Design of mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control systems using algorithms inspired by the immune system

Frederico G. Guimarães a, Reinaldo M. Palhares a,*, Felipe Campelo b, Hajime Igarashi b

a Department of Electronics Engineering, Federal University of Minas Gerais, Av. Antônio Carlos 6627, 31270-010, Belo Horizonte, MG, Brazil
b Research Group of Informatics for System Synthesis, Graduate School of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9, Kita-ku, Sapporo 060-0814, Japan

Received 24 November 2006; received in revised form 16 February 2007; accepted 20 February 2007

Abstract

We utilize optimization algorithms inspired by the immune system for treating the mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control problem. Both precisely known systems and uncertain systems with polytopic uncertainties are investigated. For the latter, a novel methodology is proposed to compute the worst case norms within the polytope of matrices. This methodology consists in defining the worst case norm computation as an implicit optimization problem with a special structure. We exploit this structure of the problem for its solution. The paper presents both mono and multiobjective optimization algorithms developed from the clonal selection principle. The former is the real-coded clonal selection algorithm (RCSA) and the latter is the multiobjective clonal selection algorithm (MOCSA). The complete design process involves the combination of synthesis and analysis. The RCSA is used for analysis, through the worst case norm computation for a given provided controller. The MOCSA is used for synthesis, working on a population of candidate controllers, until providing an estimate of the Pareto set for the mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control problem. The numerical examples illustrate the power and the validity of the proposed approach for robust control design. Moreover, our approach for worst case norm evaluation is compared with other approaches available in literature.

© 2007 Elsevier Inc. All rights reserved.

Keywords: Robust control; Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control, Polytope-bounded uncertainty; Artificial immune systems; Multiobjective optimization

1. Introduction

The mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control problem is an important problem in control system design [31–34,25]. Also, this problem is still a very difficult multiobjective optimization problem, which may be characterized by (i)
nonlinear and multimodal search space, (ii) non-convexity, and (iii) search space with many variables with large range sizes. All these characteristics make the problem of estimating the Pareto sets very difficult to solve with conventional optimization techniques.

Alternatively, the success and broad diffusion of evolutionary optimization methods as general problem-solving techniques have facilitated the treatment of this design problem. Particularly, genetic algorithms have been broadly used in control problems in the last 10 years [2,27,23,33] as an alternative to LMI-based approaches [31,29,17]. Although genetic algorithms are the most widespread methods in control problems, they are not the only computational tools to tackle this kind of problems. In fact, many global and general-use optimization methods are available today to the designer: simulated annealing methods [24], particle swarm methods [4], just to name a few. Particularly, in [24], the authors devise a special simulated annealing method with a specific mechanism for generating new solutions and illustrate its application in mixed $H_2/H_\infty$ design. However, they do not approach the design of uncertain systems.

Recently, a new computational intelligence paradigm known as artificial immune systems have inspired the development of novel unconventional methods for solving problems in many fields of engineering and computer science [7–10,35]. In the field of optimization, some very interesting algorithms have appeared [14,11,1], but the development of multiobjective algorithms inspired by principles from theoretical immunology is very recent [6]. These works have presented very interesting and promising results, outperforming genetic algorithms and other evolutionary methods in some problems. Also recently, immune programming has appeared as an interesting alternative to genetic programming in the optimization of programs and tree structures in general [28]. Furthermore, there is no work in literature investigating these tools with the mixed $H_2/H_\infty$ control problem.

In this paper, we address the mixed $H_2/H_\infty$ control design with optimization algorithms inspired by the immune system. We present both the real-coded clonal selection algorithm (RCSA) for single objective problems and the multiobjective clonal selection algorithm (MOCSA) for multiobjective problems. The RCSA was proposed first in [1] as a real-coded optimization algorithm within the artificial immune systems paradigm. Here, we also propose a new multiobjective version of RCSA, the MOCSA, which is also one of the contributions of this paper. A very interesting characteristic in these algorithms is that they present local and global search capabilities.

Furthermore, both precisely known systems and systems with polytopic uncertainties are investigated. For the last ones in particular, worst case norms must to be considered. In order to approach this kind of problem, a novel methodology is devised to compute the worst case norms within the polytope of matrices. This methodology is based on the transformation of the problem into a generic optimization problem, which is solved by using RCSA. We remark that the way how we tackle the norm computation problem is original and is also an important contribution in this paper. The complete design process involves the employment of RCSA for finding the worst case norm within the polytope of system matrices, and the use of MOCSA for finding the Pareto set of controllers that minimize the guaranteed $H_2$ and $H_\infty$ norms. We illustrate the algorithms developed and the worst case norm computation approach through numerical simulation examples.

The paper is organized as follows: Section 2 presents the statement of the $H_2/H_\infty$ control problem regarding both precisely known systems and systems with polytopic uncertainties; Section 3 discuss the new proposed methodology for worst case norm computation; Section 4 gives an overview of RCSA and MOCSA; and finally Section 5 provides some results.

2. Statement of the $H_2/H_\infty$ control problem

The mixed $H_2/H_\infty$ control problem is an important design problem in the field of control theory. This problem, formulated as an optimization problem with two objectives is examined with the immune-based algorithms to be described further on.

Consider the linear time-invariant dynamic system described by the following state space equations:

\[
\dot{x}(t) = A x(t) + B_0 u(t) + B_w w(t)
\]

\[
z_2(t) = C_2 x(t) + D_{2u} u(t)
\]
\[\begin{align*}
    z_\infty(t) &= C_{z_\infty} x(t) + D_{z_w} u(t) \\
    y(t) &= C x(t) + D u(t) + E w(t)
\end{align*}\]

in which \(x \in \mathbb{R}^n\) is the state vector, \(u \in \mathbb{R}^m\) is the control input vector, \(w \in \mathbb{R}^p\), is the exogenous disturbance vector, \(z = [z_2, z_\infty]^T \in \mathbb{R}^q\) is the controlled output, and \(y \in \mathbb{R}^r\) is the measured output.

