Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 161 (2005) 636–654
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Continuous Optimization

A hybrid method combining continuous tabu search and Nelder–Mead simplex algorithms for the global optimization of multiminima functions

Rachid Chelouah [a], Patrick Siarry [b,*]

[a] *Ecole d'Ingénieurs du canton de Vaud E.I.V.D., Institut d'Informatique, Route de Cheseaux 1, CH-1400 Yverdon les Bains, Suisse, Switzerland*
[b] *Laboratoire d'Etude et de Recherche en Instrumentation, Signaux et Systèmes, Université de Paris 12, 61 Avenue du Général de Gaulle, 94010 Créteil, France*

Received 19 October 2001; accepted 8 August 2003
Available online 11 December 2003

Abstract

Tabu search (TS) is a metaheuristic, which proved efficient to solve various combinatorial optimization problems. However, few works deal with its application to the global minimization of functions depending on continuous variables. To perform this task, we propose an hybrid method combining tabu search and simplex search (SS). TS allows to cover widely the solution space, to stimulate the search towards solutions far from the current solution, and to avoid the risk of trapping into a local minimum. SS is used to accelerate the convergence towards a minimum. The Nelder–Mead simplex algorithm is a classical very powerful local descent algorithm, making no use of the objective function derivatives. A "simplex" is a geometrical figure consisting, in n-dimensions, of $(n + 1)$ points. If any point of a simplex is taken as the origin, the n other points define vector directions that span the n-dimension vector space. Through a sequence of elementary geometric transformations (reflection, contraction and extension), the initial simplex moves, expands or contracts. To select the appropriate transformation, the method only uses the values of the function to be optimized at the vertices of the simplex considered. After each transformation, the current worst vertex is replaced by a better one. Our algorithm called continuous tabu simplex search (CTSS) implemented in two different forms ($CTSS_{single}$, $CTSS_{multiple}$) is made up of two steps: first, an adaptation of TS to continuous optimization problems, allowing to localize a "promising area"; then, intensification within this promising area, involving SS. The efficiency of CTSS is extensively tested by using analytical test functions of which global and local minima are known. A comparison is proposed with several variants of tabu search, genetic algorithms and simulated annealing. CTSS is applied to the design of a eddy current sensor aimed at non-destructive control.

Keywords: Tabu search; Simplex; Global optimization; Continuous variables; Multiminima functions; Eddy current; Non-destructive control

* Corresponding author.
E-mail address: siarry@univ-paris12.fr (P. Siarry).

0377-2217/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2003.08.053

1. Introduction

Globally optimize a function f in a given search domain consists in finding its global minima without being trapped into one of its local minima. To localize a "promising area", likely to contain a global minimum, it is necessary to well "explore" the whole domain. When a promising area is detected, the appropriate tools must be used to "exploit" this area and obtain the optimum as quickly as possible. This task is hardly performed by means of only one method. It is difficult to balance between "exploration" and "exploitation" (sometimes called "diversification" and "intensification", respectively): the tuning of the algorithm parameters must take into consideration contradictory requirements, such as accuracy, reliability and computation time. To solve this problem, closely related to the exploration-exploitation conflict, we propose an algorithm using two processes, each one devoted to one task. Global methods, such as the simulated annealing, the tabu search, and the genetic algorithms, are efficient to well explore the whole solution space, and to localize the "best" areas. On the other hand, local methods are classically available: the hill climbing (e.g. the quasi-Newton method) and the Nelder–Mead simplex method. These methods are more efficient than the previous ones for the exploitation of the best areas already detected.

The aim of this work is to investigate how the limitations inherent to both classes of methods (global and local) may be overcome, and to design an hybrid method, that combines:

- in a first step, the reliability properties of the global search adapted to the continuous case;
- in a second step, the accuracy of the "specialized" local search.

We propose to perform the exploration with tabu search, and the exploitation with Nelder–Mead simplex method. Tabu search, originally developed by Glover [10,11], has been successfully applied to a variety of combinatorial optimization problems. However, few works deal with its application to the global minimization of functions depending on continuous variables. Up to now, we are aware of works [1–5,8,12,19] related to the subject. At the beginning of Section 2, we describe shortly the adaptation of TS to continuous optimization problems, called enhanced continuous tabu search (ECTS), that we have proposed in [5]. ECTS is efficient to explore a wide search space and detect one promising "valley", but it takes too much time to reach the bottom of this valley. In this work, diversification is performed in the same way that in ECTS, but intensification is based on a local search specialized technique, called the Nelder–Mead simplex method [13,14]. We chose this local search method because it is robust (i.e. tolerant towards noise in the function value), easy to be programmed and fast. To evaluate the efficiency of CTSS in its two forms, we have implemented a set of benchmark functions, and we compare our results to the ones supplied by the other competitive methods. At last, our algorithm CTSS is applied to the design of a sensor aimed at non-destructive control.

The paper is organized as follows. Section 2 is devoted to the overall description of our optimization method. The algorithm is presented in detail in Section 3. The initialization and the tuning of the parameters are discussed in Section 4. Section 5 consists in a discussion of the experimental results. Section 6 is devoted to the industrial application of our algorithm. Some words of conclusion are given in Section 7.

2. General description of our optimization method

Our method aims at solving any optimization problem showing the following properties:

- one single objective function f;
- f depending on several continuous variables;
- problem considered as "difficult", because of the presence of numerous local optima (due to non-linearities, existence of correlated variables,...) and/or because of the unavailability of an analytical expression of f;
- derivatives of f not easily available (high computation cost, existence of "numerical calculus noises" due, for instance, to the evaluation of

f through the use of a simulator involving approximate evaluations);
- only "box type" constraints: each variable evolves within a given search domain.

Let be $\boldsymbol{x}$ one vector with a finite dimension n, of which components x_i verify:

$$a_i \leqslant x_i \leqslant b_i, \quad i = 1, \ldots, n,$$

a_i and b_i being the components of two vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, of dimension n. The vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ define an hyperrectangular solution space, noted $\boldsymbol{X} \subset R^n$.

Let be f a function defined as follows:

$$f : \boldsymbol{X} \to R,$$

$$\boldsymbol{x} \to f(\boldsymbol{x})$$

$f(\boldsymbol{x}^*)$ is a local minimum $\iff [\exists \varepsilon > 0 / \ \forall \boldsymbol{x} \in \boldsymbol{X} : //\boldsymbol{x} - \boldsymbol{x}^*// < \varepsilon \Leftarrow f(\boldsymbol{x}) \geqslant f(\boldsymbol{x}^*]$ and $\boldsymbol{x}^* \in \boldsymbol{X}$ $f(\boldsymbol{x}^*)$ is a global minimum $\iff \forall \boldsymbol{x} \in \boldsymbol{X} : f(\boldsymbol{x}) \geqslant f(\boldsymbol{x}^*)$ and $\boldsymbol{x}^* \in \boldsymbol{X}$.

2.1. Sketching features of ECTS and of our CTSS algorithm in its two forms

In ECTS, the processing was applied directly on the non-normalized initial variation domains of variables. The diversity of variables as well as the possible disparity in their variation domains lead us in CTSS to normalize these parameters, by transforming their variation domains into the same domain $[0, 1]$. The variables recover their real values when evaluating the objective function (by inverse transformation). We proposed two kinds of normalization, linear and logarithmic, according to the variation domain of each problem variable. In this way, the initial hyperrectangular search space becomes an hypercube search domain having an edge length equal to the unit. The hypercube neighborhood of the current solution is obtained by dividing each edge of the initial hypercube search domain by the given factor ρ_{neigh}.

To generate neighbors of the current solution, we proposed two methods:

(a) In the first version, we generated one neighbor inside the hypercube neighborhood following a given direction chosen randomly, with a step equal to $\frac{1}{\rho \text{ neigh}}$. This algorithm is called $\text{CTSS}_{\text{single}}$, because at each generation only one variable is disturbed (see on Fig. 1). Note that the search cannot return to the previous current solution because that solution is a member of the tabu list. This method is interesting only when the number of variables is low (less than 5).

