International Journal of Systems Science, 2001, volume 32, number 1, pages 91–102

A hybrid algorithm for infinitely constrained optimization

CORRADO GUARINO LO BIANCO† and AURELIO PIAZZI†

Infinitely constrained (or semi-infinite) optimization can be successfully used to solve a significant variety of optimization-based engineering design problems. In this paper a new algorithm for the numerical global solution of nonlinear and nonconvex, infinitely constrained problems is proposed. At the upper level this hybrid algorithm is a partially elitist genetic algorithm that uses, at the lower level, an interval procedure to compute a penalty-based fitness function. The deterministic nature of the interval procedure, whose global convergence with certainty is established by using concepts of interval analysis, guarantees the feasibility of the estimated global solution provided by the hybrid algorithm. Computational results are reported for three test problems and the hybrid algorithm is applied to the optimal worst-case H_2 design of a proportional–integral–derivative (PID) controller for an uncertain nonminimum-phase plant.

1. Introduction

In the literature it has already been recognized that many optimization-based engineering designs can be transformed into infinitely constrained optimization problems (Polak 1987, Hettich and Kortanek 1993). In particular, in control systems design it would be especially useful to have an efficient tool to solve nonlinear nonconvex semi-infinite problems (Polak *et al.* 1984). With this aim, consider the following optimization problem:

$$\min_{x\in\mathcal{X}} \{f(x) : g_i(x,q) \leqslant 0 \ \forall q \in \mathcal{Q}, \quad i = 1,2,\ldots,l\}, \quad (1)$$

where $f(x)$ is the objective function and $g_i(x,q), i = 1,\ldots,l$, are the semi-infinite constraint functions with $\mathcal{X}$ and $\mathcal{Q}$ being compact multidimensional intervals of $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively. All the functions involved in (1) are, in general, nonlinear and non-convex with the constraint functions assumed to be continuously differentiable. No particular assumptions are made upon the objective function which can be, for example, discontinuous and may admit many local minima into the feasibility region of $\mathcal{X}$, denoted by

$$F_{\mathcal{X}} := \{x \in \mathcal{X} : g_i(x,q) \leqslant 0 \quad \forall q \in \mathcal{Q}, i = 1,2,\ldots,l\}.$$

The problem addressed in this paper is to obtain a global solution for (1), that is to find $x^* \in F_{\mathcal{X}}$ such that $f(x^*) \leqslant f(x)$ for all $x \in F_{\mathcal{X}}$.

Existing algorithms for solving (1) can be roughly divided into two broad classes: firstly, descent methods for non-differentiable optimization and, secondly, recursive nonlinear programming methods. The former methods are mainly based on generalized gradients used with extended steepest-descent procedures; the papers by Polak (1987), Polak and Mayne (1985) and Polak and Wardi (1992) contain numerous references on this subject. The latter methods are based on the reduction to a sequence of constrained nonlinear programming problems, for example exchange methods, discretization methods, and sequential quadratic programming; see the paper by Hettich and Kortanek (1993) which has a fairly complete bibliography on the subject and Fiacco and Kortanek (1983) and Hettich (1979) for further coverage. All these algorithms, apart from invoking special assumptions (e.g. linearity and convexity), can only determine local solutions for (1).

In this paper a global approach to solve (1) is pursued by means of a hybrid algorithm which benefits from the combined use of tools borrowed from stochastic global optimization and deterministic global optimization, specifically genetic algorithms and interval algorithms. The former are designed by mimicking biological evolution: populations of individuals, representing tentative solutions for the given problem, evolve over the generations to maximize a 'fitness' function which captures the aim of the problem. Genetic algorithms started with a

Received 23 June 1988. Revised 28 June 1999. Accepted 25 November 1999.

† Dipartimento di Ingegneria dell'Informazione, Università di Parma, Parco Area delle Scienze, 181/A, I-43100 Parma, Italy.

fundamental study on artificial adaptation by Holland (1975), which was followed by many studies on both optimization and artificial intelligence applications (Goldberg 1989, Davis 1991, Michaelwicz 1992).

The method based on interval algorithms is one of the most promising approaches to deterministic optimization. It relies on the mathematical tools of the interval analysis (Moore 1996), which is an extension of the 'standard' analysis over the arithmetic of intervals of $\mathbb{R}$. Excellent references on these global optimization techniques are the books of Ratschek and Rokne (1988) and Hansen (1992). Recent applications of interval algorithms to control theory and trajectory planning have been given by Jaulin and Walter (1996), Piazzi and Marro (1996), Malan *et al.* (1997) and Piazzi and Visioli (1998).

Problem (1) is clearly equivalent to the following non-smooth constrained optimization problem:

$$\min_{x\in\mathcal{X}} \{f(x) : \sigma_i(x) \leqslant 0, \quad i = 1,\ldots,l\}, \tag{2}$$

with $\sigma_i(x) := \max_{q\in\mathcal{Q}} \{g_i(x,q)\}$. Problem (2) can be transformed, by means of a penalty method, into the unconstrained minimization problem

$$\min_{x\in\mathcal{X}} \left\{ f(x) + \sum_{i=1}^{l} \Phi(\sigma_i(x)) \right\}, \tag{3}$$

where the penalty function $\Phi(\cdot)$ is parametrized with $M, T > 0$ and given by

$$\Phi(\sigma) = \begin{cases} 0 & \text{if } \sigma \leqslant 0, \\ \varphi(\sigma) & \text{if } 0 < \sigma < T, \\ M & \text{if } \sigma \geqslant T, \end{cases} \tag{4}$$

with $\varphi(\sigma)$ being a monotonically increasing continuous user-chosen function satisfying the continuity conditions $\varphi(0) = 0$ and $\varphi(T) = M$ (see remark 1 in § 2.2).

A solution to the unconstrained problem (3) constitutes an approximate solution for (2) and hence for (1), which is as close as desired to an exact solution x^*, provided that M is sufficiently large while T is sufficiently small.

Roughly speaking, the proposed approach is the following. A genetic algorithm is adopted to solve the unconstrained problem (3) and the necessary computation of the penalty term $\Phi(\sigma_i(x))$ is performed by a specific interval procedure. In many cases, this procedure, owing to the structure of function $\Phi(\sigma)$, can compute $\Phi(\sigma_i(x))$ directly as a whole, that is without determining explicitly $\sigma_i(x)$ (see § 3). Indeed, it is not necessary to evaluate $\sigma_i(x)$, which corresponds to a single global bound-constrained maximization problem, for any required $x \in \mathcal{X}$ with the sole exception for the points lying in proximity of the boundary defined by $\sigma_i(x) = 0$. This feature of the interval procedure is decisive for the practical effectiveness of the proposed hybrid algorithm.

It would be conceptually possible to try to solve problem (1) by devising a purely deterministic global method (e.g. an interval algorithm) or a purely stochastic global method (e.g. a genetic algorithm). In the former case the algorithm evaluates arbitrarily good lower and upper bounds of the global extremum $f(x^*)$ and converges with certainty to a global solution x^*. Unfortunately this approach has a prohibitive computational complexity so that it would be hardly useful for anything but trivial problems. In the latter case, by using the penalty formulation (3), a sequence of maximization problems is solved by means of a stochastic global algorithm and then the final minimization problem can be solved with the same algorithm. The stochastic global convergence of the algorithm does not offer guaranteed lower and upper bounds of the global extremum at any given stage of iterations (see the paper by Rudolph (1994) on the concept of stochastic convergence to the global optimum for canonical genetic algorithms) but the main drawback of the purely stochastic approach applied to problem (1) is on the final solution that may not be feasible. Moreover, the feasibility of the final solution cannot be ascertained with sureness. On the contrary, the hybrid algorithm proposed in this paper, because of the deterministic interval procedure, guarantees the feasibility of the obtained final solution. On the other hand, this hybrid approach cannot again determine guaranteed lower and upper bounds of $f(x^*)$ but it offers an estimated global solution that can often be, in practice, an excellent solution of (1) (see § 4). Moreover, the related computational burden is moderate, permitting non-trivial problems to be solved, as long as the dimension of $\mathcal{Q}$ is low (e.g. $m \leqslant 3$–5). Informally, this hybrid algorithm can be viewed as an 'optimal' mix of the stochastic and deterministic approaches that is particularly useful when feasibility is a crucial aspect of the underlying problem, such as in control system design problems where robust stability has to be guaranteed (see § 4.2).