The mixed \(\mathcal{H}_2/\mathcal{H}_\infty\) control problem can be stated as to find a stabilizing controller \(\mathcal{K}\) such that it minimizes the \(\mathcal{H}_2\) norm of the closed-loop system while maintaining the \(\mathcal{H}_\infty\) norm below a prescribed level \(\gamma\). The controller \(\mathcal{K}\) may be described as a static system:

\[u(t) = K y(t)\]  \hspace{1cm} (5)

or a dynamic one:

\[\begin{align*}
    \dot{x}_c(t) &= A_{x_c} x_c(t) + B_{y_c} y(t) \\
    u(t) &= C_{x_c} x_c(t) + D_{y_c} y(t)
\end{align*}\]  \hspace{1cm} (6)

The case of state feedback can be obtained from (5) just by considering \(y = x\).

The parameters of the controller are the elements of the matrices that define the controller, and are given by

\[k = k(\mathcal{K}), \quad k \in \mathbb{R}^d\]  \hspace{1cm} (8)

where each variable \(k_i, i = 1, \ldots, d\), in the vector \(k\) is an element of the matrices that define the controller in (5) or (6), (7). The dimension \(d\) is given by the total number of these elements.

Consider that \(T_{z_2 w}\) represents the closed-loop system from \(w\) to \(z_2\) and \(T_{z_\infty w}\) represents the closed-loop system from \(w\) to \(z_\infty\). Typical approaches to the \(\mathcal{H}_2/\mathcal{H}_\infty\) control synthesis set the problem as

\[k^* = \arg \min_k \|T_{z_2 w}\|_2\]

subject to \(\|T_{z_\infty w}\|_\infty \leq \gamma\) \hspace{1cm} (9)

In the multiobjective point of view, we may state the problem as to minimize simultaneously the objectives \(\|T_{z_2 w}\|_2\), the \(\mathcal{H}_2\) norm, and \(\|T_{z_\infty w}\|_\infty\), the \(\mathcal{H}_\infty\) norm:

\[k^* = \arg \min_k \left[ \|T_{z_2 w}\|_2 \right] \left[ \|T_{z_\infty w}\|_\infty \right]\]  \hspace{1cm} (10)

However, now \(k^*\) is not only one controller as in the problem defined in (9), but also a set of possible controllers that characterize the Pareto set for the multioobjective problem. In fact, the solution of (9) provides one solution of (10). By varying the level \(\gamma\), it is possible to obtain various estimates of the solutions of (10). The subtle difference is that multiojective methodologies likewise evolutionary algorithms solve the multiojective problem directly, as stated in (10). Thus, one single run of the algorithm is capable of providing a representative set of estimates of the trade-off curve between the objectives, instead of solving a sequence of problems in the form posed in (9).

Nevertheless, the formulation in (10) is valid only for precisely known systems. For uncertain systems whose the uncertainty is modelled as a polytope-bounded set of matrices, we need to redefine the computation of the norms. This is one of the main contributions of this work.

Consider that the system matrix \(S\) contains all the system matrices:

\[S = \begin{bmatrix} A & B_u & B_w \\ C_{z_2} & D_{z_2 u} & 0 \\ C_{z_\infty} & D_{z_\infty u} & 0 \\ C & D & E \end{bmatrix}\]  \hspace{1cm} (11)

The uncertain parameters define a polytope-bounded set of matrices:

\[\mathcal{P}(\lambda) = \left\{ S : S = \sum_{i=1}^N \lambda_i S_i ; \lambda \in A \right\}\]  \hspace{1cm} (12)
with:
\[
A = \left\{ \lambda \in \mathbb{R}^N : \lambda_i \geq 0, \sum_{i=1}^N \lambda_i = 1 \right\}
\]

Therefore, \( \mathcal{P}(\lambda) \) represents a polytope of matrices, and every system within the polytope may be expressed as a convex combination of the matrices in the vertices of the polytope.

In a scenario under uncertainty, we aim to minimize the \( \mathcal{H}_2/\mathcal{H}_\infty \) performances at the worst case, since we do not have a nominal system but a polytopic-bounded set of systems. Thus, the problem (10) is transformed into

\[
k^* = \arg \min_k \left[ \max_{\lambda} \|T_{z_2w}\|_2 \right]
\]

that is, it is similar to a min max problem and the norms are functions of both \( k \) and \( \lambda \):

\[
\|T_{z_2w}\|_2 = T_2(k, \lambda)
\]

\[
\|T_{z_\infty w}\|_\infty = T_\infty(k, \lambda)
\]

Nevertheless, we remark that the maximization in \( \lambda \) is only possible when a given \( k \) is defined.

Observe that the evaluation of the \( \mathcal{H}_2/\mathcal{H}_\infty \) performances involves an implicit optimization problem, which is to find the convex combination \( S \in \mathcal{P}(\lambda) \) that maximize the \( \mathcal{H}_2/\mathcal{H}_\infty \) norm, i.e., that gives the worst case value.

In the following section, we develop a procedure for tackling this problem.

3. Proposed methodology

In this section, we devise a methodology for transforming the problem in (14) into the following one:

\[
k^* = \arg \min_k \left[ \frac{J_2(k)}{J_\infty(k)} \right]
\]

This transformation is attained by defining adequately the functions \( J_2 \) and \( J_\infty \). Following, we define the implicit optimization problem and the design methodology.

3.1. Worst case norm computation

For a given \( k \), the worst case estimation can be formulated as the auxiliary optimization problem:

\[
\max_{\lambda} T_\bullet(k, \lambda) = \|T_{z_\infty w}\|_\bullet
\]
\[
\text{s.t. } \sum_{i=1}^N \lambda_i = 1
\]
\[
0 \leq \lambda_i \leq 1
\]

in which \( \bullet \) represents 2 or \( \infty \).