(b) In the second version, we have adapted the method described in detail in [19]. The hypercube neighborhood is partitioned into concentric hypercubes around the current solution $\boldsymbol{x}$, and the neighbors of $\boldsymbol{x}$ are obtained by random selection of one point inside each thus delimited region (see on Fig. 2). This algorithm is called $\text{CTSS}_{\text{multiple}}$; we disturb several variables to obtain a new neighbor

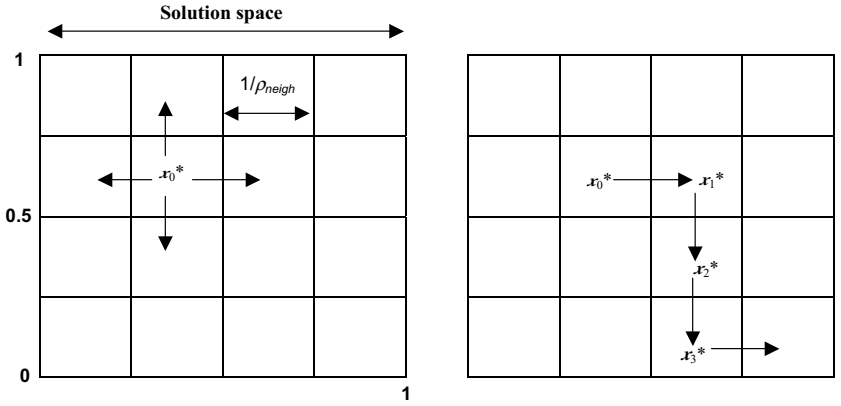

Fig. 1. Neighborhood and diversification strategy in $\text{CTSS}_{\text{single}}$.

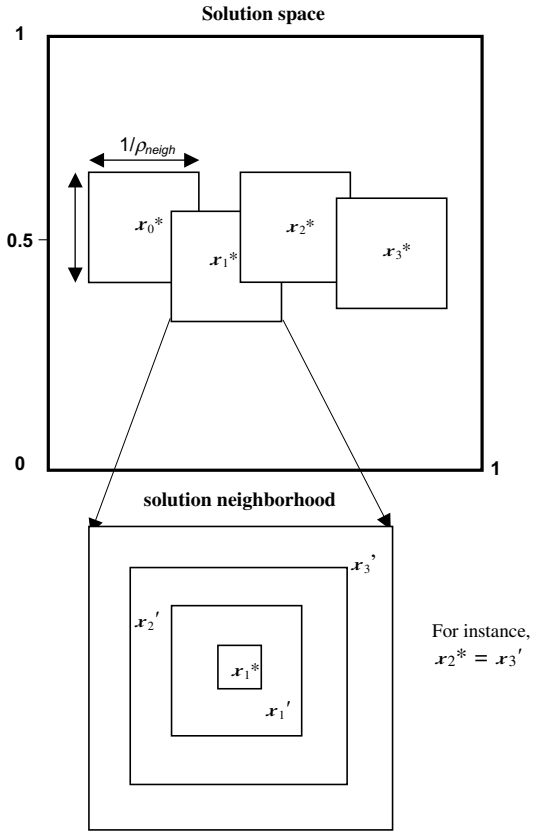

Fig. 2. Partition of current solution neighborhood and diversification strategy in $CTSS_{multiple}$.

(1/3 of the variables if the number of variables is more than five, otherwise two variables are disturbed).

In both versions, each neighbor is accepted only if it belongs neither to the "tabu" list, nor to the "promising" list, subsequently defined. This process avoids the danger of the appearance of a cycle: the neighbors of the current solution, which belong to the tabu list, are systematically eliminated. The objective function to be minimized is evaluated for each accepted solution $\boldsymbol{x}'$, and the best of these neighbors becomes the new current solution, even if it is worse than the previous current solution.

Our hybrid algorithm CTSS in its two forms ($CTSS_{single}$ and $CTSS_{multiple}$), as ECTS, performs first diversification for locating a "promising area". A new promising area is detected whenever an "unacceptable" deterioration (in the sense defined below) of the objective function occurs inside the neighborhood area centered around the current solution: if all neighbors at hand are worse than the current solution, this solution is a local optimum. If all related deteriorations of the value of the objective function are higher than a given threshold, we can then consider that this local optimum is at the center of a hyperrectangular zone called "promising area". The centers of successively obtained promising areas are stored in a list, called the "promising list". To avoid returning to already visited areas, we check that each newly detected "promising solution" is not inside one of the promising balls, the centers of which are saved in the promising list. This restriction forces the search towards solutions far enough away from the solutions previously obtained.

If a new promising area is accepted, CTSS stops the exploration process, and starts the exploitation by intensifying the search inside this newly detected promising area. In this intensification phase, the strategy of renewal of the promising list and the determination of the best solution are the same that in ECTS.

2.2. Nelder–Mead simplex search

A "simplex" is a geometrical figure consisting, in n-dimensions, of $(n+1)$ points $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n$ [13,14]. If any point of a simplex is taken as the origin, the n other points define vector directions that span the n-dimension vector space.

If we randomly draw as initial starting point s_0, then we generate the other n points s_i according to the relation $\boldsymbol{x}_i = \boldsymbol{x}_0 + \lambda e_j$, where the e_j are n unit vectors, and λ is a constant which is typically equal to one (but may be adapted to the problem characteristics).

Through a sequence of elementary geometric transformations (reflection, contraction, expansion and multi-contraction), the initial simplex S_0 moves, expands or contracts. To select the appropriate transformation, the method only uses the values of the function to be optimized at the vertices of the simplex considered. After each transformation, the current worst vertex is replaced by a

better one. Trial moves shown on Fig. 3 are generated according to the following basic operations (where $\overline{\boldsymbol{x}}$ is defined by $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$, and α, β, γ are constants):

reflection: $\boldsymbol{x}_r = (1+\alpha)\overline{\boldsymbol{x}} - \alpha \boldsymbol{x}_n$,

expansion: $\boldsymbol{x}_e = \gamma \boldsymbol{x}_r + (1-\gamma)\overline{\boldsymbol{x}}$,

contraction: $\boldsymbol{x}_c = \beta \boldsymbol{x}_n + (1-\beta)\overline{\boldsymbol{x}}$.

At the beginning of the algorithm, one moves only the point of the simplex, where the objective function is worst (this point is called "high"), and one generates another point image of the worst point. This operation is the reflection. If the reflected point is better than all other points, the method expands the simplex in this direction, otherwise, if it is at least better than the worst one, the algorithm performs again the reflection with the new worst point. The contraction step is performed when the worst point is at least as good as the reflected point, in such a way that the simplex adapts itself to the function landscape and finally surrounds the optimum. If the worst point is better than the contracted point, the multi-contraction is performed. For each rejected contraction step, we replace all $\boldsymbol{x}_i$ of the simplex by $\frac{(\boldsymbol{x}_i+\boldsymbol{x}_l)}{2}$ ($\boldsymbol{x}_l$ is the vertex of the simplex where the objective function is "low"); thus we obtain the multi-contraction (or "shrink") of the simplex, and the process restarts.

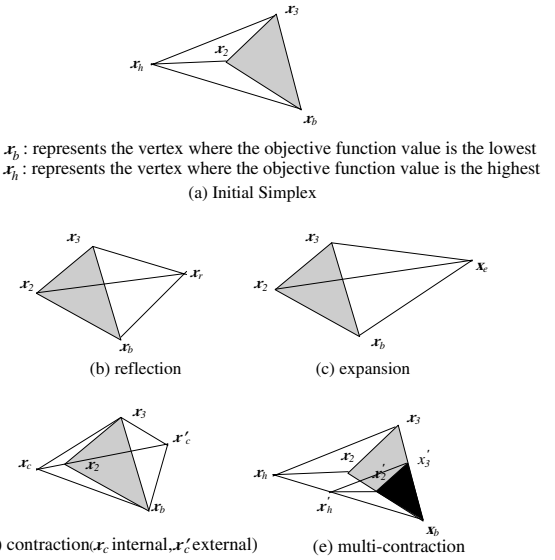

Fig. 3. Available moves in the Nelder–Mead simplex method, in the case of 3 variables.

The stopping criterion is a measure of how far the simplex was moved from one iteration k to the following one $(k+1)$. The algorithm stops when:

$$\frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{x}_i^k - \boldsymbol{x}_i^{k+1}\|^2 < \varepsilon,$$

where $\boldsymbol{x}^{k+1}$ is the vertex replacing $\boldsymbol{x}^k$ at the iteration $(k+1)$, and ε is a given "small" positive real number.

3. Detailed presentation of the algorithm (CTSS)

The structure of CTSS in its two forms is shown in Fig. 4, and the pseudocode is listed in Fig. 5. In addition to the main stages, exploration and exploitation, it exhibits also the initialization phase and the selection of the best solution among the ones saved inside the promising list.

3.1. Initialization

In this section, we introduce several parameters required by our algorithm. Some of them are initialized at the beginning and other ones, called "control parameters", are carefully deduced in taking function characteristics into account (this issue is reported in Section 4). In the first step, we define the following parameters: the length L_t of the tabu list, the length L_p of the promising list, the definition domain of each variable and the relative difference between two successive solutions. In the second step, only in case of $\text{CTSS}_{\text{multiple}}$, we set the number η of neighbors, the parameters ρ_t and ρ_p which determine the radius ε_t and ε_p of the tabu and promising balls, respectively (see details in Section 4), otherwise we set the neighborhood reduction parameter ρ_{neigh}, and the stopping criteria. We also set the Simplex Search parameters (α, β, γ) that are used by elementary geometric transformations (reflection, contraction and expansion, respectively).