The paper is organized as follows. In § 2 the partially elitist genetic algorithm for inequality constrained optimization, which is the upper level of the hybrid algorithm, is given. In § 3 the interval procedure to compute $\Phi(\sigma_i(x))$ is reported and the global convergence of such a procedure is established (theorem 1). Application examples are included in § 4. In § 4.1, computational results for three test problems including a multimodal example (i.e. a problem with many local minima) are reported. In § 4.2 the application of the hybrid algorithm to the optimal worst-case H_2 design of a proportional–integral–derivative (PID) controller for an uncertain nonminimum-phase plant is presented. Conclusions are drawn in § 5.

2. A genetic approach for inequality-constrained optimization

In the following the genetic algorithm is designed to solve inequality constrained optimization. It uses a two-phases procedure for searching the feasible optimal solution. First it neglects the objective function $f(x)$ and attempts to find a population whose individuals satisfy all constraints. Later, if the feasible region $F_{\mathcal{X}}$ is reached, it starts solving the complete problem (3). This two-phase idea originated in the work of Box (1965) and was reproposed by Schwefel (1977, pp. 134–135, 176) in the context of evolutionary algorithms and by Schoenauer and Xanthakis (1993) for genetic algorithms.

In this work the genetic algorithm is based on the concept of partial elitism (already successfully used by Menozzi *et al.* (1996)) and on a dynamically scaled penalty-based fitness function. The genetic algorithm has been developed to deal with semi-infinite optimization but, obviously, it can also be used for standard (finite) inequality-constrained optimization. The outline of the algorithm is given below followed by explanations in §§ 2.1–2.8.

Algorithm:

Step 1. Generate randomly the initial population $\mathbf{P}$ composed of γ individuals (phase I starts).

Step 2. Evaluate or update the fitness of each individual of $\mathbf{P}$.

Step 3. Select, with a stochastic fitness-biased procedure, $\rho \leqslant \gamma$ individuals of $\mathbf{P}$ to create the offspring population $\mathbf{P}'$.

Step 4. Apply genetic operators (crossover and mutation).

Step 5. Evaluate the fitness of $\mathbf{P}'$ and, if phase II is active, update the fitness of $\mathbf{P}$ (see § 2.2).

Step 6. Apply a local improvement operator to the best individual of $\mathbf{P}'$.

Step 7. From $\mathbf{P}$ randomly choose $\eta\gamma$ individuals such that $\rho + \eta\gamma \geqslant \gamma$ and add them to $\mathbf{P}'$ ($\eta \in [0,1]$ is the elitist factor; see § 2.3).

Step 8. Apply a local improvement operator to the best ξ individuals of $\mathbf{P}'$.

Step 9. Replace the population $\mathbf{P}$ with the best γ individuals of $\mathbf{P}'$.

Step 10. If phase I is active and more than $\lambda\%$ individuals of $\mathbf{P}$ are feasible, switch to phase II.

Step 11. If the stopping criterion is not satisfied go to step 2.

Step 12. Display the final report and exit.

2.1. *The population*

A simple binary encoding has been adopted to represent the real vector $x := [x_1 \cdots x_n]^{\mathrm{T}} \in \mathcal{X}$. The coded representation of x is stored into a vector of integers and denoted by $\underline{x} = [\underline{x}_1 \cdots \underline{x}_n]^{\mathrm{T}} \in \mathbf{P}$. Each component of x is coded by using b_{c} bits, so that the underlying chromosome structure of $\underline{x}$ is equivalent to a string of nb_{c} genes.

2.2. *The fitness*

Two different fitness functions are used, depending on the advancement of the algorithm. During phase I the fitness function is

$$\Gamma := \sum_{i=1}^{l} [M - \Phi(\sigma_i(x))]. \tag{5}$$

At this stage the choice of M is not critical and, for example, it can be set to unity; the 'threshold' parameter T is here set to a relative large T_{I}. Phase I terminates if at least $\lambda\%$ individuals of $\mathbf{P}$ are feasible: $\underline{x} \in \mathbf{P}$ is feasible when $\Phi(\sigma_i(x)) = 0$ for all $i = 1, \ldots, l$.

With phase II the fitness function is

$$\Gamma := [N - f(x)] + \sum_{i=1}^{l} [M - \Phi(\sigma_i(x))]. \tag{6}$$

During this phase the threshold T is set to a small constant T_{II} compatible with the required precision. The parameters N and M are calculated dynamically at each iteration. At step 2 the fitness evaluation is based on

$$N := \max_{\underline{x} \in \mathbf{P}} \{f(x)\} \quad M := \max_{\underline{x} \in \mathbf{P}} \{N - f(x)\}, \tag{7}$$

whereas at step 5

$$N := \max_{\underline{x} \in \mathbf{P} \cup \mathbf{P}'} \{f(x)\}, \quad M := \max_{\underline{x} \in \mathbf{P} \cup \mathbf{P}'} \{N - f(x)\}, \tag{8}$$

with x being the point of $\mathcal{X}$ corresponding to the coded individual $\underline{x}$ of $\mathbf{P}$ or $\mathbf{P}'$. The rules (7) and (8) guarantee that all $l+1$ terms composing the fitness are not negative and equally scaled so that the objective and constraint functions are properly emphasized.

Remark 1: The given definitions of the fitness function in phase I and phase II show the reason for adopting a parametrized continuous penalty function $\Phi(\sigma)$ instead of a standard discontinuous function (usually defined as $\Phi(\sigma) = 0$ if $\sigma \leqslant 0$ and $\Phi(\sigma) = M > 0$ if $\sigma > 0$). Indeed, in order to foster the emergence of potentially good

individuals over the generations, it is useful especially during phase I, but also to a certain extent during phase II, not to penalize individuals heavily that are close to the feasible region $F_{\mathcal{X}}$.

2.3. *The partially elitist model*

A partially elitist model is adopted. The step towards obtaining the next population $\mathbf{P}$ is the merging of the offspring population $\mathbf{P}'$ with a given percentage of individuals randomly drawn from the current population $\mathbf{P}$. This is done by using the elitist factor $\eta \in [0,1]$. Finally, after the local improvement in step 8, the deterministic selection of the best γ merged individuals defines the new population $\mathbf{P}$. In practice, the appropriate tuning of η permits obtaining a reasonably high probability of converging to a global solution within a moderate number of generations.

2.4. *Selection*

A classic roulette wheel selection with a linear scaling fitness ranking is used (Goldberg 1989). The associated linear scaling coefficient is denoted by C_{sca}.

2.5. *Crossover*

Because of the partially elitist model, a crossover probability equal to unity 1 is chosen, that is the crossover operator acts over all individuals of the offspring population. The crossover adopted is an uniform multipoint crossover (Ackley 1987, p. 79). By considering $\underline{a}$ and $\underline{b}$, two randomly chosen individual of $\mathbf{P}'$ corresponding to points $a,b \in \mathcal{X}$, the crossover is made between the homologous components $\underline{a}_i$ and $\underline{b}_i$ as is shown in figure 1. The crossover point is chosen randomly for each component with an uniform probability distribution. This crossover operator has a neat geometrical insight. The aim is the same as pursued by Mühlenbein *et al.* (1991); depending on the crossover point, some components may change almost completely, so that new regions of the searching area are investigated, while some others may change just a little because only the less important bits are affected. In this way, some important information may be retained while new solutions are tested.

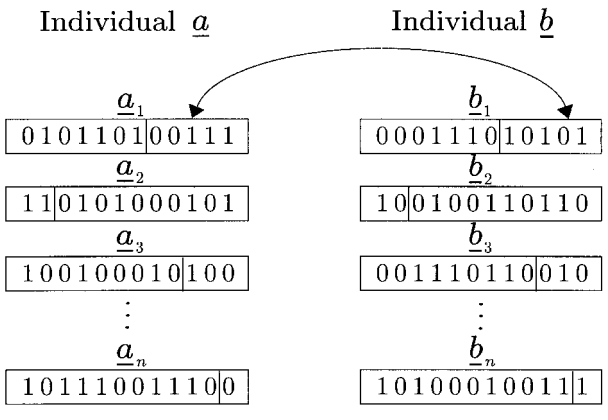

Figure 1. Example of crossover between two individuals.