Since only the points that reside on the plane defined by the equality constraint must be considered, it is useful to search the solution directly in this plane. In order to do this, we consider \( N - 1 \) optimization variables and obtain the last parameter through

\[
\lambda_N = 1 - \sum_{i=1}^{N-1} \lambda_i
\]

Therefore, the problem has now \( N - 1 \) optimization variables and a different constraint:

\[
\lambda_N \geq 0 \Rightarrow \sum_{i=1}^{N-1} \lambda_i \leq 1
\]
This transformation has the benefit of reducing the number of variables and eliminating the equality constraint, which is substituted by an inequality one. This is a great advantage from a computational point of view. Fig. 1 shows the existing parallel between the polytope of matrices and the $\lambda$ parameters.

If $\lambda_N < 0$ then a high penalty is attributed to the objective function and the $\mathcal{H}_\infty$ norm is not evaluated. Otherwise, the $\mathcal{H}_\infty$ norm is calculated for the system:

$$S = \sum_{i=1}^{N} \lambda_i S_i$$

The implicit optimization problem has to be solved with an adequate optimization algorithm. After a solution $\lambda^*$ is found, we have

$$J_2(k) = T_2(k, \lambda^*)$$
$$J_\infty(k) = T_\infty(k, \lambda^*)$$

i.e., $J_2(k)$ and $J_\infty(k)$ represent the worst value achieved within the polytope of matrices for a given controller $k$.

The procedure developed in this section for evaluating the worst case norms within the polytope can be summarized in the following generic steps:

**Worst Case Norms Evaluation Procedure**

**Step (1)** Given the vector of controller parameters $k$, initialize the vector $\lambda$ with $N - 1$ parameters and select an appropriate optimization algorithm.

**Step (2)** Solve the problem:

$$\lambda^* = \arg \max_\lambda T_*(k, \lambda)$$

subject to

$$\sum_{i=1}^{N-1} \lambda_i \leq 1$$
$$0 \leq \lambda_i \leq 1$$

in which

$$\lambda_N = 1 - \sum_{i=1}^{N-1} \lambda_i$$

**Step (3)** Make $J_*(k) = T_*(k, \lambda^*)$.

![Fig. 1. Relation between the polytope of matrices and the equality constraint defined by the convex combination of the vertices of the polytope. The values $\lambda_1, \lambda_2, \lambda_3$ reside in the plane defined by the equality constraint.](image-url)
3.2. Design procedure

With a method to evaluate $J_\kappa(k)$ in hand, we may proceed to the design methodology. Observe that the worst case evaluation presented before corresponds to the analysis stage. In order to get a complete design procedure we need to combine analysis and design, i.e., we need a method to evaluate the quality of a controller $k$ and a method that exploit this information to search the best solution vector $k$.

**Design Procedure**

Step (1) Select a population-based multiobjective algorithm;
Step (2) Utilize this algorithm to solve

$$k^* = \arg \min_k \begin{bmatrix} J_2(k) \\ J_\infty(k) \end{bmatrix}$$

Step (3) The outcome of the algorithm is an estimative set of the trade-off curve of non-dominated solutions that minimize the worst-case norm values.

Notice that this design procedure involves the evaluation of $J_\kappa(k)$ for the controllers in the population of the multiobjective evolutionary algorithm, which was described previously. This design procedure is general and can be associated with any evolutionary algorithm for multiobjective optimization, see for instance [15,12,3,36]. Nonetheless, in this paper we employ optimization algorithms inspired by the immune system. Hence, in the following section we provide a detailed description of both RCSA and MOCSA. The first is used in the analysis step, that is, to evaluate $J_\kappa(k)$, and the second is used for the multiobjective design.

4. Immune-based optimization algorithms

4.1. The real-coded clonal selection algorithm (RCSA)

The RCSA was initially proposed in [1] as a mono-objective optimization algorithm based on the clonal selection principle (CSP), a theory from immunology for explaining the dynamics of the immune system. A good review of the CSP can be found at [30] from a biological point of view and at [9] from a computational perspective. Here, we give a brief overview of this method. Let the following maximization problem:

$$\begin{align*}
\max & \quad f(x) \in \mathbb{R} \\
\text{s.t.} & \quad g(x) \in \mathbb{R}^{N_g} \leq 0 \quad x \in \mathcal{D}
\end{align*}$$

where $f(x)$ is the objective function, $g(x)$ are the $N_g$ inequality constraint functions, and $\mathcal{D}$ is the search space defined by

$$\mathcal{D} = \{x \in \mathbb{R}^n : l_j \leq x_j \leq u_j\}$$

also known as box constraints.

The RCSA starts with the generation of a random initial population of $\mu$ points (antibodies) over the search space $\mathcal{D}$ of the optimization problem. The constraints may be treated by incorporating them to the objective function using the penalty approach. The iterative cycle then starts with the evaluation of these antibodies over an affinity function, which is simply the objective function value at that point, penalized by the eventual violation of the constraints.

The antibodies are then ordered according to their affinity values, and the first (i.e., highest affinity) $\sigma\mu$ ones, with $\sigma \in [0,1]$, are selected for cloning. Each of the selected antibodies receive a number of identical copies (clones) proportional to their position in the affinity ranking, as given by

$$\mu_c^i = \text{round}\left(\frac{\beta\mu}{i}\right)$$
where $\mu^*$ is the number of clones given to the $i$th antibody, $\beta$ is a multiplying factor, and $\text{round}(\cdot)$ is an operator that gives the nearest integer of its argument.

The clones are then submitted to the hypermutation (maturation) process. In this process, each clone is mutated by adding a random Gaussian noise to its current value:

$$x \leftarrow x + v$$

(27)

where $v$ represents a disturbance vector with each component given by

$$v_j = 0.1(u_j - l_j)N(0, 1)$$

(28)

and $N(0, 1)$ is the realization of a random variable with Gaussian distribution with zero mean and unitary standard deviation.