3.2. Exploration

In this sub-section, we summarize the diversification phase. To explore the whole search domain, and localize the promising areas, the algorithm

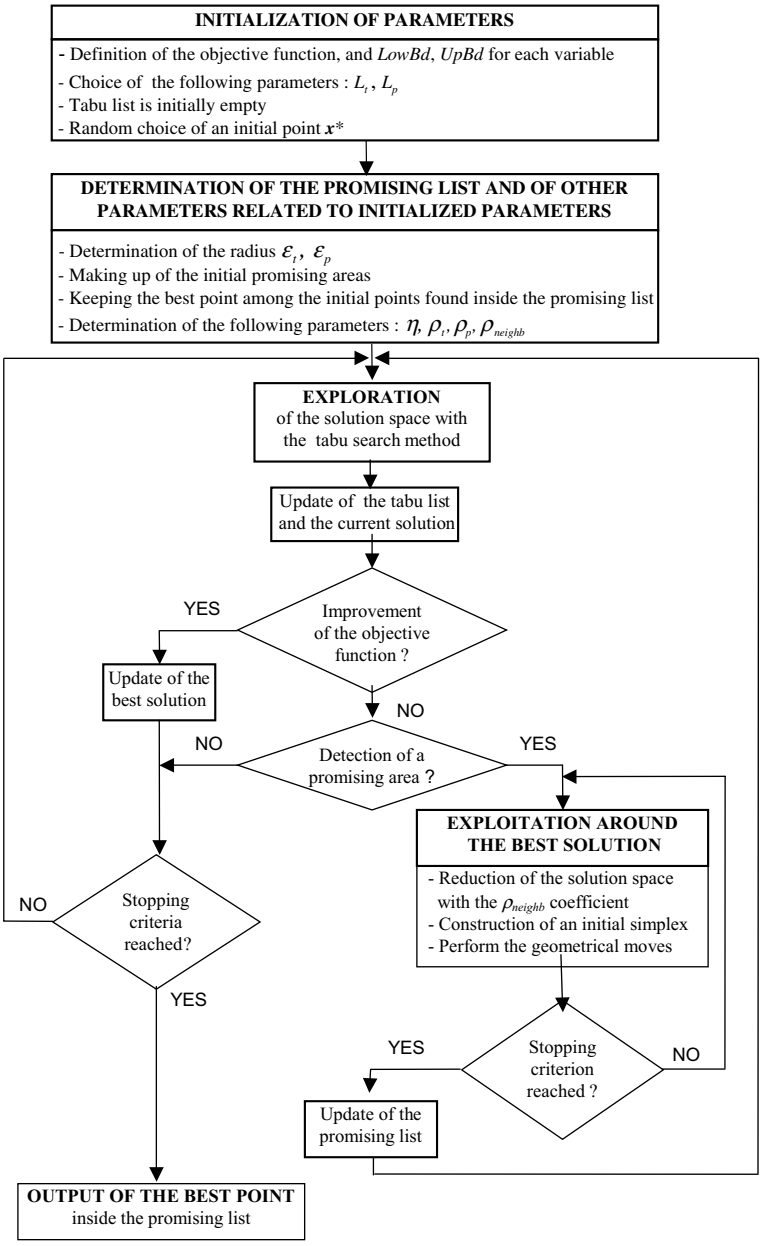

Fig. 4. Simplified chart of the algorithm CTSS in its two forms.

starts from the initial solution x_0 randomly generated. Temporarily this solution is the best one, called x^*. The algorithm generates and accepts a specified number η of neighbors (called x') in case of $\text{CTSS}_{\text{multiple}}$ or 3 neighbors in case of $\text{CTSS}_{\text{single}}$; the objective function to be minimized is evaluated for each accepted solution x', and the best of these neighbors becomes the new current solution as explained in Section 2.1. This new current solution is added to the tabu list of length L_t. When this list is full, it is updated by removing the first solution entered (FIFO memory). When a new promising

TabuList : list of recently visited solutions of length L_t
PromList : list of local or global minima of length L_p
PromList := generation of L_p solutions well dispersed within the search domain
TabuList := empty
$\boldsymbol{x}$:= random solution
$\boldsymbol{x}^* := \boldsymbol{x}$
$f_{\min} := f(\boldsymbol{x}^*)$
$\boldsymbol{x}_{min} := \boldsymbol{x}^*$

REPEAT ***DIVERSIFICATION***

$\overline{f} := \frac{1}{L_p}\sum_{i=1}^{Lp} f_i$ the unacceptable degradation of the objective function

CASE : $CTSS_{multiple}$
generate a N-sample such that $s_i(\boldsymbol{x}^*) \in$ neighborhood $S(\boldsymbol{x}^*)$ and
$s_i(\boldsymbol{x}^*) \notin$ *TabuList* and $s_i(\boldsymbol{x}^*) \notin$ *PromList*
$f(s(\boldsymbol{x}^*)) = \min_{1 \le i \le N}[f(s_i(\boldsymbol{x}^*))]$

CASE : $CTSS_{single}$
For each three orthogonal directions
$s_i(\boldsymbol{x}) = s(\boldsymbol{x}) + hi$
calculate $f(s_i(\boldsymbol{x}))$
$f(s(\boldsymbol{x}^*)) = \min_{1 \le i \le 3}[f(s_i(\boldsymbol{x}^*))]$

END CASE

add ($s(\boldsymbol{x}^*)$, *TabuList*)
$\boldsymbol{x}^* := s(\boldsymbol{x}^*)$

IF $f(\boldsymbol{x}^*) < f_{\min}$
$f_{\min} := f(\boldsymbol{x}^*)$
$\boldsymbol{x}_{min} := \boldsymbol{x}^*$
ELSE
IF (*detection of a promising area*) ***INTENSIFICATION***
$\boldsymbol{X} := \boldsymbol{X} / \rho_{neigh}$ reduction of the search domain
$\{\boldsymbol{x}, f(\boldsymbol{x})\}$: best solution obtained by simplex search
replace the worst solution inside *PromList* by $\{\boldsymbol{x}, f(\boldsymbol{x})\}$
END IF
END IF

UNTIL stopping criterion are reached (no promising area after a given number of iterations)

Fig. 5. Pseudocode of CTSS algorithm in its two forms.

area is localized as described in Section 2.1, we check that it does not yet belong to a promising ball. If a new promising area is accepted, we stop the exploration process, and we start the exploitation process inside this newly detected promising area. When the best point inside this area is calculated, we add it to the promising list of constant length L_p, and we replace the worst one inside it. The use of the promising and tabu lists stimulates the search for solutions far from the current one and the yet identified promising areas.

The exploration stops when one of the following conditions is reached:

- the given number of successive evaluations without any detection of a new promising area MaxEval;
- the maximum number of iterations MaxIter.

3.3. Exploitation

The difference lies in the way the moves are generated. The moves inside the promising area are performed by using the Nelder–Mead simplex algorithm. This algorithm is very simple to be programmed. It starts from the center of the promising area found so far, each domain edge is reduced in a ratio given by the reduction parameter ρ_{red}, and the Nelder–Mead simplex algorithm performs the initial simplex S_0. The search is performed by using trial points, obtained through elementary geometric transformations (reflection, contraction and expansion, see on Fig. 3).

The algorithm performs this exploitation as long as the local stopping criterion is not reached. Otherwise, the algorithm stops the exploitation phase, puts the best point of the simplex in the promising list, replacing thus the worst one, and starts again the diversification phase from the best point of the simplex with a new tabu list.

3.4. Stopping criteria

The local stopping criterion is a measure (called ε in Section 2.2) of how far the simplex was moved. Other parameters defining the transition between the diversification phase and the intensification phase, such as the given number of successive evaluations MaxEval, without any detection of a new promising area, greatly influence the CPU time.

The program exhibits the best point among the ones inside the promising list as the global minimum.