2.6. *Mutation*

The adopted gene mutation probability is not uniform and, moreover, it depends on the generation being processed. At the generation g, the mutation probability of a generic gene at the position c in the chromosome string is given by the equation

$$P_{\text{mut}}(c,g) = P_{\text{M}}\, \mathrm{e}^{-[\delta(c,g)]^2}, \tag{9}$$

where P_{M} is the maximum allowed probability and

$$\delta(c,g) = \frac{3}{nb_{\text{c}}}\left(1 - c + \frac{nb_{\text{c}} - 1}{g_{\text{M}}}g\right). \tag{10}$$

In (10), g_{M} is the maximum allowed number of generations. The genes (bits) in the chromosome string are arranged with decreasing importance order; the most important bits are placed at the beginning of the string whereas the least important bits can be found at the end. To understand the meaning of (9), the shapes of P_{mut} corresponding to three different g values are shown in figure 2. During the first generations the mutation probability of the most important bits is the highest (about P_{M}) because the algorithm aims to investigate an area as wide as possible, while the least significant bits have a much smaller probability of mutating. The opposite situation can be observed in later generations.

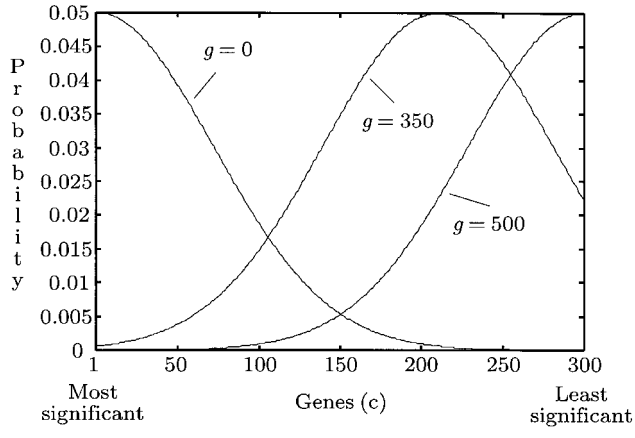

Figure 2. Mutation probability of gene c for three different generations. It is drawn with $P_{\text{M}} = 0.05$, $g_{\text{M}} = 500$ and $nb_{\text{c}} = 300$.

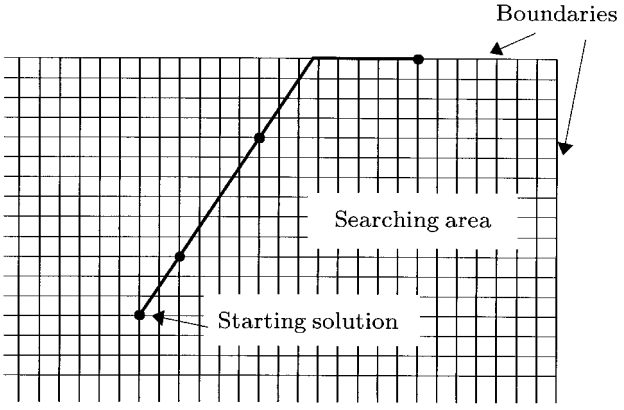

Figure 3. Example of local improvement for a two-dimensional x. In this case, $\upsilon = [2\ 3]^{\mathrm{T}}$ with $L_{\mathrm{step}} = 4$.

2.7. *Local improvement*

A local investigation is used to speed up the convergence of the algorithm. It is used two times during each iteration (steps 6 and 8). The first time, the local search attempts to improve the best individual in the offspring population $\mathbf{P}'$. The second time it works over the ξ best individuals of the final population. The search direction is chosen taking into account the discrete encoding of x. In figure 3 a two-dimensional example is shown. An integer $\upsilon_i, i = 1,2,\ldots,n$, is randomly drawn out, for each coordinate direction, from the interval $[-L_{\mathrm{step}}, L_{\mathrm{step}}]$ with uniform probability. The resulting vector $\upsilon = [\upsilon_1\ \upsilon_2\ \cdots\ \upsilon_n]^{\mathrm{T}}$ represents the searching direction taken from a starting discrete grid defined by L_{step}. If along that direction the fitness decreases, the opposite direction is selected. On the contrary, if the fitness increases, the step used for the next attempt is doubled. A particular solution is provided if the parallelepiped boundary of $\mathcal{X}$ is reached. In that case the algorithm projects the searching direction along that boundary and carries on trying to increase the fitness. The local investigation terminates if the fitness stops improving.

2.8. *Stopping criterion*

During phase I the algorithm ends only if the maximum number g_{M} of generations is reached. In that case the genetic algorithm has to warn that no optimal feasible solution has been found. During phase II, the stopping criterion is based on genetic saturation (Goldberg 1989). Considering the current population if the difference between the average population fitness and the fitness of the best individual is less than a given $\varepsilon_{\mathrm{end}}$, then the algorithm has to halt. The individual $\underline{x}^*$ which is the best, in terms of objective function, among the feasible individuals evaluated all over the generations of phase II is the estimated global solution provided by the genetic algorithm.

The presented partially elitist genetic algorithm can be considered as a generalization of the simple genetic algorithm of Goldberg (1989). For the latter, Rudolph (1994) presented a convergence analysis. He emphasized that stochastic convergence to the global optimum is secured provided that the best solution found over the generations is retained. The actual effectiveness of a given genetic algorithm depends on its parameter and probability rate settings (DeJong and Spears 1990). With regard to the exposed partially elitist genetic algorithm, experience in setting genetic parameter values has been gained by Menozzi *et al.* (1996) and Menozzi and Piazzi (1996) and refined with experiments on an optimization test battery taken from Mühlenbein *et al.* (1991).

3. The interval procedure for computing the penalty terms

The interval procedure to compute the penalty term $\Phi(\sigma(x))$ (for simplicity in this section the subscript i is dropped) is now determined for a generic semi-infinite constraint function $g(x,q)$.

3.1. *Notation*

The set for positive real numbers is $\mathbb{R}^+$. $\mathcal{Q}, \mathcal{D} \subseteq \mathbb{R}^m$ denote finite multidimensional real intervals or 'boxes'; considering the box $\mathcal{D} := [\underline{d}_1, \overline{d}_1] \times [\underline{d}_2, \overline{d}_2] \times \cdots \times [\underline{d}_m, \overline{d}_m]$, then $\mathrm{mid}(\mathcal{D}) \in \mathcal{D}$ is the so-called midpoint of $\mathcal{D}$ whose ith component is given by $(\underline{d}_i + \overline{d}_i)/2$ and $w(\mathcal{D}) := \max_{i=1,\ldots,m}\{\overline{d}_i - \underline{d}_i\}$ is the width of $\mathcal{D}$. The set of real intervals is denoted by $I := \{[a,b] : a,b \in \mathbb{R}, a \leqslant b\}$.

$g_x(q) \equiv g(x,q)$ denotes the semi-infinite constraint function for which x acts as fixed parameter. The image set of $\mathcal{D}$ under function $g_x(q)$ is denoted by

$$g_x(\mathcal{D}) := \{y \in \mathbb{R} : y = g_x(q), q \in \mathcal{D}\};$$

the continuity of $g_x(q)$ implies that $g_x(\mathcal{D}) \in I$. The global maximum value of g_x over $\mathcal{Q}$ is denoted by $g_x^* := \sigma(x) = \max_{q \in \mathcal{Q}}\{g_x(q)\}$; g_x^* is the upper endpoint of $g_x(\mathcal{Q})$. Moreover, $\mathcal{Q}^* := \{q^* \in \mathcal{Q} : g_x(q^*) = g_x^*\}$ denotes the set of global maximizers which may have cardinality greater than one. The lower and upper bounds of g_x^* are denoted by l_{b} and u_{b} respectively. $K_{\mathrm{pre}} \in \mathbb{N}$ denotes the precision factor to be used by the termination test of the interval procedure ($\mathbb{N}$ is the set for positive integers).