By using the Gaussian mutation, the algorithm spreads the clones of a given antibody in the space surrounding the original point, thus performing a local search which helps refining the current solutions. An antibody and its maturated clones compose a subpopulation of cells in the search space.

The maturated clones are evaluated over the affinity function, and the best point from each subpopulation is allowed to survive to the next generation of the algorithm, while all the others are destroyed. Finally, the $(1 - \sigma)\mu$ worst antibodies (the ones not selected for cloning) are replaced by new, random points. This replacement operator is responsible by the global search characteristics of the algorithm. The iterative cycle is then repeated, until a given stop criterion is met (for example, a maximum number of generations is reached). The RCSA can be summarized as

**Algorithm 1 (The Real-coded Clonal Selection Algorithm).**

**Step (1)** Define $\mathcal{D}$, $\mu$, $\sigma$, $\beta$;

**Step (2)** Initialize the population of antibodies $P(t = 0) = \{p^{(1)}, \ldots, p^{(n)}\}$;

**Step (3)** $P(t) = \{\phi(p^{(1)}), \ldots, \phi(p^{(n)})\}$ ← evaluate affinity ($P(t)$);

**Step (4)** While (Stop criterion not met) do:

- **Step (4.1)** $S(t) = \{s^{(1)}, \ldots, s^{(m)}\}$ ← selection of the best antibodies($P(t), \Phi(t), \sigma$);
- **Step (4.2)** $C(t), \Phi_C(t)$ ← Cloning and maturation($S(t), \beta$);
- **Step (4.3)** $D(t)$ ← diversity generation();
- **Step (4.4)** $\Phi_D(t)$ ← evaluate affinity($D(t)$);
- **Step (4.5)** $Q(t)$ ← $C(t) \cup D(t)$;
- **Step (4.6)** $\Phi_Q(t)$ ← $\Phi_C(t) \cup \Phi_D(t)$;
- **Step (4.7)** $P(t + 1)$ ← $Q(t)$;
- **Step (4.8)** $\Phi(t + 1)$ ← $\Phi_Q(t)$;
- **Step (4.9)** $t \leftarrow t + 1$.

The combination of local and global search capabilities is a very interesting characteristic of this algorithm. The absence of recombination operators maintains the population spread over the most promising regions, while the affinity maturation phase performs a local search on these regions. Furthermore, it is possible to control the balance between local and global search through the parameters $\beta$ (intensity of the local search) and $\sigma$ (level of global search).

An example of the application of the RCSA is given by the following analytical example: consider the problem of maximizing the 2-dimensional Rastrigin function, given by

$$f(x) = 20 + \sum_{j=1}^{2} x_j^2 - 10 \cos(2\pi x_j)$$

(29)

This function is characterized by the existence of 100 optima in the interval $-5.12 \leq x_1, x_2 \leq 5.12$, with just one global optimum, located at the point $x = [0, 0]$.

The RCSA was applied to solve this problem, with $\mu = 15$, $\sigma = 0.6$, and $\beta = 0.8$. The stop criterion was chosen as the maximum number of generations, $\text{maxgen} = 10$. This instance of the algorithm was able to find
the global optima with good precision, requiring only 415 evaluations of the objective function. We refer to [1]
for a detailed sensitivity analysis of the algorithm with respect to its parameters.

4.2. The multiobjective clonal selection algorithm

In [5,6], the authors propose a multiobjective optimization algorithm based on the clonal selection algo-
rithm, however, it utilizes binary representation of the variables and it has many parameters to adjust. We
describe here a version also based on the clonal selection principle, but using real coding and having less
parameters. The MOCSA presented here can be considered as an extension of the RCSA to multiobjective
optimization, i.e., a problem of the form:

\[
\begin{align*}
\text{max} & \quad f(x) \in \mathbb{R}^{N_f} \\
\text{s.t.} & \quad g(x) \in \mathbb{R}^m \leq 0 \quad x \in \mathcal{X}
\end{align*}
\]  

(30)

where \( N_f \) is the number of objective functions.

It uses the same immune-based principles, but two main differences deserve attention: (i) the existence of an
archive population with maximum size \( \mathcal{X} \), which stores the current Pareto estimates found by the algorithm,
and (ii) a method for defining the best solutions in a multiobjective context, which will be selected for the clon-
ing step. The baseline of MOCSA is shown below.

**Algorithm 2** *(The Multiobjective Clonal Selection Algorithm).*

**Step (1)** Define \( \mathcal{X}, \mu, \sigma, \beta, \mathcal{Y} \);
**Step (2)** initialize population \( P(t=0) = \{p^{(1)}, \ldots, p^{(m)}\} \);
**Step (3)** \( \Phi(t) = \{\phi(p^{(1)}), \ldots, \phi(p^{(m)})\} \leftarrow \text{evaluate affinity}(P(t)) \);
**Step (4)** initialize archive population \( A(t=0) \) with the non-dominated solutions in \( P(t) \);
**Step (5)** While (Stop criterion not met) do:

\[ \begin{aligned}
\text{Step (5.1)} & \quad S(t) = \{s^{(1)}, \ldots, s^{(m)}\} \leftarrow \text{selection of the best antibodies (}P(t), \Phi(t), \sigma) ;
\text{Step (5.2)} & \quad C(t) \leftarrow \text{Cloning and maturation (}S(t)) ;
\text{Step (5.3)} & \quad D(t) \leftarrow \text{diversity generation(} ;
\text{Step (5.4)} & \quad Q(t) \leftarrow C(t) \cup D(t) ;
\text{Step (5.5)} & \quad Q(t) \leftarrow \text{evaluate affinity (}Q(t)) ;
\text{Step (5.6)} & \quad A(t+1) \leftarrow \text{update archive (}Q(t), A(t)) ;
\text{Step (5.7)} & \quad P(t+1) \leftarrow S(t) \cup Q(t) ;
\text{Step (5.8)} & \quad \Phi(t+1) \leftarrow \Phi_S(t) \cup \Phi_D(t) ;
\text{Step (5.9)} & \quad t \leftarrow t + 1 .
\end{aligned} \]

Basically, this algorithm relies on the cloning, maturation and replacement operators for exploring the
search space, with the difference that, instead of returning a single optimum, the algorithm will usually return
a set of non-dominated solutions, also called the Pareto front. This set is stored on the archive population
\( A(t) \).