3.5. Comparison between the two forms of our algorithm CTSS for one example

To compare these two versions of our algorithm, we applied them to the function B2 of two variables, defined in the domain $[-1,1]^2$. Figs. 6 and 7 show the path followed during the diversification phase by $CTSS_{single}$ and $CTSS_{multiple}$, respectively. For $CTSS_{single}$, the diversification phase takes 97 evaluations, and localizes 6 promising areas. The best area is found around the solution ($\boldsymbol{x}_{opt} = -0.00905$, $\boldsymbol{y}_{opt} = 0.0338$) and the function value at this solution is $f_{opt} = 0.039$. For $CTSS_{multiple}$, the diversification requires more iterations than for $CTSS_{single}$ because four neighbors

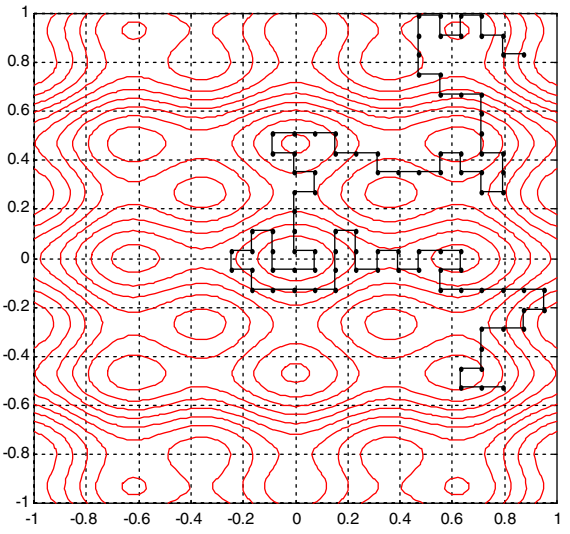

Fig. 6. Search path followed in the solution space $[-1,1]^2$, for the B2 function, during the diversification phase, for the first version of the neighborhood ($CTSS_{single}$).

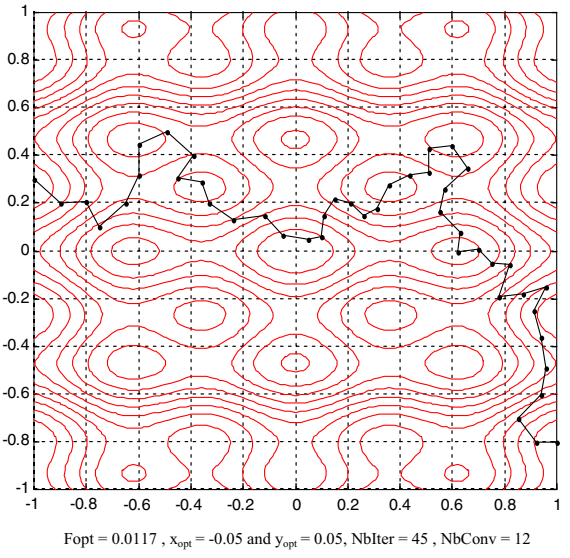

Fig. 7. Search path followed in the solution space $[-1,1]^2$, for the B2 function, during the diversification phase, for the second version of the neighborhood ($CTSS_{multiple}$).

are generated at each iteration. The number of evaluations is equal to 4 times the iterations number, that is $4\times45=180$. With $\mathrm{CTSS}_{\mathrm{multiple}}$ we again localized 6 promising areas. The best area is found around the solution ($\boldsymbol{x}_{\mathrm{opt}}=-0.05$, $\boldsymbol{y}_{\mathrm{opt}}=0.05$) and the function value at this solution is $f_{\mathrm{opt}}=0.0117$.

Figs. 8 and 9 show the plotting of the objective function per the number of evaluations during the intensification phase in both cases, respectively, illustrated by Figs. 6 and 7. In the first case, the previous best solution becomes ($\boldsymbol{x}_{\mathrm{opt}}=-0.0676\times10^{-3}$, $\boldsymbol{y}_{\mathrm{opt}}=0.9955\times10^{-3}$), with the function value equal to $f_{\mathrm{opt}}=0.3344\times10^{-5}$, after 26 evaluations. The objective function value is divided by 13. In the second case, the previous best solution becomes ($\boldsymbol{x}_{\mathrm{opt}}=-0.0991\times10^{-3}$, $\boldsymbol{y}_{\mathrm{opt}}=0.9978\times10^{-3}$) with the function value equal to $f_{\mathrm{opt}}=0.3563\times10^{-4}$, after 29 evaluations. The objective function value is divided by 17.

Plottings of Figs. 8 and 9 allow to visually show the fall in the objective function value as optimization progresses. In addition, these plottings clearly illustrate the meaning of NbIter and NbConv parameters: for instance, for the path in the solution space drawn on Fig. 6, we have NbIter = 97 and NbConv = 17. These parameters are easily readable on Fig. 8: NbIter is the number of evaluations (abscissa) and NbConv is the total number of improvements in the objective function value, each improvement corresponding to a drop in the ordinate.

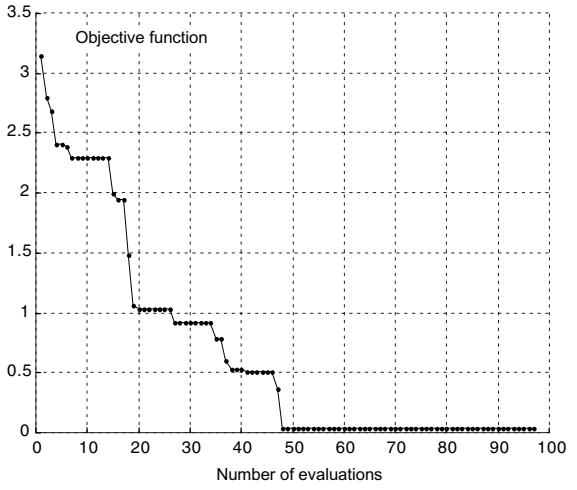

Fig. 8. Convergence of the objective function during the search for the first version of the neighborhood ($\mathrm{CTSS}_{\mathrm{single}}$).

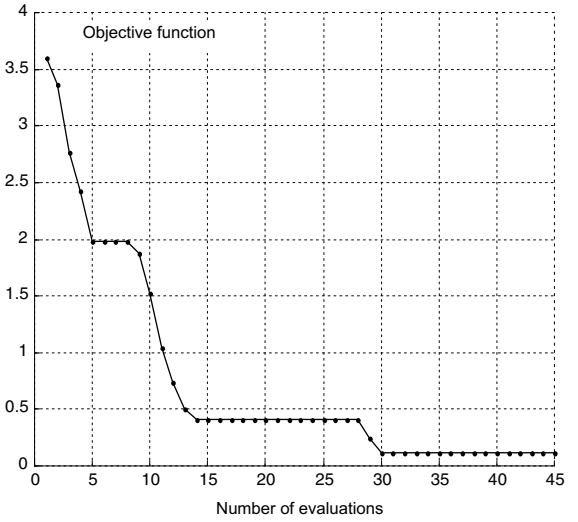

Fig. 9. Convergence of the objective function during the search for the second version of the neighborhood ($\mathrm{CTSS}_{\mathrm{multiple}}$).

4. Initialization of some parameters and tuning of the control parameters

In this section we describe the initialization of some parameters and the tuning of the *control parameters* (which allow to manage the overall behavior of the algorithm).

We first list the parameters which are fixed at the beginning or which are automatically built by using the parameters fixed at the beginning. Then, we explain how the control parameters may be tuned.

The parameters fixed at the beginning are fixed in the following way:

- the search domain of analytical test functions is set as prescribed in the literature,
- the initial solution $\boldsymbol{x}^*$ is randomly chosen,
- the tabu list is initially empty,
- to complete the promising list, the algorithm randomly draws a point; this point is accepted

as the center of an initial promising ball, if it does not belong to an already generated ball. By this way the algorithm generates L_p sample points, which are uniformly dispersed in the whole space solution $\boldsymbol{X}$,
- the sample defined above is used to initialize the initial threshold, taken equal to the average of the objective function values over L_p sample points,
- the previous sample is also used to initialize the best point found, taken equal to the best point among the L_p sample points,
- the number η of neighbors of the current solution investigated at each iteration is set to twice the number of variables, if this number is equal or small than five, otherwise η is set to ten,
- the maximum number of successive iterations without any detection of a new promising area is equal to twice the number of variables,
- the maximum number of successive iterations without any improvement of the objective function value is equal to five times the number of variables,
- the maximum number of successive reductions of the hypercube neighborhood and of the radius of tabu balls without any improvement of the objective function value is set to twice the number of variables,
- the maximum number of iterations is equal to 50 times the number of variables,
- simplex parameters (α, β, γ) are equal to 1, 2 and 1/2, respectively.

After a series of trials using the analytical test functions listed in the Appendix A, we set the value of some of the control parameters. We consider that the chosen setting is satisfactory for two reasons:

- this setting represents a compromise over a large set of analytical test functions (20 functions of 2 to 100 variables; several tests are performed for each function);
- the adopted tunings are often near from those adopted by other authors, what makes easier the further comparison between competitive algorithms.

That setting is as follows:

- the length L_t of the tabu list is set to 20;
- the length L_p of the promising list is set to 10, like in [4];
- the parameter ρ_t allowing to calculate the radius of tabu balls, and the parameter ρ_p allowing to calculate the radius of promising balls are set to 100 and 50, respectively;
- the parameter ρ_{neigh} allowing to calculate the size of the hypercube neighborhood is equal to twice the number of variables;
- the values of the radius ε_t of tabu balls and of the radius ε_p of promising balls are equal to $1/\rho_t$ and $1/\rho_p$, respectively.