3.2. *Interval computation of upper bounds*

An *inclusion function* with respect to g_x is an interval-valued function $G_x : \{\mathcal{D} : \mathcal{D} \subseteq \mathcal{Q}\} \to I$ satisfying

$$g_x(\mathcal{D}) \subseteq G_x(\mathcal{D}) \quad \forall \mathcal{D} \subseteq \mathcal{Q}.$$

Once an inclusion function is known, an upper bound of the global maximum of $g_x(q)$ over $\mathcal{D} \subseteq \mathcal{Q}$, denoted by $\mathrm{ub}(g_x, \mathcal{D})$, can be easily determined as the upper endpoint of $G_x(\mathcal{D})$. Interval analysis is a straightforward tool to obtain a variety of inclusion functions. The simplest of these is the so-called natural interval extension. Roughly speaking, it is obtained by evaluating a given form of g_x with the substitution of the usual arithmetic with the interval arithmetic. This is summarized as follows:

$$[a,b] + [c,d] = [a+c, b+d],$$

$$[a,b] - [c,d] = [a-d, b-c],$$

$$[a,b][c,d] = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}],$$

$$\frac{[a,b]}{[c,d]} = [a,b] \cdot \left[\frac{1}{d}, \frac{1}{c}\right] \quad \text{if } 0 \notin [c,d].$$

As an example consider $g_x(q) = q_1 + 3q_1q_2^2 - q_1^2q_2$ and $\mathcal{D} = [0,2] \times [1,3]$. Hence, by using the natural interval extension we have

$$\begin{aligned} G_x(\mathcal{D}) &= [0,2] + 3[0,2][1,3]^2 - [0,2]^2[1,3] \\ &= [0,2] + 3[0,2][1,9] - [0,4][1,3] \\ &= [0,2] + [0,54] - [0,12] \\ &= [-12, 56]. \end{aligned}$$

Other noteworthy inclusion functions are the 'mean-value forms' and 'Taylor forms'; both forms belong to the class of 'centred forms' introduced by Moore (1966). For the purpose of the interval procedure to follow, the optimal mean-value form of Baumann has been chosen as the inclusion function for its sharp bounds and moderate computational burden. An introduction to inclusion functions with details on the Baumann form has been given by Ratschek and Rokne (1988).

3.3. *Interval procedure to compute* $\Phi(\sigma(x))$

Procedure:

Step 1. Locally maximize the function $g_x(q)$ with starting point $\mathrm{mid}(\mathcal{Q})$ to obtain $\widetilde{q}$ and set $l_b := g_x(\widetilde{q})$. ($\widetilde{q} \in \mathcal{Q}$ and $g_x(\widetilde{q}) \geqslant g_x(\mathrm{mid}(\mathcal{Q}))$.)

Step 2. Set $u_b := \mathrm{ub}(g_x, \mathcal{Q})$.

Step 3. Initialize the list, denoted *List*, inserting the pair $(\mathcal{Q}, u_b)$.

Step 4. If $l_b \geqslant T$, then set $\Phi(\sigma(x)) := M$ and terminate.

Step 5. If $u_b \leqslant 0$, then set $\Phi(\sigma(x)) := 0$ and terminate.

Step 6. If $u_b - l_b \leqslant T/K_{\mathrm{pre}}$, then set $\Phi(\sigma(x)) := \Phi((l_b + u_b)/2)$ and terminate.

Step 7. The first pair of the *List* is popped out and its box, by halving the largest edge, is split into boxes $\mathcal{D}_1$ and $\mathcal{D}_2$.

Step 8. For $i := 1,2$ do.
If $g_x(\mathrm{mid}(\mathcal{D}_i)) > l_b$, then locally maximize the function $g_x(q)$ with starting point $\mathrm{mid}(\mathcal{D}_i)$ to obtain $\widetilde{q}$ and set $l_b := g_x(\widetilde{q})$. ($\widetilde{q} \in \mathcal{Q}$ and $g_x(\widetilde{q}) \geqslant g_x(\mathrm{mid}(\mathcal{D}_i))$.)

Step 9. For $i := 1,2$ do.
If $\mathrm{ub}(g_x, \mathcal{D}_i) \geqslant l_b$, then insert pair $(\mathcal{D}_i, \mathrm{ub}(g_x, \mathcal{D}_i))$ in the *List*, in such a way that the second elements of the pairs be placed in decreasing order.

Step 10. Discard from the *List*, without perturbing the decreasing order, any pair such that its second element is smaller than l_b.

Step 11. Set $u_b :=$ the second element of the first pair of the *List*.

Step 12. Go to step 4.

Step 13. End.

Remark 2: The exposed procedure is based on the branch-and-bound principle where the bounding is made via inclusion functions and the branching is made by splitting the box which has the largest upper bound. In this way, at the core of the procedure, an interval algorithm emerges which can compute g_x^* with arbitrary precision. However, not always g_x^* is computed within the precision given by T/K_{pre} because of steps 4 and 5. □

Remark 3: The function $\Phi(\sigma)$, defined in (4), uses the parabolic interpolation given by $\varphi(\sigma) := M - M(T-\sigma)^2/T^2$ but other choices could be made as well. □

Remark 4: At steps 1 and 8 a local maximization is simply performed with the steepest-ascent method (Luenberger 1989, p. 214). This accelerates the procedure convergence because it helps to discard portions of $\mathcal{Q}$ not containing global maximizers (see step 10). □

3.4. *Convergence analysis*

The considered assumption of $g_x(q)$ being continuously differentiable on the compact $\mathcal{Q}$ implies that $g_x(q)$ is Lipshitzian as well as continuous on $\mathcal{Q}$. The

following properties are essential to establish the main result of this section (theorem 1).

Property 1: For any mean-value form the following limit holds uniformly for $D \subseteq Q$:

$$\lim_{w(\mathcal{D})\to 0} [\mathrm{ub}(g_x, \mathcal{D})] = \max_{q\in\mathcal{D}} \{g_x(q)\}; \tag{11}$$

moreover assuming that the gradient of $g_x(q)$ is Lipschitzian on $\mathcal{Q}$, limit (11) has convergence order 2. □

Property 2: The following limit holds uniformly for $D \subseteq Q$:

$$\lim_{w(\mathcal{D})\to 0} [g_x(\mathrm{mid}(\mathcal{D}))] = \max_{q\in\mathcal{D}} \{g_x(q)\}. \tag{12}$$

□

Property 1 is a well-known result in the interval analysis literature (see for example Ratschek and Rokne (1988), and property 2 is an obvious consequence of the continuity of function $g_x(q)$ over $\mathcal{Q}$.

Theorem 1: *For any $T \in \mathbb{R}^+$ and $K_{\mathrm{pre}} \in \mathbb{N}$ the interval procedure converges with certainty. At any stage of iterations, g_x^* belongs to $[l_b, u_b]$.*

The proof of theorem 1 can be derived from standard interval analysis reasoning (Ratschek and Rokne 1988). For reader's convenience a succinct self-contained proof follows.

Proof: The *List* at the ith iteration is composed of h_i boxes $\mathcal{B}_i$ such that:

$$\textit{List} = \{(\mathcal{B}_1^{(i)}, \mathrm{ub}_1^{(i)}), (\mathcal{B}_2^{(i)}, \mathrm{ub}_2^{(i)}), \ldots, (\mathcal{B}_{h_i}^{(i)}, \mathrm{ub}_{h_i}^{(i)})\},$$

where $\mathrm{ub}_j^{(i)} := \mathrm{ub}(g_x, \mathcal{B}_j^{(i)})$, $j = 1, 2, \ldots, h_i$. By virtue of steps 9 and 10 the upper bounds $\mathrm{ub}_j^{(i)}$ are placed in the *List* with decreasing order:

$$\mathrm{ub}_1^{(i)} \geqslant \mathrm{ub}_2^{(i)} \geqslant \mathrm{ub}_3^{(i)} \geqslant \cdots \geqslant \mathrm{ub}_{h_i}^{(i)}. \tag{13}$$

Also denote by $l_b^{(i)}$ and $u_b^{(i)}$ the values of variables l_b and u_b at the ith iteration. First note that $l_b^{(i)}$ is a lower bound of g_x^* at any iteration i: $l_b^{(i)} \leqslant g_x^*$. Indeed $l_b^{(i)}$ is the maximum of all the g_x function values computed up to the ith iteration.