A key concept in MOCSA is the attribution of a scalar value for each individual of the population, given
that we have a vector of objective function values. In order to do that, we employ the classification of the pop-
ulation into non-dominated fronts. The fast non-dominated sorting technique used in the NSGA-II [13] is
adopted here to classify the population into fronts. This technique is briefly described here:

**Algorithm 3** *(Fast non-dominated sorting of a population \( P(t) \) into non-dominated fronts).*

**Step (1)** \( s(P) \leftarrow \text{size of } P(t) ;
**Step (2)** for each individual \( p^{(i)} \in P(t), i = 1, \ldots, s(P) \), do:

\[ \begin{aligned}
\text{Step (2.1)} & \quad c(i) \leftarrow \text{number of individuals in } P(t) \text{ that dominate } p^{(i)} ;
\text{Step (2.2)} & \quad L(i) \leftarrow \text{set of solutions in } P(t) \text{ that } p^{(i)} \text{ dominates} ;
\end{aligned} \]

**Step (3)** \( k \leftarrow 1 ;
**Step (4)** while not all solutions were classified, do:
Algorithm 4 (Reduction by the k-neighbour rule).

Step (4.1) \( \mathcal{I} = \{ i : c(i) = 0 \} \);
Step (4.2) put all non-dominated solutions \( p^{(i)} \) into the front \( \partial \mathcal{P}_k \);
Step (4.3) \( \mathcal{I} = \{ j : p^{(i)} \in L(\mathcal{I}) \} \);
Step (4.4) \( c(\mathcal{I}) \leftarrow c(\mathcal{I}) - 1 \);
Step (4.5) \( k \leftarrow k + 1 \).

The non-dominated sorting stands for the classification of all the candidate solutions into successive non-dominated fronts in the space of objectives, as illustrated in Fig. 2 for a minimization problem. This classification will be used for determining the number of clones each point will receive. The number of clones generated from an individual belonging to the \( i \)th front is given by

\[
\mu_i' = \text{round} \left( \frac{\mu}{i} \right)
\]

Likewise the RCSA, the MOCSA also starts with the generation of an initial population of \( \mu \) antibodies. This population can be composed of random points, or a user-defined set of points. The iterative cycle starts with the evaluation of these antibodies over all objectives, penalizing eventual violations of the constraints. The points are sorted in non-dominated fronts, and after that they receive the same number of clones based on the front they belong. The equation for the number of clones is the same used for the RCSA (26), but with \( i \) standing for the front number. Only the best \( \sigma \mu \) individuals are selected for cloning using the non-dominated sorting classification. For individuals in the same front, the distance in the objective space to its \( k \)th closest neighbour, with \( k = \text{round}(\sqrt{\mu}) \), is used to break ties. The vectors in the objective space are normalized to the unitary hypercube before calculating the distances. The minimum and maximum values (used for that normalization) of each objective are determined from the current archive.

The clones are maturated through Gaussian mutation and new solutions are introduced likewise the RCSA. The whole population (original antibodies + maturated clones + new antibodies) is again classified in fronts, and the individuals from the first front are stored in the archive population.

In order to avoid repeated or very close points in the archive population, the solutions in \( A(t) \) are submitted to a suppression operator, which reduces the size of \( A(t) \) to a maximum size \( \chi \). This procedure is shown below.

**Algorithm 4 (Reduction by the k-neighbour rule).**

**Step (1)** \( s(A) \leftarrow \text{size of } A(t) \);
**Step (2) while** \( s(A) > \chi \), **do:**

**Step (2.1)** normalize the vectors in the space of objectives;
**Step (2.2)** calculate the distance matrix \( d_{ij} \) amongst the individuals in \( A(t) \);
**Step (2.3)** \( k \leftarrow \text{round}(\sqrt{s(A)}) \);
**Step (2.4) for** \( i = 1, \ldots, s(A) \), **do:**

\[
c(i) \leftarrow \text{sum of the } k \text{ smallest distances } d_{ij}, j = 1, \ldots, s(A);
\]

**Step (2.5)** eliminate the individual \( p^{(i)} \in A(t) \) with the smallest value \( c(i) \);
**Step (2.6)** \( s(A) \leftarrow \text{size of } A(t) \).

The vectors in the objective space are normalized to the unitary hypercube to eliminate the relative difference among the range values of each objective. The distances of each individual to its \( k \) closest neighbours are determined. After that, that individual with the smallest sum of these distances is eliminated, since it is in a dense region of the Pareto. The procedure is repeated until the archive returns to the maximum size defined by the user. Thus, this procedure helps to maintain a good distribution of the solutions found.

The iterative cycle continues until a stop criterion is met. When it happens, the antibodies from the memory population are returned as the Pareto-front of optimal solutions.

As an example, consider the following multiobjective optimization problem:

\[
\min F = [f_1, f_2]^T
\]
in which

\[
f_1(x) = x
\]

\[
f_2(x) = (1 + 10x_2) \left[ 1 - \left( \frac{x_1}{1 + 10x_2} \right)^2 - \left( \frac{x_1}{1 + 10x_2} \right) \sin (8\pi x_1) \right]
\]

with \(0 \leq x_1, x_2 \leq 1\). The MOCSA was set up with \(\chi = 80, \mu = 35, \beta = 0.4, \) and \(\sigma = 0.75\). The stop criterium chosen was the maximum number of generations, \(\text{maxgen} = 40\). The solutions obtained for this problem are shown in Fig. 3. The true (analytical) Pareto-front is also shown, as the dotted line. It is important to notice that the horizontal lines are not part of the front. The MOCSA was able to find a good distribution of solutions over this discontinuous, non-convex Pareto-front, which illustrates the efficiency of this technique.