5. Experimental results

The efficiency of CTSS in its two forms was tested using a set of benchmark functions [9,15–18], then a comparison was performed with various methods listed in Table 1. To avoid any misinterpretation of the optimization results, related to the choice of any particular initial population, we performed each test 100 times, starting from various randomly selected points, inside the hyperrectangular search domain specified in the usual literature, and normalized to $[0, 1]$.

The results of $CTSS_{multiple}$ tests performed on 12 functions listed in the Appendix A are shown in Table 2. To evaluate the program efficiency, we

Table 1
List of various methods used in the comparison

Method	References
Continuous tabu simplex search (CTSS) in its two forms	This work
Enhanced continuous tabu search (ECTS)	[5]
Continuous genetic algorithm (CGA)	[6]
Continuous hybrid algorithm (CHA)	[7]
Enhanced simulated annealing (ESA)	[20]
Continuous reactive tabu search ($CRTS_{min}$) minimum	[1]
Continuous reactive tabu search ($CRTS_{ave}$) average	[1]
Taboo search (TS)	[8]
INTEROPT	[2]

retained the following criteria summarizing results from 100 minimizations per function: the rate of successful minimizations, the average of the objective function evaluation numbers and the average "error". These criteria are defined precisely below.

When at least one of the stopping tests is verified, CTSS in its two forms stops and provides the coordinates of a located point, and the objective function value "$\mathrm{FOBJ}_{\mathrm{CTSS}}$" at this point. We compared this result with the known analytical minimum "$\mathrm{FOBJ}_{\mathrm{ANAL}}$". To decide whether a solution found by CTSS is "acceptable", we claim that it is necessary to have an idea about the landscape of the objective function f to be optimized. If f is very flat, a solution x can be accepted only if $f(x)$ is very "near" from $f(x^*)$, where x^* stands for the global optimum: otherwise x could be far from x^*, at worse anywhere within the solution domain. On the contrary, if the landscape of f is very steep, comprising several valleys of unequal depths, a solution x such that $f(x)$ is relatively more distant from $f(x^*)$ may be accepted, because, in that case, x will nevertheless be in the neighborhood of x^*. To estimate the flatness of f landscape, various methods of statistical type may be used. For reasons of simplicity, we decided to exploit only an empirical evaluation of f average value. Therefore we retained the following criterion for a successful optimization: the solution produced by CTSS was considered to be "successful" if the following inequality held:$|\mathrm{FOBJ}_{\mathrm{CTSS}} - \mathrm{FOBJ}_{\mathrm{ANAL}}| < \varepsilon^*_{\mathrm{rel}}| < \mathrm{FOBJ}_{\mathrm{INIT}} > | + \varepsilon_{\mathrm{abs}}$, where $\varepsilon_{\mathrm{rel}} = 10^{-4}$, $\varepsilon_{\mathrm{abs}} = 10^{-6}$ and $< \mathrm{FOBJ}_{\mathrm{INIT}} >$ is an empirical average of the objective function value, calculated over typically 100 points, randomly selected inside the search domain, before running the algorithm.

The average of the objective function evaluation numbers is evaluated in relation to only the successful minimizations. The average error is defined as the average of FOBJ gaps between the best successful point found and the known global optimum.

The results from the various methods that we developed are compared in Tables 2 and 3.

The performance of $\mathrm{CTSS}_{\mathrm{multiple}}$ was then compared to other published versions of continuous GA algorithms, simulated annealing and tabu search, listed in Table 1. For this purpose, it was necessary to use the following more classical definition of successful minimizations: a result was considered as successful if the following inequality held: $|\mathrm{FOBJ}_{\mathrm{CTSS}} - \mathrm{FOBJ}_{\mathrm{ANAL}}| < \varepsilon^*_{\mathrm{rel}}\mathrm{FOBJ}_{\mathrm{ANAL}} + \varepsilon_{\mathrm{abs}}$. Results are gathered in Table 4.

CGA and CHA are the names given to our two algorithms involving a genetic algorithm for the global optimization of multiminima functions. CGA and CHA are described in detail in Refs. [6,7], respectively. We recall hereafter some features of both algorithms.

Table 2
Results provided by $\mathrm{CTSS}_{\mathrm{multiple}}$ and CHA for 12 test functions

Test function	Rate of successful minimizations (%)		Average of objective function evaluation numbers		Average of FOBJ gaps between the best successful point found and the known global optimum	
	$\mathrm{CTSS}_{\mathrm{multiple}}$	CHA	$\mathrm{CTSS}_{\mathrm{multiple}}$	CHA	$\mathrm{CTSS}_{\mathrm{multiple}}$	CHA
RC	100	100	125	295	5e−3	1e−4
B2	100	100	98	132	5e−6	2e−7
ES	100	100	325	952	5e−3	0.001
GP	100	100	119	259	0.001	0.001
SH	100	100	283	345	0.001	0.005
R_2	100	100	369	459	0.004	0.004
Z_2	100	100	78	215	3e−7	3e−6
DJ	100	100	155	371	2e−4	2e−4
$H_{3,4}$	100	100	225	492	0.005	0.005
$S_{4,5}$	75	85	538	598	0.007	0.009
$S_{4,7}$	77	83	590	620	0.001	0.01
$S_{4,10}$	74	81	555	635	0.001	0.015

Table 3
Average number of objective function evaluations used by our 4 methods to optimize 8 functions of less than 5 variables

Function	Method			
	$CTSS_{multiple}$	CHA	CGA	ECTS
RC	125	295	620	245
B2	98	132	220	170
GP	119	259	410	231
SH	283	345	575	370
R_2	369	459	960	480
Z_2	78	215	620	195
$H_{3,4}$	225	492	582	548
$S_{4,5}$	538 (0.75)	598 (0.85)	610 (0.86)	825 (0.75)

The numbers in parentheses are the ratios of runs for which the algorithm found the global minimum rather than being trapped into a local minimum.

Table 4
Comparison of our method to five other methods to optimize 10 functions of less than 10 variables

Function	Method					
	$CTSS_{multiple}$	$CRTS_{min}$	$CRTS_{ave}$	TS	ESA	INTEROPT
RC	125	41	38	492	–	4172
B2	175	–	–	–	–	–
GP	151	171	248	486	783	6375
SH	279	–	–	727	–	–
R_2	428	–	–	–	796	–
Z_2	7835	–	–	–	15820	–
$H_{3,4}$	258	609	513	508	698	1113
$S_{4,5}$	545 (0.69)	664	812	–	1137 (0.54)	3700 (0.4)
$S_{4,7}$	620 (0.68)	871	960	–	1223 (0.54)	2426 (0.6)
$S_{4,10}$	589 (0.65)	693	921	–	1189 (0.5)	3463 (0.5)

The numbers in parentheses are the ratios of runs for which the algorithm found the global minimum rather than being trapped into a local minimum.

In CGA and CHA, to localize a "promising area", likely to contain a global minimum, we argued that it is necessary to well "explore" the whole search domain. When a promising area is detected, the appropriate tools must be used to "exploit" this area and obtain the optimum as accurately and quickly as possible. Each algorithm uses one process for the exploration and one process for the exploitation: in CGA, both processes are performed with a genetic algorithm; in CHA, the exploration is performed with a genetic algorithm, and the exploitation with a Nelder–Mead simplex search. The tuning of control parameters during exploration phases is the same in two algorithms CGA and CHA.

The initial population is chosen to cover homogeneously the whole solution space, and to avoid the risk of having too many individuals in the same region: its size is set to 30. The crossover probability is set to 0.9; the initial mutation probability is set to 0.85, and then follows a decreasing law. Usually the mutation probability is in inverse ratio to the number of variables: accordingly, we introduced an exponential reduction of the mutation probability. When performing exploitation, CGA reduces the search domain until it finds a promising area. The population size is then gradually reduced. Too fast reduction leads to risks of premature convergence of the algorithm, but too low reduction penalizes the computing time: the rate of population size decreasing was empirically set to 2.

Table 2 shows the comparison between our two algorithms ($CTSS_{multiple}$ and CHA) using the simplex search during the intensification phase. Table 2

displays the average number of objective function evaluations used to optimize 12 functions, including 7 functions of 2 variables. Results appeal the following comments. For functions of 2 variables, the average of the objective function evaluation numbers does not exceed 1000, with a good accuracy in most cases. Some less satisfactory results are due to particularities: for instance, the global minimum of the Easom (ES) function is in a very narrow hole of the solution space. Furthermore, for the three Shekel functions ($S_{4,5}$, $S_{4,7}$ and $S_{4,10}$), $\text{CTSS}_{\text{multiple}}$ is sometimes trapped into a local minimum. We can remedy to this problem by increasing the size of the promising and tabu lists in the diversification phase. We remark that $\text{CTSS}_{\text{multiple}}$ is faster than CHA because, during the diversification phase, CHA uses a genetic algorithm, what needs handling a population of solutions. But using such a population better prevents CHA from trapping into one local minimum.