The procedure searches exhaustively the box $\mathcal{Q}$. Specifically at the ith iteration, and subsequent iterations, the procedural action is performed over $\bigcup_{j=1}^{h_i} \mathcal{B}_j^{(i)} \subseteq \mathcal{Q}$ because, by virtue of steps 9 and 10, the discarded region $\mathcal{Q} - \bigcup_{j=1}^{h_i} \mathcal{B}_j^{(i)}$ does not contain any global maximizer, that is

$$q^* \notin \mathcal{Q} - \bigcup_{j=1}^{h_i} \mathcal{B}_j^{(i)}, \quad \forall q^* \in \mathcal{Q}^*. \tag{14}$$

Indeed a subbox of $\mathcal{Q}$ is discarded from the *List* only when the upper bound of the global maximum over such subbox is less than the lower bound $l_b^{(i)}$ of the global maximum over $\mathcal{Q}$ (bounding principle). As a consequence of (14) it is inferred that

$$\mathcal{Q}^* \subseteq \bigcup_{j=1}^{h_i} \mathcal{B}_j^{(i)}. \tag{15}$$

Step 11 determines the equality $u_b^{(i)} = \mathrm{ub}_1^{(i)}$, so that ordering (13) leads to $u_b^{(i)} = \max_{j=1,2,\ldots h_i} \{\mathrm{ub}_j^{(i)}\}$. Hence the

set inclusion (15) implies that $g_x^* \leqslant u_b^{(i)}$.

The branching mechanism issued at step 7, in the absence of the exit tests 4–6, permits us to write

$$\lim_{i\to\infty} [w(\mathcal{B}_1^{(i)})] = 0. \tag{16}$$

Property 1 implies that for any given $\varepsilon > 0$ there exists $\delta_u > 0$ such that for any $\mathcal{B}_1^{(i)}$ satisfying $w(\mathcal{B}_1^{(i)}) < \delta_u$ it follows that

$$\mathrm{ub}(g_x, \mathcal{B}_1^{(i)}) - \max_{q\in\mathcal{B}_1^{(i)}} \{g_x(q)\} < \varepsilon,$$

which is equivalent to

$$\mathrm{ub}_1^{(i)} - \varepsilon < \max_{q\in\mathcal{B}_1^{(i)}} \{g_x(q)\}. \tag{17}$$

On the other hand, with property 2, for any given $\varepsilon > 0$ there exists $\delta_m > 0$ such that for any $\mathcal{B}_1^{(i)}$ satisfying $w(\mathcal{B}_1^{(i)}) < \delta_m$ it follows that

$$\max_{q\in\mathcal{B}_1^{(i)}} \{g_x(q)\} - g_x(\mathrm{mid}\,(\mathcal{B}_1^{(i)})) < \varepsilon,$$

$$\max_{q\in\mathcal{B}_1^{(i)}} \{g_x(q)\} < \varepsilon + g_x(\mathrm{mid} < (\mathcal{B}_1^{(i)})). \tag{18}$$

By considering that function $g_x(q)$ is evaluated at the midpoint of every box inserted in the *List* (see steps 1 and 8), it is clearly verified that $g_x(\mathrm{mid}\,(\mathcal{B}_1^{(i)})) \leqslant l_b^{(i)}$. Then inequality (18) implies that

$$\max_{q\in\mathcal{B}_1^{(i)}} \{g_x(q)\} < \varepsilon + l_b^{(i)}. \tag{19}$$

The limit (16) assures the existence of $i^* \in \mathbb{N}$ such that, for any $i \geqslant i^*$, $w(\mathcal{B}_1^{(i)}) < \min\{\delta_u, \delta_m\}$. Therefore from (17) and (19) we obtain

$$u_b^{(i)} - l_b^{(i)} < 2\varepsilon \quad \forall i \geqslant i^*.$$

By choosing $\varepsilon = T/(2K_{\mathrm{pre}})$ the above inequality becomes

$$u_b^{(i)} - l_b^{(i)} < \frac{T}{K_{\mathrm{pre}}} \quad \forall i \geqslant i^*. \tag{20}$$

If the interval procedure does not halt at step 4 or 5 then necessarily, by statement (20), it halts at step 6 not exceeding iteration i^*. □

The following corollary of theorem 1 is acquired by considering the structure (4) of the penalty function $\Phi(\cdot)$ and steps 4–6 of the above procedure.

Corollary 1: *The presented interval procedure computes $\Phi(\sigma(x))$ with arbitrarily good precision.*

4. Application examples

The potentiality of the hybrid algorithm proposed in this work is tested in the following by means of several examples. In the §4.1, three test problems are considered. The first two are drawn from a set of test problems examined by Watson (1983). The third is the most demanding, being strongly multimodal for both the objective function and the constraint function. In §4.2 a solution to the optimal worst-case H_2 design of a PID controller is given for an example plant that is uncertain and non-minimum phase.

The parameters of the genetic algorithm have been chosen via a preliminary investigation made on some classical unconstrained test problems (Mühlenbein *et al.* 1991). The aim was to identify the most critical parameters from the viewpoint of global convergence and computational time efficiency. The genetic algorithm was found to be marginally affected by some parameters while its behaviour rapidly changes depending on some other parameters. In particular, the choice of $\xi = 4$, $\lambda = 80$, $P_M = 0.05$ and $\rho = 0.8\gamma$ is satisfactory for all problems; variations in these parameters lead to only a small improvement in the efficiency. Conversely, depending on the problem, other parameters are more critical. For example g_M, γ, η and C_{sca} need to be carefully chosen in the following ranges: $g_M = [50, 500]$, $\gamma = [30, 300]$, $\eta = [0.7, 0.8]$ and $C_{sca} = [1.2, 2.0]$. Multimodal functions require, in general, larger values for g_M and γ, while smaller values are needed for η and C_{sca}.

The algorithm has been coded in $C++$ and compiled for Windows 95 using the Watcom compiler. The computation times reported in the following refer to a Pentium 120 MHz personal computer.

4.1. *Test problems for the genetic–interval algorithm*

Problem 1:

$$\min_{x \in \mathcal{X}} \{f(x) := e^{x_1} + e^{x_2} + e^{x_3}\}, \tag{21}$$

subject to

$$g(x,q) := \frac{1}{1+q_1^2} - x_1 - x_2 q_1 - x_3 q_1^2 \leqslant 0 \quad \forall q \in \mathcal{Q}, \tag{22}$$

where

$$x := [x_1\, x_2\, x_3]^T \in \mathcal{X} := [-2, 2]^3,$$
$$q := q_1 \in \mathcal{Q} := [0, 1]. \qquad \square$$

Problem 2:

$$\min_{x \in \mathcal{X}} \{f(x) := -4x_1 - \tfrac{2}{3}(x_4 + x_6)\}, \tag{23}$$

subject to

$$g(x,q) := x_1 + x_2 q_1 + x_3 q_2 + x_4 q_1^2 + x_5 q_1 q_2 + x_6 q_2^2 - 3 - (q_1 - q_2)^2 (q_1 + q_2)^2 \leqslant 0 \quad \forall q \in \mathcal{Q} \tag{24}$$

where

$$x := [x_1\, x_2\, x_3\, x_4\, x_5\, x_6]^T \in \mathcal{X} := [-5, 5]^6,$$
$$q := [q_1 q_2]^T \in \mathcal{Q} := [-1, 1]^2. \qquad \square$$

Problem 3:

$$\min_{x \in \mathcal{X}} \{f(x) := 4(x_1 - 0.2)^2 + 4(x_2 - 0.2)^2 + 4(x_3 - 0.2)^2 - 0.2\cos(30\pi x_1) - 0.2\cos(30\pi x_2) - 0.2\cos(30\pi x_3) + 0.6\}, \tag{25}$$

subject to

$$g(x,q) := \sin(q_1 x_1 + q_2 x_2) - \tfrac{1}{10}(q_1 - 5)^2 - \tfrac{1}{10}(q_2 - 5)^2 - x_2 - x_3 - \sin(6 q_1 x_3) - \sin(6 q_2 x_3) - 2 \leqslant 0 \quad \forall q \in \mathcal{Q}, \tag{26}$$

where

$$x := [x_1\, x_2\, x_3]^T \in \mathcal{X} := [0, 0.4]^3$$
$$q := [q_1 q_2]^T \in \mathcal{Q} := [1, 10]^2. \qquad \square$$

In table 1 the settings of the genetic algorithm are reported. The objective functions of the first two problems are unimodal. For this reason a low number of individuals are sufficient to estimate the global minimizer. On the contrary, a larger population has been chosen for problem 3 owing to its strongly multimodal objective function. The overall performances of the hybrid algorithm have been tested by considering 50 runs. For each problem, table 2 reports some related statistics including total computational times and the number of convergences to the estimated global solution. It could be interesting to add that the computational time spent in the genetic part of the algorithm

Table 1. Settings for the genetic algorithm.

