Proceedings of the 2006 IEEE
Conference on Computer Aided Control Systems Design
Munich, Germany, October 4-6, 2006

Efficient Design of Low-Order H_∞ Optimal Controllers Using Evolutionary Algorithms and a Bisection Approach

Andrey Popov and Herbert Werner

Hamburg University of Technology
Institute of Control Systems
21073 Hamburg, Germany
andrey.popov@gmx.net, h.werner@tu-harburg.de

***Abstract*— This paper considers a hybrid evolutionary-algebraic approach to the non-convex problem of designing low-order H_∞ optimal controllers. It is shown that using the closed-loop H_∞ norm as fitness measure in a population-based, evolutionary search does not require the computation of the H_∞ norm for each controller of the population. Instead, the fact that evolutionary algorithms assign fitness measures to individuals based on a ranking is exploited and a bisection approach is proposed that allows to trade accuracy that is not needed against computational efficiency without loosing information. Three design examples are used to illustrate the improvement in computational speed achieved with the proposed method.**

I. INTRODUCTION

Minimizing the closed-loop H_∞ norm is a powerful method for loop shaping or for the design of robust controllers. However, a drawback of this approach that prevents its widespread use in industrial applications is the fact that the controller order is equal to the order of the plant model, including shaping filters, which is often considered too high for implementation.

One way to use a H_∞ design to obtain low-order controllers is to employ model order reduction - either by first reducing the plant order and then designing a controller for it or by designing a full order controller first and reducing its order afterwards. Unfortunately both of these approaches suffer from the same problem - they cannot guarantee that the performance of the full-order controller - or even closed-loop stability - is also achieved by the low-order controller. Very often such a design approach turns into a tedious trial and error procedure where the outcome is unclear.

The underlying problem - designing a low-order controller for a high-order plant - is non-convex and difficult to solve, see e.g. [1]. Evolutionary algorithms are powerful tools for attacking such problems. A hybrid evolutionary-algebraic design appraoch that combines evolutionary search techniques with modern control tools such as H_2 and H_∞ optimization to solve this problem was proposed in [2]. The idea of this approach is to conduct a population-based, evolutionary search for the best controller, and to determine the fitness of individual controllers by solving computationally cheap analysis problems. In many cases, the relevant measure for determining the fitness is the closed-loop H_∞ norm. Unfortunately, computing the H_∞ norm is itself an iterative procedure and computationally expensive when applied to all individuals of a population in an evolutionary search. On the other hand, evolutionary algorithms typically assign fitness measures to individuals based on ranking, which suggests that precise knowledge of the H_∞ norm of individual controllers is not required. Exploiting this observation and carefully trading accuracy that is not needed against speed of computation, the efficiency of hybrid evolutionary-algebraic algorithms can be considerably increased. In this paper we propose a novel approach to the evolutionary search for low-order H_∞ optimal controllers. This approach is based on a well-known bisection approach [3] and the fact that evolutionary algorithms work with a population of controllers.

The paper is structured as follows. In Section II the design problem is formulated. A brief review of evolutionary algorithms and ranking selection is given in Section III. The novel population bisection approach is presented in section IV. Three design examples are presented in Section V to illustrate the achieved improvement in efficiency. Conclusions are drawn in Section VI.

II. PROBLEM FORMULATION

Consider a plant with state-space representation

$$\begin{array}{rcccccc} \dot{x} & = & Ax & + & B_w w & + & Bu \\ z & = & C_z x & + & D_{zw} w & + & D_{zu} u \\ y & = & Cx & + & D_{yw} w & & \end{array} \tag{1}$$

where u is the control input; y is the measured output; w and z are the input and output, respectively, for evaluating the H_∞ norm. The order of the plant is n.

The problem considered in this paper is to design a controller $K(s)$ of order $k < n$

$$K(s) = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right] \tag{2}$$

that minimizes the H_∞ norm of the closed loop system $\|T(s)\|_\infty$, shown in Fig. 1.

$$T(s) = \text{lft}(P(s), K(s)) = \left[\begin{array}{c|c} A_{cl} & B_{cl} \\ \hline C_{cl} & D_{cl} \end{array}\right] \tag{3}$$

As already mentioned, this problem is non-convex and cannot be solved using the standard design techniques. In the

following sections it will be shown how one can use evolutionary algorithms for an efficient search for the controller parameters.