Table 3 shows the comparison between our four algorithms ($\text{CTSS}_{\text{multiple}}$, CHA, ECTS and CGA), related to the average number of objective function evaluations. We can notice a clear improvement of algorithms based on tabu search ($\text{CTSS}_{\text{multiple}}$ and ECTS) in comparison with algorithms based on the genetic process. In other respects $\text{CTSS}_{\text{multiple}}$ is more simple to be implemented than ECTS, because it uses the simplex search during the intensification.

In Table 4, $\text{CTSS}_{\text{multiple}}$ is compared to five other algorithms. We first notice that $\text{CTSS}_{\text{multiple}}$ is more efficient that other methods for functions having more than 2 variables. For functions of less than 2 variables, $\text{CTSS}_{\text{multiple}}$ is a little less better than CRTS. We can consider that $\text{CTSS}_{\text{multiple}}$ results are satisfactory for all the functions.

6. Application to the design of an eddy current sensor for non-destructive control

Eddy currents in a conducting material are induced currents resulting from an external variation of a magnetic field. According to Lenz's law, eddy currents create a magnetic flux tending to oppose that which gave them birth. The resulting field varies in amplitude and phase, according to its depth inside the material. The intensity and distribution of the eddy currents and the resulting field depend on:

- the conductivity σ of the material;
- its permeability μ;
- its geometry;
- the presence of possible defects inside the controlled material.

The induced currents circulate mainly in the proximity of the surface: known as the "skin effect". A characteristic parameter, in the study of the sinusoidal eddy currents, is the skin depth:

$\delta = \dfrac{1}{\sqrt{\pi\sigma\mu f}}$, where f is the frequency of the induction current.

This parameter expresses the standard penetration depth of the induced currents inside the material. The difference in phase $(\Phi_z - \Phi_0)$ between the surface current density and the density at depth z linearly varies with this depth:

$$\Phi z - \Phi 0 = -\frac{z}{\delta}. \tag{1}$$

6.1. Principle of control

Drills, or electromagnetic sensors, bearing an AC current, induce currents in controlled conducting materials. The eddy currents give information about tested materials and their geometry, it is however not possible to measure them experimentally. To overcome this difficulty, we use the impedance measurement of the system sensor/material. The interesting parameter to exploit, in this case, is the impedance variation between material with defect and material without defect. The control consists of inducing currents inside the material to be controlled, using an exciting bobbin, while a detecting bobbin is used to detect the current flow resulting from the combination of inductive and induced flows. In the configurations which we studied, the same bobbin is used for exciting and receiving: we speak of a "double function sensor", schematized in Fig. 10.

Finally the resulting current flow, and the total impedance of the sensor/material, vary in amplitude and in phase, according to the density of the currents induced inside the material. Investigation of the real and imaginary components of the impedance Z (resistance R and reactance X, respectively) not only allows a defect to be localised, but also the nature of the defect to be specified.

6.2. Calculation of the bobbin impedance in the presence of a material

The bobbin impedance in the presence of a material (Z_{mat}) is obtained by dividing the voltage of the bobbin by the intensity of the current which circulates in the bobbin.

As $U = \int_{Bobbin} \vec{E} \cdot d\vec{l}$ and $\vec{E} = -\frac{\partial \vec{A}}{\partial t} = -j\omega\vec{A}$, then $U = -j\omega \int_{Bobbin} \vec{A} \cdot d\vec{l}$

With $Z = \frac{U}{I}$, we obtain after calculations:

$$Z_{mat} = \frac{j\omega\pi\mu_0 n^2}{S^2} \int_0^\infty \frac{1}{\alpha^6} J^2(r_1, r_2)\{2\alpha(l_2 - l_1) + [2e^{-\alpha(l_2 - l_1)} - 2 + \{e^{-2\alpha l_2} + e^{-2\alpha l_1} - 2e^{-\alpha(l_2 + l_1)}\}B_1'(\alpha)]\}\, d\alpha,$$

where $S = (l_2 - l_1)(r_2 - r_1)$ is the surface of the spire section and $B_1'(\alpha)$ is the Bessel function' coefficient; other parameters are defined in Fig. 11.

The un-normalized ranges of the problem variables are listed below:

R_{int}: 1–100 mm
R_{ext}: calculated as $R_{int} + \Delta R$, with $\Delta R \leqslant 100$ mm
H: 1–100 mm
e: 10^{-6}–5 mm
f: 1 kHz to 10 MHz
dfil: 0.1–5 mm

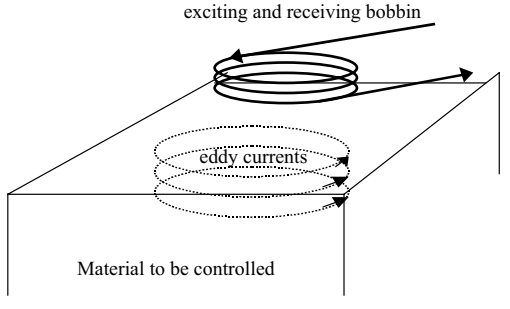

Fig. 10. Schematic drawing of a double function sensor.

6.3. Presentation of the applications treated within the framework of this work

Within the framework of this work, we will optimize an inversion model, using a sensor on a conducting plate. The model enables us to determine the parameters of a bobbin designed to control a material, thus determining if it contains a defect due to wear or corrosion. In order to determine the parameters of this bobbin, the study of $\Delta Z/Z$ is performed.

The objective of this study is to detect deep defects in non-magnetic strongly conducting materials. The modeling of the defects is very complex and involves:

- a truly 3D problem (simultaneously taking into account the device to be controlled and the defect),
- physical properties which are not easily measurable,
- a complex geometrical structure; in particular, the thickness is small (about some micrometers): accordingly, we cannot consider a real defect as a homogeneous volume (for example, a lack of matter).

In the case of a plate, the problem, fundamentally, consists of optimizing the maximum of $\Delta Z/Z$, which is the ratio between the variation of impedance (between the case with defect and the case without defect) and the impedance of the case without defect, according to the constitutive parameters of the bobbin (interior radius R_{int}, exterior radius R_{ext}, height H, air-gap e, frequency f and spire gap dfil). The geometry of the sensor influences the sensitivity of control. The influence of the frequency is directly related to the characteristics (conductivity and permeability) of the metal plate to be controlled.

Parameters R and X depend on the bobbin geometrical properties and on the distribution of the eddy currents inside the material. To be independent of the properties of the bobbin, we use the concept of standardized impedance.

(r_0, l) : punctual coordinates of the bobbin
Rint : internal radius of the bobbin
Rext : external radius of the bobbin
l_1 : coordinate according to z (the top of the bobbin)
l_2 : coordinate according to z (the bottom of the bobbin)
H : height of the bobbin
e : air-gap
C : thickness of the conductor
D : thickness of the defect

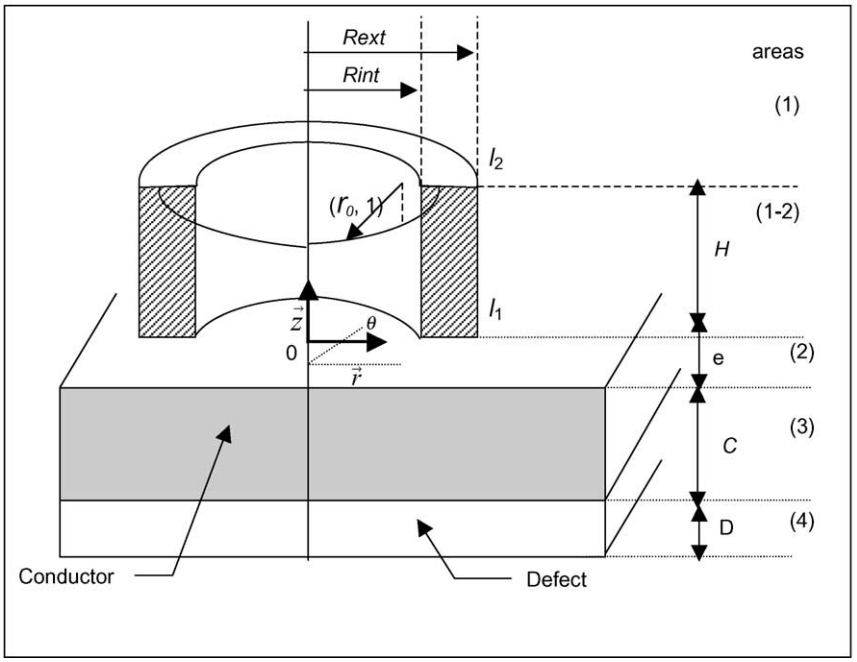

Fig. 11. Bobbin on plate with a defect.