	γ	ρ	η	C_{sca}	g_M	ε_{end}	K_{pre}	T_I	T_{II}	b_c
Problem 1	30	24	0.7	1.8	300	10^{-5}	3	10	10^{-6}	20
Problem 2	30	24	0.7	1.8	200	10^{-2}	3	10	10^{-6}	20
Problem 3	200	160	0.7	1.2	80	10^{-4}	3	1	10^{-6}	20

Table 2. Statistics for the three test problems.

	Generations required, average (min–max)	Computational time, average (min–max)	Convergence to the global solution
Problem 1	285 (251–300)	1 min 31 s (1 min 16 s–2 min 2 s)	50 on 50 runs
Problem 2	189 (162–200)	8 min 34 s (5 min 10 s–13 min 29 s)	49 on 50 runs
Problem 3	67 (59–80)	4 min 49 s (4 min 07 s–5 min 54 s)	47 on 50 runs

Table 3. Solutions for the three test problems.

	Estimated minimizer x^*	Estimated global minimum $f(x^*)$	Average objective function (on 50 runs)	Standard deviation (on 50 runs)
Problem 1	$\begin{bmatrix} 1.0066 \\ -0.1271 \\ -0.3795 \end{bmatrix}$	4.3012	4.3071	7.0259×10^{-3}
Problems 2	$\begin{bmatrix} 2.9997 \\ 0.0014 \\ -0.0001 \\ -0.0034 \\ 0.0003 \\ 0.0014 \end{bmatrix}$	−11.998	−11.921	0.1903
Problems 3	$\begin{bmatrix} 0.20001 \\ 0.20001 \\ 0.26638 \end{bmatrix}$	1.7702×10^{-2}	1.8764×10^{-2}	4.2462×10^{-3}

is, in the worst case, shorter than 24 s for problem 1 and shorter than 18 s for problems 2 and 3. The obtained numerical solutions are summarized in table 3. It shows the estimated global minima and associated minimizers taken from the best solutions over 50 runs. Moreover, the average objective function and the standard deviation of all the solutions for every problem are included. It is worth stressing that all the obtained solutions are feasible with certainty.

The first problem appears to be the simplest. In all the runs the algorithm has converged to estimated global minimizers that are very close to each other and all have approximately the same value for the objective function, as can be indirectly shown by the small standard deviation. The shape of the constraint function at the solution x^* is shown in figure 4.

Problem 2 is more complex. From the genetic point of view, the complexity is given by the search box with six dimensions. From the interval point of view the situation is complicated by the infinite number of points in which the constraint function is active: in $q_1 \pm q_2 = 0$ we have $g(x^*, q) = 0$ (figure 5). For this reason, relatively longer computational times were obtained in spite of the small number of individuals used.

In problem 3 the objective function has been designed with many local minima. Only in three cases (over 50 runs) has the genetic algorithm been entrapped in one of these local minima. In figure 6 the shape of $g(x^*, q)$ is shown. It is evident that the constraint function is multimodal too.

The solutions obtained for problems 1 and 2 are the same as evaluated by Watson (1983) (only the last digits differ) where an algorithm, globally convergent to a stationary point of the problem, is proposed. This algorithm requires a starting point and does not determine a global solution unless a 'right' starting point is selected by the user. Problem 3 has also been checked with the MATLAB optimization toolbox (Grace 1994). Using the specific routine for semi-infinite optimization, different solutions have been obtained depending on the chosen starting point x_0: for example $x_{\text{in}} = [0\ 0\ 0]^{\text{T}}$ leads to the local mini-

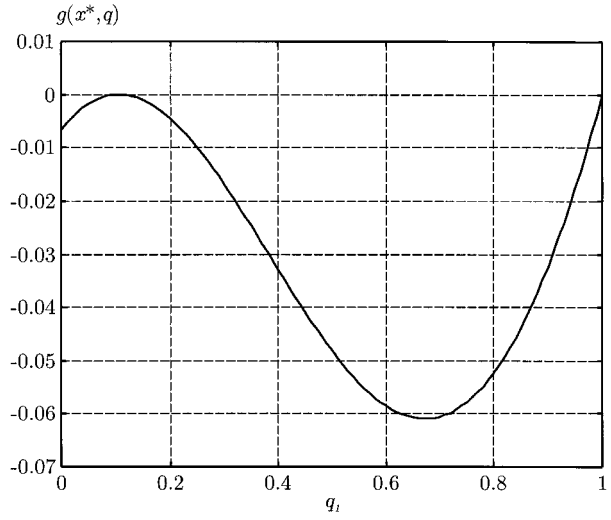

Figure 4. Shape of the constraint function $g(x^*, q)$ for problem 1.

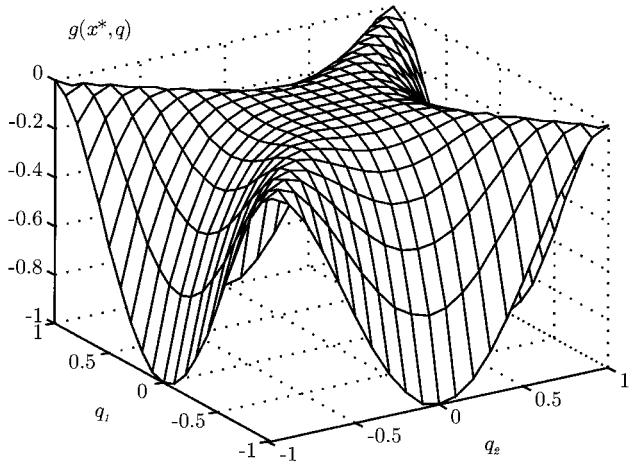

Figure 5. Shape of the constraint function $g(x^*, q)$ for problem 2.

mizer $x_{\mathrm{fi}} = [0.20000\ 0.26637\ 0.26636]^{\mathrm{T}}$ with $f(x_{\mathrm{fi}}) = 3.5396 \times 10^{-2}$ while on selecting $x_{\mathrm{in}} = [0.4\ 0.4\ 0.4]^{\mathrm{T}}$ the routine returns $x_{\mathrm{fi}} = [0.26640\ 0.33274\ 0.26642]^{\mathrm{T}}$ with $f(x_{\mathrm{fi}}) = 10.619 \times 10^{-2}$. The same solution achieved by the genetic–interval algorithm is only obtained when the starting point lies in a small neighbourhood of the genetic minimizer (e.g. choosing $x_{\mathrm{in}} = [0.2\ 0.2\ 0.25]^{\mathrm{T}}$). In fact the MATLAB routine, analogously to Watson's algorithm, can only claim convergence to local minima. Moreover there is no guarantee on the feasibility of the reached minimizer because this is simply checked by means of a grid on $\mathcal{Q}$ whose starting step must be carefully chosen by the user. On the contrary, owing to the deterministic interval procedure, the proposed hybrid algorithm guarantees the feasibility of the obtained minimizer (see § 2.2).