This problem was also used in [6] for comparing performance of the multiobjective immune system algorithm, MISA, with other multiobjective algorithms. Table 1 shows comparative results for average performance using the same metrics as in the work mentioned above. With the exception of MOCSA, all other performance values were obtained from [6].

The **spacing** is a performance index that measures the variance of the distance between neighboring points in the Pareto front and is defined as

\[
SP = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (d_i - \bar{d})^2}
\]

where \(d_i\) is the distance between two adjacent points in the Pareto front, \(\bar{d}\) is the mean of all \(d_i\), and \(n\) is the number of points found by the algorithm. \(SP = 0\) means that all these points are equally spaced, and so is the best value possible for this metric.

The **generational distance** is a metric that measures how far from the true Pareto front are the elements found by the algorithm. It is defined as

\[
GD = \sqrt{\frac{\sum_{i=1}^{n} d_i^2}{n}}
\]

where \(n\) is the number of points found by the algorithm, and \(d_i\) is the distance between one of these points and the closest point in the true Pareto front. Again, a value of zero is the best possible, since it means that all points found are over the true Pareto front.

In this problem, MOCSA showed the best performance for the GD metric, closely followed by PAES algorithm (Pareto Archive Evolution Strategy), although all the algorithms presented very good values for GD. The good performance obtained by MOCSA in this metric is probably due to the cloning and maturation

Fig. 2. In the non-dominated sorting technique, the points are ranked by their relative dominance.
operations. These operations make the algorithm to perform local exploration of the search space, thus improving the solutions locally and increasing the chances of getting closer to the true Pareto front. Regarding spacing, NSGA-II (Non-dominated Sorting Genetic Algorithm II) and MISA scored first, closely followed by MOCSA. PAES (Pareto Archive Evolution Strategy), and specially MicroGA (Microgenetic Algorithm), were not able to provide fairly distributed solutions for this problem.

This simple example shows that MOCSA is a fairly efficient tool for multiobjective optimization. However, the comparison of multiobjective algorithms is not an easy task, and much work is devoted to this subject in the literature, including the definition of more reasonable metrics for establishing these comparisons [26,37].

Optimization algorithms based on artificial immune systems have demonstrated good results when compared with other evolutionary algorithms, see for instance [1,6], particularly because of the good balance between local and global search, and the absence of recombination operators.

4.3. Coupling the immune-based optimization algorithms with the proposed design procedure

In this section, we describe how to couple the immune-based optimization algorithms with the proposed design procedure presented in this paper.

The computation of the worst case norms can be performed using the RCSA. Following what was already said in Section 3.1, the objective function to be maximized by the RCSA is given by:

$$
\Gamma(\lambda) = \begin{cases} 
T_\ast(k, \lambda), & \text{if } \lambda_N \geq 0 \quad \text{and} \quad S = \sum_{i=1}^{N} \lambda_i S_i \text{ is stable} \\
-100\lambda_N^2, & \text{if } \lambda_N < 0 \quad \text{and} \quad S = \sum_{i=1}^{N} \lambda_i S_i \text{ is stable} \\
10 + 100|\max \text{Real}(\sigma(S))|, & \text{if } S = \sum_{i=1}^{N} \lambda_i S_i \text{ is unstable}
\end{cases}
$$

(36)

where $\sigma(S)$ is the set of eigenvalues of the closed loop system.

---

Please cite this article in press as: F.G. Guimarães et al., Design of mixed $H_2/H_\infty$ control systems using ..., Informat. Sci. (2007), doi:10.1016/j.ins.2007.02.046
If $\lambda_i < 0$ it is not possible to evaluate the norm, so a penalty value is adopted in order to force the population to migrate to the feasible region:

$$\sum_{i=1}^{N-1} \lambda_i \leq 1$$

(37)

Moreover, if the system is unstable, it is not possible to evaluate the norm, thus a value is also adopted to force the population to converge to the region of non-stability if it exists. Therefore, if there is a combination $\lambda$ within the polytope that provides an unstable system, this vector is returned, and the value of $\Gamma(\lambda)$ is elevated.

Additionally, the low and upper limits of each $\lambda_i$ is set to $l_i = 0$ and $u_i = 1$, which define the search space $\mathcal{D}$. In the objective function $\Gamma(\lambda)$, the controller $k$ is fixed, hence this is an analysis procedure.

Finally, the $\lambda^* = \arg \max \Gamma(\lambda)$ provides the value of the worst case norm:

$$J_*(k) = \Gamma(\lambda^*) = T_*(k, \lambda^*)$$

(38)

if the system is stable.

To find the set of controllers that provide the trade-off curve of the mixed $\mathcal{H}_2/\mathcal{H}_\infty$ design, we apply the MOCSA for solving:

$$k^* = \arg \min_k \begin{bmatrix} J_2(k) \\ J_\infty(k) \end{bmatrix}$$

(39)

in which $J_{ik}(k)$ is given by (36). Notice that if there is a combination within the polytope that provide an unstable system, a high value is returned by $\Gamma(\lambda^*)$ and therefore it is eliminated by MOCSA since it is minimizing $J_*(k)$.

5. Numerical illustrative examples

The numerical examples in this section demonstrate the usefulness of the immune-based optimization tools presented and also the proposed design procedure for uncertain systems.