6.4. Normalization of the impedance

Eddy currents are obtained in response to a sinusoidal current with a given frequency. The control is carried out by measurement of the bobbin impedance. A characteristic of the eddy currents distribution in a material is the equivalent impedance Z of the sensor, defined as follows:

$$Z = R + \mathrm{j}X,$$

where R is the resistance and X the reactance. Magnitudes of R and X depend on geometrical properties of the bobbin and the eddy currents distribution in the material. To free oneself from the own properties of the bobbin, we use the concept of ''standardized impedance''.

The following magnitudes are used:

- X_0: reactance of the bobbin alone,
- X: reactance of the bobbin in the presence of a material,
- R_0: resistance of the bobbin alone,
- R: resistance of the bobbin in the presence of a material.

We call R_N and X_N the standardized values of R and X, defined as follows:

- the normalized resistance: $R_N = \frac{R-R_0}{X_0}$,
- the normalized reactance: $X_N = \frac{X}{X_0}$.

For the detection of the defects the determination of ΔR_N and ΔX_N are required where:

$$\Delta R_N = R_N(\text{without defect}) - R_N(\text{with defect}),$$

$$\Delta X_N = X_N(\text{without defect}) - X_N(\text{with defect}).$$

Fig. 11 gives the drawing of a specific punctual drill on a plate including a defect within its depth.

6.5. Normalization of variables

The diversity of variables, as well as the disparity in their variation domains (some millimeters for an air-gap, and of the order of MHz for a frequency) lead us to normalize these parameters, by transforming their variation domains into the same domain $[0,1]$. Such normalization allows a better exploration of the solution domain. Variables recover their real values by inverse transformation when evaluating the objective function.

According to the extent of the variable variations, two kinds of normalizations can be performed:

Linear normalization: for variables ranging over a few decades, $[\boldsymbol{x}_{\text{real\_min}}(\boldsymbol{i}), \boldsymbol{x}_{\text{real\_max}}(\boldsymbol{i})]$. The normalized value $\boldsymbol{x}_{\text{norm}}(\boldsymbol{i})$ for the variable $\boldsymbol{x}_{\text{real}}(\boldsymbol{i})$ is given by

$$x_{\text{norm}}(i) = \alpha_{\text{norm}}(i) \cdot x_{\text{real}}(i) + \beta_{norm}(i),$$

$$\text{with} \quad \alpha_{\text{norm}}(i) = \frac{1}{x_{\text{real\_max}}(i) - x_{\text{real\_min}}(i)},$$

$$\text{and} \quad \beta_{\text{norm}}(i) = \frac{-x_{\text{real\_min}}(i)}{x_{\text{real\_max}}(i) - x_{\text{real\_min}}(i)}.$$

The inverse operation is given by

$$x_{\text{real}}(i) = \alpha_{\text{denorm}}(i) \cdot x_{\text{norm}}(i) + \beta_{\text{denorm}}(i)$$

$$\text{with} \quad \alpha_{\text{denorm}}(i) = \frac{1}{\alpha_{\text{norm}}(i)} = x_{\text{real\_max}}(i) - x_{\text{real\_min}}(i),$$

$$\text{and} \quad \beta_{\text{denorm}}(i) = \frac{-\beta_{\text{norm}}(i)}{\alpha_{\text{norm}}(i)} = x_{\text{real\_min}}(i).$$

Logarithmic normalization: for variables ranging over several decades (typically, more than 5 decades). The normalized value of the variable $\boldsymbol{x}_{\text{real}}(i)$ is given by

$$x_{\text{norm}}(i) = \gamma_{\text{norm}}(i) \cdot \log(x_{\text{real}}(i)) + \delta(i),$$

$$\text{with} \quad \gamma_{\text{norm}}(i) = \frac{1}{\log(x_{\text{real\_max}}(i)) - \log(x_{\text{real\_min}}(i))},$$

$$\text{and} \quad \delta(i) = \frac{-\log(x_{\text{real\_min}}(i))}{\log(x_{\text{real\_max}}(i)) - \log(x_{\text{real\_min}}(i))}.$$

The inverse operation is defined as follows:

$$x_{\text{real}}(i) = \exp[(x_{\text{norm}}(i) - \delta(i)) \cdot \gamma_{\text{denorm}}(i) \cdot \text{Log}(10)],$$

$$\text{with} \quad \gamma_{\text{denorm}}(i) = \frac{1}{\alpha n(i)} = \log(x_{\text{real\_max}}(i)) - \log(x_{\text{real\_min}}(i)).$$

Log and log stand, respectively, for Napierian and decimal logarithms.

6.6. Results obtained with CTSS to optimize the function $\Delta Z/Z$

After eliminating the impossible results, corresponding to non-realizable geometrical configurations (internal radius of the bobbin equal to its external radius, and so on), we obtain the results presented in Table 5: the bobbin parameters are those listed above; NbIter stands for the number of iterations and FOBJ for the best objective function value.

Contrary to the case of analytical objective functions studied in Section 5, the minimal value of the objective function FOBJ $= \Delta Z/Z$ is unknown. Consequently, we cannot consider the rate of successful minimizations of a given method, because we cannot define the threshold of acceptance of a solution. Table 5 shows the average results of 100 executions for each algorithm. All algorithms achieved FOBJ value lower than 20, except for the simplex search method (SS), which was trapped into local minima.

For this kind of objective function, the "pure" tabu search is the fastest method; when it is coupled with the local Nelder–Mead technique, it becomes more accurate. Algorithm CTSS (here $\text{CTSS}_{\text{multiple}}$)

Table 5
Results obtained by 6 algorithms for the optimization of $\Delta Z/Z$

Method	R_{int} (mm)	R_{ext} (mm)	H (mm)	e (mm)	f (kHz)	dfil (mm)	NbIter	FOBJ
SS	14.033	147.422	7.307	0.582	30.059	0.471	120	38.578
CGA	28.391	95.811	10.101	0.172	58.745	0.393	450	19.064
CHA	44.225	126.202	10.105	0.129	30.755	0.551	253	14.953
ECTS	32.931	78.935	11.514	0.121	30.000	0.321	199	15.013
CTSS	33.328	105.381	10.121	0.023	30.000	0.473	189	14.322
ESA	34.875	115.377	15.629	0.640	50.660	0.320	620	16.535

is clearly the most efficient method from the speed and accuracy points of view.

7. Conclusion

Generally hybrid methods achieve better solutions than "pure" methods, and converge more quickly. Among pure global methods, the genetic algorithm is the least accurate, and the tabu search is the fastest. If the genetic algorithm handles a population well distributed within the solution space at the beginning of the search, it localizes the promising zones more quickly than tabu search and simulated annealing, but its convergence towards the solutions is slower.

In this paper, we showed that tabu search can be efficiently applied to the optimization of continuous multiminima functions. Our main contribution is the adaptation to the continuous case of two concepts widely used in combinatorial tabu search, namely diversification and intensification. We proposed diversification for the detection of promising areas, and intensification for searching within the most promising area. We implemented two simple versions of the neighborhood. We consider that the first version of CTSS ($CTSS_{single}$) is simpler than the second version ($CTSS_{multiple}$), but both versions are a lot simpler than other tabu search algorithms using sophisticated neighborhood structures.

For functions of less than 10 variables, $CTSS_{multiple}$ provides similar or better results that other methods or other versions of continuous tabu search.

$CTSS_{multiple}$ avoids CPU time costly partitions of the solution space and too sophisticated approaches for the neighborhood structure. In consequence, $CTSS_{multiple}$ can be applied to functions having a large number of variables without prohibitive increase of the CPU time. Furthermore, $CTSS_{multiple}$ is much simpler that other versions of continuous tabu search, because it is straightforwardly derived from classical combinatorial tabu search, and uses the Simplex Search during the intensification phase.

It can be pointed out that the results we obtained for the design of a eddy current sensor have been validated by experts relating to that field: bobbins having geometric parameters similar to those displayed on Table 5 (large surface, small thickness) are currently used, under the "pancake" denomination.