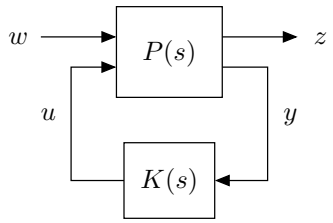

Fig. 1. Closed-loop system

III. Evolutionary Algorithms

Evolutionary Algorithms (EA) are direct, stochastic, parallel search methods for global optimization, see e.g. [4], [5]. They are well suited for solving non-convex, multi-variable and multi-objective optimization tasks. EA are working with a population of individuals, where each individual represents an admissible set of values of optimization parameters. The following notation will be used: $P(c_1, \ldots, c_\mu)$ is a population; c^i is the i^{th} individual in the population; $f(c^i)$ is the cost function value of c^i and $F(c^i)$ is the associated fitness value. Without loss of generality we consider only minimization problems. The general structure of an EA is

1) Generate a random initial population of individuals.
2) Evaluate the cost function $f(c^i)$ for each individual in the population.
3) If a termination condition is met STOP.
4) Depending on the cost function values assign fitness values $F(c^i)$ to each individual. Select the "parent" individuals that are to transmit their information into the next generation.
5) Apply evolutionary operators (crossover, mutation and elitism);
6) Go back to 2;

In the case of minimizing the closed loop H_∞ norm EA will start by randomly generating a population of low-order controllers. Each controller will then be tested on the given plant and $\|T(s)\|_\infty$ will be calculated. Next selection and evolutionary operators will be applied, generating the next generation of controllers.

A. Cost Function

Since it is not guaranteed that randomly generated controllers will stabilize the closed loop, a penalty term is included in the cost function in order to eliminate destabilizing controllers.

$$f(c^i) = \begin{cases} \beta + \max\left(\text{real}\left(\text{eig}(A^i_{cl})\right)\right) & \text{if unstable} \\ \|T^i(s)\|_\infty & \text{if stable} \end{cases} \quad (4)$$

where β is a penalty factor (e.g. $\beta = 10^{10}$).

B. Optimization Variables

Since we are interested in multivariable controllers, it is convenient to use a state-space representation of the controller. Here we denote the number of controlled inputs u by r and the number of measured outputs y by m. To keep the number of decision variables low, the following canonical state space model is used.

$$A_K = \begin{bmatrix} 0 & \ldots & 0 & a_1 \\ 1 & \ldots & 0 & a_2 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & 1 & a_k \end{bmatrix}; B_K = \begin{bmatrix} 1 & b_{12} & \ldots & b_{1m} \\ 0 & b_{22} & \ldots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & b_{k2} & \ldots & a_{km} \end{bmatrix} \quad (5)$$

C_K and D_K are $r \times k$ and $r \times m$ full variable blocks. Then the number of decision variables is $N_{dec.var} = km + rk + rm$ for a bi-proper controller, or $N_{opt.var} = km + rk$ for a strictly proper controller.

C. Selection

In EA search, the probability of an individual to be selected is determined by its fitness. Ranking selection is generally recognized (see e.g. [4]) to reduce the likelihood, due to a super individual in the population, of convergence to local minimum. In this technique the best individual in the population is assigned rank 1, the second best rank 2, etc and the fitness of each individual computed as $F(c^i) = 1/\sqrt{rank(c^i)}$ ([6]). The actual selection is performed using stochastic-uniform sampling [5], [6].

Furthermore ranking also offers the possibility of reducing computational effort, because not the precise value of the performance measure $f(c^i)$ is required, but rather one only needs to assign the correct rank to each individual (e.g. for H_∞ controller design, one only needs to know which controller has the smallest H_∞ norm, which one has the second smallest one etc).

Fig. 2 shows the fitness values vs. rank, and it can be seen that there is no significant difference between the fitness of the weakest individuals in the population. This observation can be used to further reduce computational effort: after correctly ranking the good controllers in the population, it is not necessary to determine the exact rank of the poor controllers (with large H_∞ norm) - several of them may be lumped into one group and assigned the same rank.

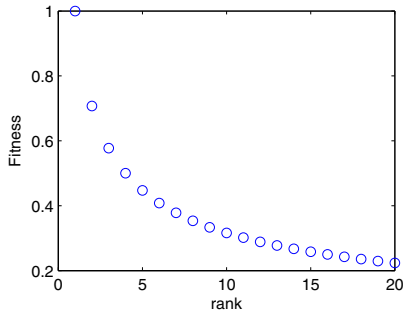

Fig. 2. Fitness as function of rank

IV. H_∞ NORM RANKING USING POPULATION BISECTION

A. Standard Algorithm

A well-known bisection technique for computing the H_∞ norm was proposed in [3]. For convenience and the sake of comparison with the population bisection technique proposed in this section, the method is briefly summarized:

1) Calculate lower and upper bound for the norm

$$\gamma_{low} = \max\left\{\sigma_{\max}(D_{cl}), \max_i[\sigma_{H_i}]\right\}$$
$$\gamma_{up} = \sigma_{\max}(D_{cl}) + 2\sum_{i=1}^{n}\sigma_{H_i} \quad (6)$$

where σ_{H_i} are the Hankel singular values of the closed loop system.