5.1. Comparison with respect to LMI algorithms

The first design example from [33] is investigated with MOCSA. It is a simple mixed $\mathcal{H}_2/\mathcal{H}_\infty$ design without uncertainty. The objective here is to demonstrate its solution with MOCSA. The system equations are

$$\dot{x} = \begin{bmatrix} -0.3868 & 0.0751 \\ 0 & -0.0352 \end{bmatrix} x + \begin{bmatrix} -0.6965 \\ 1.6961 \end{bmatrix} u + \begin{bmatrix} 0.0591 & 0 \\ 0 & 1.7971 \end{bmatrix} w$$

(40)

$$z = \begin{bmatrix} 0.0346 & 0.0535 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0.5297 \end{bmatrix} u$$

(41)

in which $z = z_2 = z_\infty$.

The system is controlled with a state-feedback controller:

$$u = [k_1 \ k_2] x$$

(42)

The standard LMI formulation transforms the general problem into a linear and convex optimization problem, which can be solved using common LMI solvers that are based on interior point methods [31].

This problem is solved using the LMI formulation and the MOCSA. Fig. 4 illustrates the Pareto estimates obtained by each methodology. MOCSA is able to find a set of estimates of the Pareto frontier that are more equally distributed and with a better extension over the conflicting objectives. Fig. 5 shows the solution in the parameters space $(k_1 \times k_2)$.

MOCSA was set up with the following parameters: $\chi = 80$, $\mu = 40$, $\beta = 0.5$, $\sigma = 0.7$, and maxgen = 8, giving a total of 656 function evaluations.
5.2. Norm computation for uncertain systems

This second example illustrates the norm computation for uncertain systems by using the RCSA algorithm for solving the associated optimization problem.

A satellite system consisting of two rigid bodies (main module and sensor module) connected by an elastic link that is modelled as a spring with torque constant $\tau$ and viscous damping $f$ is considered [16]. The uncertain parameters have the following uncertainty ranges:

$$0.09 \leq \tau \leq 0.40$$
$$0.0038 \leq f \leq 0.0400$$

which define the vertices of the polytope of system matrices.

The satellite system equations are

$$
\begin{bmatrix}
\dot{\theta}_1 \\
\dot{\theta}_2 \\
\dot{\theta}_1 \\
\dot{\theta}_2
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
-\tau & \tau & -f & f \\
\tau & -\tau & f & -f
\end{bmatrix}
\begin{bmatrix}
\theta_1 \\
\theta_2 \\
\dot{\theta}_1 \\
\dot{\theta}_2
\end{bmatrix} +
\begin{bmatrix}
0 \\
0 \\
1 \\
1
\end{bmatrix} u +
\begin{bmatrix}
0 \\
0
\end{bmatrix} w
$$

Fig. 4. Estimates of the Pareto frontier using the LMI formulation and MOCSA.

Fig. 5. Parameters of the controllers corresponding to the Pareto estimates obtained by MOCSA.
Consider the following state-feedback controller, as in [18]:

\[ K : K = \begin{bmatrix} C_0 \\ 4 \\ 5238 \\ C_0 \\ 2 \\ 0526 \\ C_0 \\ 3 \\ 6143 \\ C_0 \\ 16 \\ 7482 \end{bmatrix} \]

For this controller, and considering the whole polytope, [18] presents the following values for the \( H_2 \) and \( H_\infty \) norms computed via a different approach described in [22, 21]:

\[ T_2(k, \lambda^*) = 1.7382 \]
\[ T_\infty(k, \lambda^*) = 0.1989 \]

Using the proposed approach in Section 3 and the RCSA described in Section 4, we get

\[ T_2(k, \lambda^*) = 1.7373 \]
\[ T_\infty(k, \lambda^*) = 0.1990 \]

which shows the validity of the proposed methodology for computing the worst case norms for the whole polytope.

### 5.3. Mixed \( H_2/H_\infty \) design for uncertain systems

Now we consider the mixed \( H_2/H_\infty \) design for the satellite system presented before, using a state-feedback controller.

The worst case norms are evaluated using the RCSA algorithm and the procedure in Section 3. The multiobjective design is solved by using the MOCSA algorithm. The solutions obtained by an LMI-based approach (msfsyn function of Matlab) are illustrated in Fig. 6.

According to the procedure suggested in [33], these LMI solutions are used to initialize the MOCSA algorithm. This procedure accelerates the searching process of the stochastic method. The final solutions obtained by the MOCSA are also shown in Fig. 6. MOCSA was set up with the following parameters: \( \chi = 40, \mu = 50, \beta = 0.6, \sigma = 0.7, \) and maxgen = 20, giving a total of 2370 function calls.

As an example, we show one of the controllers found by MOCSA:

\[ K : K = \begin{bmatrix} C_0 \\ 4 \\ 5238 \\ C_0 \\ 2 \\ 0526 \\ C_0 \\ 3 \\ 6143 \\ C_0 \\ 16 \\ 7482 \end{bmatrix} \]

for which the worst case norms are

\[ T_2(k, \lambda^*) = 1.6861 \]
\[ T_\infty(k, \lambda^*) = 0.1841 \]

### 5.4. Design of dynamic controllers

Finally, we consider a dynamic controller for the same uncertain satellite system considered before. The controller \( \mathcal{H} \) is given by a dynamic system:
\[ \dot{x}(t) = A_x x(t) + B_x y(t) \]
\[ u(t) = C_x x(t) + D_x y(t) \]

By considering
\[ s(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \]
we obtain the closed-loop system:
\[ \dot{s}(t) = \tilde{A}s(t) + \tilde{B}w(t) \]
\[ z_2(t) = \tilde{C}_2 s(t) + \tilde{D}_2 w(t) \]
\[ z_\infty(t) = \tilde{C}_\infty s(t) + \tilde{D}_\infty w(t) \]

where
\[ \tilde{A} = \begin{bmatrix} A + B_u D_c C_y & B_u C_c \\ B_c C_y & A_c \end{bmatrix} \]
\[ \tilde{B} = \begin{bmatrix} B_w & B_u D_c D_y w \\ B_c D_y w \end{bmatrix} \]
\[ \tilde{C}_2 = \begin{bmatrix} C_{2z} + D_{2z} D_c C_y & D_{2z} C_c \end{bmatrix} \]
\[ \tilde{D}_2 = D_{2z} C_c \]
\[ \tilde{C}_\infty = \begin{bmatrix} C_{2z} + D_{2z} D_c C_y & D_{2z} C_c \end{bmatrix} \]
\[ \tilde{D}_\infty = D_{2z} C_c + D_{2z} D_c D_y \]

