. Conf. Genetic Algorithms,* R. K. Belew and L. B. Booker, Eds. San Francisco: Morgan Kaufman, 1991, pp. 174–181.
[16] R. Belew, J. McInerney, and N. N. Schraudoph, "Evolving networks: Using the genetic algorithms with connectionist learning," in *Artificial Life II,* C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Redwood City, CA: Santa Fe Institute Studies in the Sciences of Complexity, 1991, pp. 511–547.
[17] D. Powell, M. Skolnock, and S. S. Tong, "Interdigitation: A hybrid technique for engineering design optimization employing genetic algorithms, expert systems, and numerical optimization," in *Handbook of*

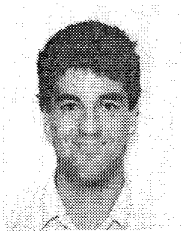
Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.

Genetic Algorithms, L. Davis, Ed. New York: Van Nostrand Reinhold, 1991, pp. 312–331.
[18] T. Dobzhansky, *Genetics of the Evolutionary Process.* New York: Columbia University Press, 1970.
[19] D. H. Ackley and M. L. Littman, "A case for Lamarckian evolution," in *Artificial Life III,* C. G. Langton, Ed. Redwood City, CA: Santa Fe Institute Studies in the Sciences of Complexity, 1994, pp. 3–10.
[20] J. J. Gefenstette, "Lamarckian learning in multi-agent environments," in *Proc. Fourth Int. Conf. on Genetic Algorithms,* R. K. Belew and L. B. Booker, Eds. Morgan Kaufman, 1991, pp. 303–310.
[21] R. Byrd, C. Dert, A. Rinnooy Kan, and R. Schnabel, "Concurrent stochastic methods for global optimization," *Mathemat. Prog.,* vol. 46, pp. 1–29, 1990.



Jean-Michel Renders was born in Brussels, Belgium. He received the Master's degree in mechanical and electrical engineering from the University of Brussels in 1987, and the Ph.D. degree from the same university in 1993.

He is currently working at Tractebel Energy Engineering, Artificial Intelligence Section, Brussels, Belgium. His research interest include neural networks, genetic algorithms, and artificial intelligence techniques applied to process control and power systems.



Stéphane P. Flasse was born in Belgium. He received the Master's degree and Ph.D. degree in agronomy engineering from the Catholic University of Louvain, Louvain-La-Neuve, Belgium, in 1987 and 1993, respectively. His doctoral research was carried out at the Joint Research Centre of the European Commission, Ispra, Italy.

He is currently Senior Scientific Officer, Natural Resources Institute, Chatham, UK. His research interests include the extraction of quantitative and qualitative information from satellite data (inversion of bidirectional reflectance models, vegetation indices), transfer of technologies and adaptive research for application in developing countries.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 8, 2009 at 09:48 from IEEE Xplore. Restrictions apply.