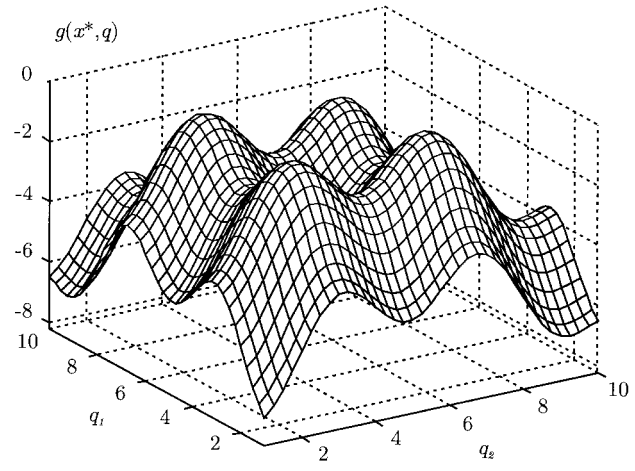

Figure 6. Shape of the constraint function $g(x^*, q)$ for problem 3.

4.2. *Minimax control of an uncertain plant with a PID controller*

Consider the unity-feedback control system of figure 7 where the uncertain plant is given by

$$P(q;s) = q_3 \frac{(1 - q_1 s) q_2^2}{s^2 + 2\delta q_2 s + q_2^2}, \tag{27}$$

with uncertain parameter vector $q := [q_1\ \ q_2\ \ q_3]^{\mathrm{T}} \in \mathcal{Q} := [0.007, 0.010] \times [4, 6] \times [0.9, 1.1]$ and $\delta = 0.5$. The PID controller transfer function is given by

$$C(k;s) = k_1 \left(1 + \frac{1}{k_2 s} + \frac{k_3 s}{1 + \tau s}\right), \tag{28}$$

where $k := [k_1\ \ k_2\ \ k_3]^{\mathrm{T}} \in \mathcal{K} := [0.5, 50] \times [0.05, 5] \times [0.05, 5]$ and $\tau = \frac{1}{50}$s. Note that, because of realizability reasons, the derivative action of the PID is filtered with the single pole $-1/\tau$.

The addressed PID design problem is the following: determine the PID parameters k_i such that, firstly closed-loop stability holds for every $q \in \mathcal{Q}$ and secondly, the worst-case objective function $\max_{q \in \mathcal{Q}} \{J(k, q)\}$ is minimized, where

$$J(k, q) := \int_0^\infty [e(k, q; t)]^2 \, \mathrm{d}t \tag{29}$$

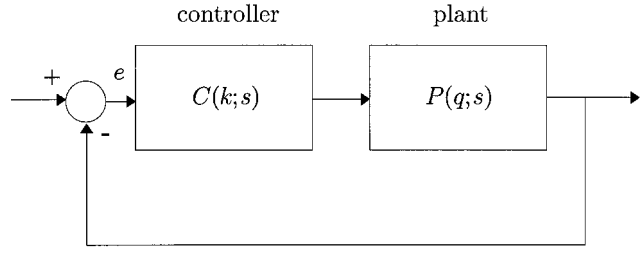

Figure 7. Control scheme.

is the integral of the squared error of the unit-step response.

These robust design specifications lead to the following minimax optimization problem:

$$\min_{k\in\mathcal{K}} \max_{q\in\mathcal{Q}} \{J(k,q)\}, \tag{30}$$

subject to

$$H_i(k,q) \geqslant \varepsilon \quad \forall q \in \mathcal{Q},\ i = 1,2,3,4,$$

where $H_i(k,q)$ denotes the Hurwitz determinants associated with the characteristic polynomial of the closed-loop system and ε is chosen to assure their strict positivity (e.g. $\varepsilon = 10^{-4}$).

Problem (30) can be easily transformed into the following equivalent infinitely constrained problem:

$$\min_{x\in\mathcal{X}} \{x_4\}, \tag{31}$$

subject to

$$g_i(x,q) \leqslant 0 \quad \forall q \in \mathcal{Q},\ i = 1,2,3,4,5,$$

where

$$\begin{aligned} x &:= [x_1\ \ x_2\ \ x_3\ \ x_4]^{\mathrm{T}} \\ &\equiv [k_1\ \ k_2\ \ k_3\ \ x_4]^{\mathrm{T}} \in \mathcal{X} := \mathcal{K}[0.001,10], \\ g_i(x,q) &= \varepsilon - H_i(k,q), i = 1,2,3,4, \\ g_5(x,q) &= J(k,q) - x_4. \end{aligned}$$

This latter constraint is introduced to take into account the worst-case objective function while performing a standard semi-infinite optimization. By denoting, in fact, the optimal solutions of (30) and (31) by k^* and x^* respectively it can be verified that $x_4^* = \max_{q\in\mathcal{Q}} \{J(k^*,q)\}$ and, obviously, $k_j^* = x_j^*$ for $j = 1,2,3$.

Closed-form expressions can be given for both $H_i(k,q)$ and $J(k,q)$. In particular, $J(k,q)$ can be symbolically evaluated using the determinantal method of Katz (1952) (see also Jury and Dewey (1965)). The partial derivatives of $H_i(k,q)$ that are needed in order to use the interval procedure of the hybrid algorithm can be easily determined because $H_i(k,q)$ is a multivariate polynomial in both the arguments k and q.

The genetic–interval algorithm has been applied to (31), executing 50 runs, with $\gamma = 30$, $\rho = 24$, $\eta = 0.7$, $C_{\mathrm{sca}} = 1.8$, $g_{\mathrm{M}} = 200$, $\varepsilon_{\mathrm{end}} = 10^{-6}$, $K_{\mathrm{pre}} = 3$, $T_{\mathrm{I}} = 100$, $T_{\mathrm{II}} = 10^{-5}$ and $b_{\mathrm{c}} = 20$. The convergence is reached within 136–200 generations with computational times in the interval 3 min 32 s–8 min 22 s. On average the solution is obtained in 172 generations with a mean computational time equal to 5 min 17 s. The hybrid algorithm has converged 44 times (out of 50) to the same approximate global solution. Also for this problem the time consumption related to the genetic part of the algorithm is negligible (it is close to 1 s). The mean value of the objective function is 4.341×10^{-2} with a standard deviation of 2.422×10^{-3}. The estimated global solution is $k_1^* = 3.62, k_2^* = 0.139$ s, $k_3^* = 0.412$ s and $x_4^* = \max_{q\in\mathcal{Q}} \{J(k^*,q)\} = 4.19 \times 10^{-2}$. The optimal PID zeros of $C(k^*;s)$ are equal to $-1.323 \pm 3.858\,\mathrm{i\,s^{-1}}$.

The behaviour of the optimal closed-loop system has been analysed for three different points of the uncertain box $\mathcal{Q}$. One of them, $q_{\mathrm{b}} = [0.0085,5,1]$, refers to the 'nominal' plant (q_{b} is the midpoint of $\mathcal{Q}$) while the others, $q_{\mathrm{a}} = [0.010,6,1.1]$ and $q_{\mathrm{c}} = [0.007,4,0.9]$, correspond to particular vertic of $\mathcal{Q}$. The zeros and poles of the closed-loop system are reported in table 4. The dominating poles depend only slightly on the variability of the plant parameters; this is not true for the fastest poles. As a consequence, the closed-loop system has two different dynamics; the slowest of them is nearly independent of the point $q \in \mathcal{Q}$ considered.

In figure 8 the optimal step responses are shown. In the worst case (plant a), a 50% overshoot has been detected. The integral of the squared error has been evaluated by means of simulations; for the nominal plant $J(k^*,q_{\mathrm{b}}) = 0.0334$ and in correspondence to the other two vertex points $J(k^*,q_{\mathrm{a}}) = 0.0418$ and $J(k^*,q_{\mathrm{c}}) = 0.0388$. Note that vertex a has the worst-cost index which is approximately equal to the value x_4^* found by the genetic–interval algorithm. The good

Table 4. Poles, zeros and gains of theoptimal closed-loop system.