The parameters of the controller are the elements of the matrices that define the controller, and are given by
\[ k = k(\mathcal{X}), \quad k \in \mathbb{R}^d \]

We defined the search space for each variable as
\[ -300 \leq k_i \leq +300, \quad i = 1, \ldots, d \]

Depending on the order of the controller, we have a different number of variables in the optimization problem. Respectively, from a first-order controller to a fourth-order one (full-order controller), we get \( d = 6, 12, 20, 30 \).
Fig. 7 shows the obtained Pareto curves for each order. As we can see, by increasing the order of the controller, we achieve improved results. On the other hand, one can select a reduced-order controller, which is simpler to implement, at the price of augmented norm values. The worst case norms were computed as before.

In order to illustrate some results, we pick one controller from each Pareto front in Fig. 7. Following, we provide these controllers and their respective worst case norms, so one can locate them in Fig. 7.

For the first-order controller, we get

\[
\mathcal{K}(1): \begin{cases}
\dot{x}_c(t) &= [-4.05919]x_c(t) + [-13.70579 -23.01981]y(t) \\
u(t) &= [-7.07024]x_c(t) + [-34.19871 -34.80152]y(t)
\end{cases}
\]

which gives

\[
\begin{align*}
T_2(k, \lambda^*) &= 2.6470 \\
T_\infty(k, \lambda^*) &= 0.3689
\end{align*}
\]

For the second-order controller, we get

\[
\mathcal{K}(2): \begin{cases}
\dot{x}_c(t) &= [-49.50529 -40.40953]x_c(t) + [-28.17633 -0.28944]y(t) \\
u(t) &= [+1.49760 -6.21553]x_c(t) + [-26.61388 -41.31308]y(t)
\end{cases}
\]

which gives

\[
\begin{align*}
T_2(k, \lambda^*) &= 2.5179 \\
T_\infty(k, \lambda^*) &= 0.2780
\end{align*}
\]

For the third-order controller, we get

\[
\mathcal{K}(3): \begin{cases}
\dot{x}_c(t) &= [-19.03880 -33.82286 -26.09492]x_c(t) + [+12.60491 -47.18574 -18.97756]x_c(t) \\
&\quad + [-18.46374 -14.34930 -19.46960]x_c(t) \\
u(t) &= [-23.53001 -13.90361 -24.75864]x_c(t) + [45.06669 -41.46958]y(t)
\end{cases}
\]
which gives
\[ T_2(k, \lambda^*) = 2.1468 \]
\[ T_\infty(k, \lambda^*) = 0.2364 \] (77) (78)

Finally, for the fourth-order controller, we get
\[
\begin{aligned}
\dot{x}_c(t) &= \begin{bmatrix}
-62.88546 & -47.84083 & -45.56828 & -16.70574 \\
-14.77389 & -29.86640 & -4.96582 & -41.90584 \\
-41.69525 & -41.12704 & -48.67824 & -45.52279 \\
-26.97281 & +10.13982 & -61.01247 & -56.59403
\end{bmatrix} x_c(t) \\
+ \begin{bmatrix}
-63.31488 & -70.31994 \\
-29.09432 & +11.26641 \\
-73.59100 & -71.41508 \\
-67.98692 & -36.83440
\end{bmatrix} y(t) \\
+ \begin{bmatrix}
+7.53084 & -67.42278 & -4.13982 & -78.17500 \\
-60.41974 & -44.22328
\end{bmatrix} y(t)
\end{aligned}
\] (79)

which gives
\[ T_2(k, \lambda^*) = 1.7846 \]
\[ T_\infty(k, \lambda^*) = 0.2225 \] (80) (81)

We remark that there is no LMI formulation for solving this type of \( \mathcal{H}_2/\mathcal{H}_\infty \) design problem. Therefore, the use of population-based optimization methodologies like the one proposed in this paper is a very helpful alternative.

6. Conclusions

The characteristics of the immune system have inspired the development of very interesting computational tools. In particular, it is possible to devise unconventional optimization algorithms for generic problems, characterized by difficulties as non-convexity and multimodality. This paper presented two instances of immune-based algorithms for problems with both single and multiple objectives that present very promising results.

In addition, we propose a novel methodology for the computation of the \( \mathcal{H}_2/\mathcal{H}_\infty \) norms for uncertain systems whose uncertainty is modelled by a polytope of system matrices. The computation of the norms is established as an implicit optimization problem, which can be solved with any adequate optimization method. In the problems investigated in this paper, we employed the RCSA for solving this auxiliary optimization problem. The results achieved, for state-feedback configuration, are very similar to those presented in another recent work, which utilizes a different methodology. Although the discrete-time case has not been considered it is an easy extension.

The norm computation procedure and the MOCSA are put together in the mixed \( \mathcal{H}_2/\mathcal{H}_\infty \) control design, showing the power and adequacy of the proposed methodology. Future research includes the development of the proposed method by considering the \( \mathcal{H}_2/\mathcal{H}_\infty \) filtering problem [19], and application to robust stability tests as in [20].

Acknowledgement

This work has been supported in part by the Brazilian agencies CNPq, CAPES and FAPEMIG.

References


Please cite this article in press as: F.G. Guimaraes et al., Design of mixed $H_2/H_{\infty}$ control systems using ..., Informat. Sci. (2007), doi:10.1016/j.ins.2007.02.046