Appendix A. List of test functions

Branin RCOS (RC) (2 variables):
$\mathrm{RC}(x_1,x_2) = (x_2 - (5/(4\cdot\pi^2))\cdot x_1^2 + (5/\pi)\cdot x_1 - 6)^2 + 10(1-(1/(8\cdot\pi)))\cdot\cos(x_1) + 10;$
search domain: $-5 < x_1 < 10$, $0 < x_2 < 15$;
no local minimum;
3 global minima: $(x_1,x_2)^* = (-\pi, 12.275)$, $(\pi, 2.275)$, (9.42478, 2.475);
$\mathrm{RC}((x_1,x_2)^*) = 0.397887.$

B2 (2 variables):
$B2(x_1,x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7;$
search domain: $-100 < x_j < 100$, $j = 1,2$;
several local minima (exact number unspecified in usual literature);
1 global minimum: $(x_1,x_2)* = (0,0)$; $B2((x_1, x_2)*) = 0.$

Easom (ES) (2 variables):

$\mathrm{ES}(x_1, x_2) = -\cos(x_1)\cdot\cos(x_2)\cdot\exp(-((x_1-\pi)^2+(x_2-\pi)^2))$;
search domain: $-100 < x_j < 100$, $j = 1, 2$;
several local minima (exact number unspecified in usual literature);
1 global minimum: $(x_1, x_2)^* = (\pi, \pi)$;
$\mathrm{ES}((x_1, x_2)^*) = -1$.

Goldstein and Price (GP) (2 variables):

$\mathrm{GP}(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2 * (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] * [30 + (2x_1 - 3x_2)^2 * (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$;
search domain: $-2 < x_j < 2$, $j = 1, 2$;
4 local minima;
1 global minimum: $(x_1, x_2)^* = (0, -1)$;
$\mathrm{GP}((x_1, x_2)^*) = 3$.

Shubert (SH) (2 variables):

$\mathrm{SH}(x_1, x_2) = \left\{\sum_{j=1}^{5} j\cos[(j+1)x_1 + j]\right\} * \left\{\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]\right\}$;
search domain: $-10 < x_j < 10$, $j = 1, 2$;
760 local minima;
18 global minima: $\mathrm{SH}((x_1, x_2)^*) = -186.7309$.

De Joung (DJ) (3 variables):

$\mathrm{ES}(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$;
search domain: $-5.12 < x_j < 5.12$, $j = 1, 3$;
1 single minimum (local and global): $(x_1, x_2, x_3)^* = (0, 0, 0)$; $\mathrm{ES}((x_1, x_2, x_3)^*) = 0$.

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

$H_{3,4}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right]$;
search domain: $0 < x_j < 1$, $j = 1, 3$;
4 local minima: $\mathbf{p_i} = (p_{i1}, p_{i2}, p_{i3}) = i$th local minimum approximation; $f((\mathbf{p}_i)) \cong -c_i$;
1 global minimum: $\mathbf{x}^* = (0.11, 0.555, 0.855)$;
$H_{3,4}(\mathbf{x}^*) = -3.86278$.

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

$S_{4,n}(\mathbf{x}) = -\sum_{i=1}^{n}[(\mathbf{x} - \mathbf{a}_i)^{\mathrm{T}}(\mathbf{x} - \mathbf{a}_i) + c_i]^{-1}$;
$\mathbf{x} = (x_1, x_2, x_3, x_4)^{\mathrm{T}}$;
$\mathbf{a_i} = (a_i^1, a_i^2, a_i^3, a_i^4)^{\mathrm{T}}$;
3 functions $S_{4,n}$ were considered: $S_{4,5}$, $S_{4,7}$ and $S_{4,10}$;
search domain: $0 < x_j < 10$, $j = 1, \ldots, 4$;
n local minima ($n = 5, 7$ or 10): $\mathbf{a_i}^{\mathrm{T}} = i$th local minimum approximation: $S_{4,n}((\mathbf{a_i}^{\mathrm{T}})) \cong 1/c_i$;

$S_{4,5}$	$n = 5$	5 minima with 1 global minimum: $S_{4,5}(\mathbf{x}) = -10.1532$
$S_{4,7}$	$n = 7$	7 minima with 1 global minimum: $S_{4,7}(\mathbf{x}) = -10.40294$
$S_{4,10}$	$n = 10$	10 minima with 1 global minimum: $S_{4,10}(\mathbf{x}) = -10.53641$.

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):

$H_{6,4}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2]$;
search domain: $0 < x_j < 1$, $j = 1, 6$;
4 local minima: $\mathbf{p_i} = (p_{i1}, \ldots, p_{i6}) = i$th local minimum approximation; $f((\mathbf{p_i})) \cong -c_i$;
1 global minimum: $\mathbf{x}^* = (0.20169, 0.150011, 0.47687, 0.275332, 0.311652, 0.6573)$;
$H_{6,4}(\mathbf{x}^*) = -3.32237$.

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

i	a_i						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

Rosenbrock (R_n (n variables):

$R_n(\mathbf{x}) = \sum_{j=1}^{n-1} [100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2]$;
5 functions were considered: R_2, R_5, R_{10}, R_{50} and R_{100};
search domain: $-5 < x_j < 10$, $j = 1, \ldots, n$;
several local minima (exact number unspecified in usual literature);
1 global minimum: $\mathbf{x}^* = (1, \ldots, 1)$;
$R_n(\mathbf{x}^*) = 0$.

Zakharov (Z_n) (n variables):

$Z_n(\mathbf{x}) = (\sum_{j=1}^{n} x_j^2)$
$+(\sum_{j=1}^{n} 0.5jx_j)^2 + (\sum_{j=1}^{n} 0.5jx_j)^4$;
5 functions were considered: Z_2, Z_5, Z_{10}, Z_{50} and Z_{100};
search domain: $-5 < x_j < 10$, $j = 1, \ldots, n$;
several local minima (exact number unspecified in usual literature);
1 global minimum: $\mathbf{x}^* = (0, \ldots, 0)$; $Z_n(\mathbf{x}^*) = 0$.

References

[1] R. Battiti, G. Tecchiolli, The continuous reactive tabu search: Blending combinatorial optimization and stochastic search for global optimization, Annals of Operations Research 63 (1996) 53–188.

[2] G.L. Bilbro, W.E. Snyder, Optimization of functions with many minima, IEEE Transactions on Systems, Man, and Cybernetics 21 (4) (1991) 840–849.

[3] J.A. Bland, Nonlinear optimization of constrained functions using tabu search, International Journal of Mathematical Education, Science and Technology 24 (5) (1993) 741–747.

[4] J.A. Bland, A derivative-free exploratory tool for function minimization based on tabu search, Advances in Engineering Software 19 (1994) 91–96.

[5] R. Chelouah, P. Siarry, Enhanced continuous tabu search: An algorithm for the global optimization of multiminima functions, in: S. Voss, S. Martello, I.H. Osman, C. Roucairol (Eds.), Meta-heuristics, Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, 1999, pp. 49–61 (Chapter 4).

[6] R. Chelouah, P. Siarry, A continuous genetic algorithm designed for the global optimization of multimodal functions, Journal of Heuristics 6 (2) (2000) 191–213.

[7] R. Chelouah, P. Siarry, Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions, European Journal of Operational Research 148 (2003) 335–348.

[8] D. Cvijovic, J. Klinowski, Taboo search. An approach to the multiple minima problem, Science 667 (1995) 664–666.

[9] L.C.W. Dixon, G.P. Szegö (Eds.), Towards Global Optimization, vol. 2, North-Holland, Amsterdam, 1978.

[10] F. Glover, Tabu search. Part I, ORSA Journal on Computing 1 (3) (1989) 190–206.

[11] F. Glover, Tabu search. Part II, ORSA Journal on Computing 2 (1) (1990) 4–32.

[12] N. Hu, Tabu search method with random moves for globally optimal design, International Journal for Numerical Methods in Engineering 35 (1992) 1055–1070.

[13] J.A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal 7 (1965) 308–313.

[14] W.H. Press, B.P. Flanery, S.A. Teukolsky, W.T. Vatterling, Numerical recipes in C, in: The Art of Science Computing, Cambridge University Press, New York, 1988.

[15] W.H. Press, S.A. Teukolsky, Simulated annealing optimization over continuous spaces, Computer in Physics (Jul/Aug) (1991) 427–429.

[16] J.M. Renders, S.P. Flasse, Hybrid method using genetic algorithms for the global optimization, IEEE Transactions on Systems, Man, and Cybernetics 26 (2) (1996) 243–258.

[17] K. Schittkowski, W. Hock, Test examples for nonlinear programming codes, in: Lecture Notes in Economics Math. Syst, vol. 187, Springer-Verlag, 1981.

[18] K. Schittkowski, W. Hock, More test examples for nonlinear programming codes, in: Lecture Notes in Economics Math. Syst, vol. 282, Springer-Verlag, 1987.

[19] P. Siarry, G. Berthiau, Fitting of tabu search to optimize functions of continuous variables, International Journal for Numerical Methods in Engineering 40 (1997) 2449–2457.

[20] P. Siarry, G. Berthiau, F. Durbin, J. Haussy, Enhanced simulated annealing for globally minimizing functions of many continuous variables, ACM Transactions of Mathematical Software 23 (2) (1997) 209–228.