System	Poles	Zeros	Gain
a	-11.11 ± 56.15i	100.00	-31.00
	-1.394 ± 3.714i	-1.323 ± 3.858i	
b	-17.84 ± 41.93i	117.7	-16.63
	-1.347 ± 3.724i	-1.323 ± 3.858i	
c	-21.82 ± 26.88i	142.9	-7.891
	-1.238 ± 3.757i	-1.323 ± 3.858i	

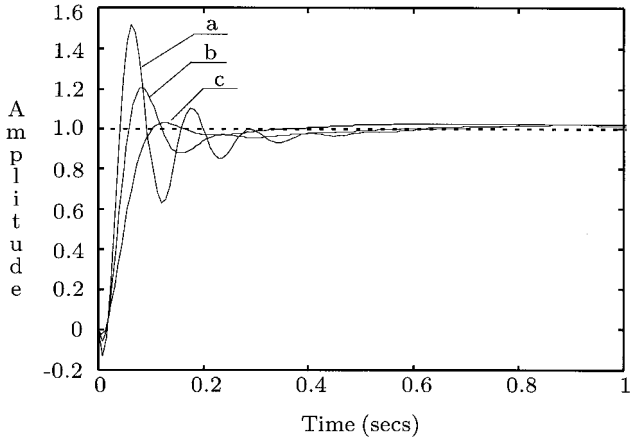

Figure 8. Optimal step responses of the closed-loop system.

behaviour of the closed-loop system in the steady state is demonstrated by the velocity constant, which is sufficiently large for the three considered plants: $K_v(k^*, q_a) = 28.65, K_v(k^*, q_b) = 26.05$ and $K_v(k^*, q_c) = 23.44$.

5. Conclusions

In this paper a hybrid genetic–interval algorithm has been proposed to obtain an estimate of the global solution to a non-convex, infinitely constrained optimization problem. Although typically this is a difficult NP-hard problem, the algorithm has proved to be robust and effective for non-trivial problems with acceptable computation times. In particular, the algorithm has been successfully used to solve test problems, including a multimodal example where a traditional method can be easily entrapped in local minima, and to design an optimal worst-case H_2 PID controller for a non-minimum plant with real parametric uncertainties. More elaborate control applications have been reported by Guarino Lo Bianco and Piazzi (1997, 1998, 1999).

To the present authors' knowledge the proposed hybrid algorithm for semi-infinite problems is the first attempt that conjugates stochastic global optimization with deterministic global optimization. For this reason it can be improved by benefiting from advancements drawn from both approaches. Future research will be dedicated to upgrading this hybrid approach.

References

ACKLEY, D. H., 1987, *A Connectionist Machine for Genetic Hillclimbing* (Boston, Massachusetts: Kluwer).

BOX, M., 1965, Complex method. *The Computer Journal*, **8**, 42–52.

DAVIS, L. (editor), 1991, *Handbook of Genetic Algorithms* (New York: Van Nostrand Reinhold).

DEJONG, K., and SPEARS, W., 1990, An analysis of the interacting roles of population size and crossover in genetic algorithms, *Proceedings of the First Workshop on Parallel Problem Solving from Nature* (Berlin: Springer-Verlag), pp. 38–47.

FIACCO, A., and KORTANEK, K. (editors), 1983, *Semi-Infinite Programming and Applications*, Lecture Notes in Economics and Mathematical Systems, Vol. 215 (Berlin: Springer-Verlag).

GOLDBERG, D., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Reading, Massachusetts: Addison-Wesley).

GRACE, A., 1994, *Optimization Toolbox User's Guide* (Reading, Massachusetts: The Mathworks, Inc.).

GUARINO LO BIANCO, D., and PIAZZI, A., 1997, Mixed H_2/H_∞ fixed-structure control via semi-infinite optimization. *Proceedings of the Seventh IFAC Symposium on Computer Aided Control Systems Design*, Gent, Belgium, 1997, pp. 329–334; 1998, A worst-case approach to SISO mixed H_2/H_∞ control. *Proceedings of the IEEE Conference on Control Applications*, Trieste, Italy, 1998 (New York: IEEE), pp. 684–688; 1999, A genetic/interval approach to optimal trajectory planning of industrial robots under torque constraints. *Proceedings of the European Control Conference*, Karlsruhe, Germany, 1999.

HANSEN, E., 1992, *Global Optimization Using Interval Analysis* (New York: Marcel Dekker).

HETTICH, R. (editor), 1979, *Semi-Infinite Programming*, Lecture Notes in Control and Information Science, Vol. 15 (Berlin: Springer-Verlag).

HETTICH, R., and KORTANEK, K., 1993, Semi-infinite programming: theory, methods, and applications. *SIAM Review*, **35,** 380–429.

HOLLAND, J., 1975, *Adaptation in Natural and Artificial Systems* (Ann Arbor, Michigan: The University of the Michigan Press).

JAULIN, L., and WALTER, E., 1996, Guaranteed tuning with application to robust control and motion planning. *Automatica*, **32,** 1217–1221.

JURY, E., and DEWEY, A., 1965, A general formulation of the total square integrals for continuous systems. *IEEE Transactions on Automatic Control*, **10,** 119–120.

KATZ, A., 1952, On the question of calculating the quadratic criterion for regulation. *Prikladnaya Matematika; Mekhanika*, **16,** 362–364 (in Russian).

LUENBERGER, D., 1989, *Linear and Nonlinear Programming*, second edition (Reading, Massachusetts: Addison-Wesley).

MALAN, S., MILANESE, M., and TARAGNA, M., 1997, Robust analysis and design of control systems using interval arithmetic. *Automatica*, **33,** 1363–1372.

MENOZZI, R., and PIAZZI, A., 1996, On the use of a genetic algorithm for millimeter-wave FET modeling. *Proceedings of the 26th European Solid State Device Research Conference*, Bologna, Italy, 1996, pp. 663–666.

MENOZZI, R., PIAZZI, A., and CONTINI, F., 1996, Small-signal modeling for microwave FET linear circuits based on a genetic algorithm. *IEEE Transactions on Circuits and Systems, Part I, Fundamental Theory and Applications*, **43,** 839–847.

MICHALEWICZ, Z., 1992, *Genetic Algorithms + Data Structures = Evolution Programs* (Berlin: Springer-Verlag).

MOORE, R., 1996, *Interval Analysis* (Englewood Cliffs, New Jersey: Prentice-Hall).

MÜHLENBEIN, H., SCHOMISCH, M., and BORN, J., 1991, The parallel genetic algorithm as function optimizer. *Parallel Computing*, **17,** 619–632.

PIAZZI, A., and MARRO, G., 1996, Robust stability using interval analysis. *International Journal of Systems Science*, **27,** 1381–1390.

PIAZZI, A., and VISIOLI, A., 1998, Global minimum-time trajectory planning of mechanical manipulators using interval analysis. *International Journal of Control*, **71,** 631–652.

POLAK, E., 1987, On the mathematical foundation of nondifferentiable optimization in engineering design. *SIAM Review*, **29,** 21–89.

POLAK, E., and MAYNE, D., 1985, Algorithm models for non-differentiable optimization. *SIAM Journal of Control and Optimization*, **23,** 477–491.

POLAK, E., MAYNE, D., and STIMLER, D., 1984, Control system design via semi-infinite optimization: a review. *Proceedings of the IEEE*, **72,** 1777–1794.

POLAK, E., and WARDI, Y., 1992, Nondifferentiable optimization algorithm for designing control systems having singular value inequalities. *Automatica*, **18,** 267–283.

RATSCHEK, H., and ROKNE, J., 1988, *New Computer Methods for Global Optimization* (Chichester, West Sussex: Ellis Horwood).

RUDOLPH, G., 1994, Convergence analysis of canonical genetic algorithm. *IEEE Transactions on Neural Networks*, **5,** 96–101.

SCHOENAUER, M., and XANTHAKIS, S., 1993, Constrained GA optimization. *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana–Champaign, Illinois, 1993, pp. 573–580.

SCHWEFEL, H.-P., 1977, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie* (Basel: Birkenäuser).

WATSON, G. A., 1983, Numerical experiments with globally convergent methods for semi-infinite programming problems. *Semi-Infinite Programming and Applications*, Lecture Notes in Economics and Mathematical Systems, Vol. 215, edited by A. V. Fiaco and K. O. Kortanek (Berlin: Springer-Verlag), pp. 193–205.