2) If $(\gamma_{up} - \gamma_{low}) \leq 2\varepsilon\gamma_{low}$ STOP.
3) $\gamma = (\gamma_{low} + \gamma_{up})/2$;
4) Form the Hamiltonian matrix M_γ (see [3]) and compute its eigenvalues;
5) If M_γ has no imaginary eigenvalues $\gamma_{up} = \gamma$, else $\gamma_{low} = \gamma$. Go to step 2;

where ε is the machine or desired precision.

B. Population Bisection

Combining the properties of the ranking selection discussed above with the bisection method for computing the H_∞ norm leads to the following algorithm.

0 **Stability check**
Do for each controller in the population:
a) Check the stability of the closed loop system with controller c^i. If A^i_{cl} has poles in the right half plane, assign a fitness value according to (4).
b) If the controller is stabilizing the closed loop compute the lower (c^i_{low}) and upper (c^i_{up}) bounds according to (6).

If there is only one stabilizing controller assign $rank(c^i) = 1$. STOP.

1 **Initialization**
Find the minimum lower (low_{all}) and maximum upper (upp_{all}) bounds of all stabilizing controllers.
Create subpopulation 1 (SP^1), containing all stabilizing controllers. Set the lower and upper bounds for the H_∞ norm of the subpopulation $SP^1_{low} = low_{all}$, $SP^1_{upp} = upp_{all}$. The rank for the best individual will be 1.

2 **Repeat** While there is still a subpopulation: Take the subpopulation (SP^k) with the minimum rank (at the first iteration that is SP^1). If SP^k is empty remove it. Else:
1) If (a) SP^k contains only 1 controller or (b) the lower and upper boundaries of SP^k are close enough $(SP^k_{low} + SP^k_{upp}) < 2\epsilon.SP^k_{low}$, assign $rank(c^i) = k$ to all controllers in the subpopulation and remove SP^k. Else
2) Choose the bisection point $\gamma = (SP^k_{low} + SP^k_{upp})/2$.
3) Check the H_∞ norm with each controller in SP^k against γ. If $c^i_{low} \geq \gamma$ or $c^i_{upp} \leq \gamma$ no further computations are required. Else construct the Hamiltonian matrix for γ and check its eigenvalues.
4) Create subpopulation SP^{k+N}, where N is the number of controllers in SP^k with H_∞ norm $< \gamma$. Assign to it all controllers from SP^k which provide $\|T^i(s)\|_\infty \geq \gamma$. Those with smaller norm than γ remain in SP^k. Set the new boundaries for SP^k and SP^{k+N}

$$SP^{k+N}_{low} = \gamma;$$
$$SP^{k+N}_{upp} = SP^k_{upp};$$
$$SP^k_{upp} = \gamma;$$

Basically this algorithm separates the population of controllers into sub-populations. The separation process stops when either there is only one controller in the subpopulation left (i.e. its rank is known) or when the expected fitness measures of the controllers in the sub-population are sufficiently close so that there is no need to distinguish between them.

C. Further Improvements

There are several features of the problem considered here that can be used to further reduce the computational cost. Each of them will be illustrated in the next section.

- Information from previous generation to determine the position of the first population bisection point could be used. Because EA uses elitism (the best controller/s in the population is/are preserved), one can use the upper bound of the H_∞ norm of the best controller of the previous population as first bisection point.
- Use computationally less expensive bounds for the H_∞ norm. As will be illustrated with examples in the next section, the bounds (6) are computationally very expensive. Instead, one can use $\gamma_{low} = \sigma_{\max}(D_{cl})$, which is the lowest allowed value for γ (see [3]). However since for the upper bound there is no similar expression, either a heuristic approach ($\gamma_{up} = \max(10\,\gamma_{low}, \Gamma)$, where Γ is a user defined value) or an adaptable upper boundary could be used.
- Reduction of conservatism of the lower bound. A limit for the lower bound of the population can be introduced by computing the full-order H_∞ controller prior to the evolutionary search and using the norm γ_{best} achieved with this controller, and setting $\gamma_{low} = \max(\sigma_{\max}(D_{cl}), \gamma_{best})$.
- Utilize the properties of the fitness function. As discussed in Section III-C, the variation of fitness values for the weakest controllers in the population is much smaller than that of those at the top, and there is no significant loss of information if several individuals at the "bottom end" of the population are assigned the same rank. This idea is implemented by adding an additional check before each subpopulation bisection. If the difference between the fitness values of the strongest

and weakest controller in the sub-population is smaller than a "skip factor"

$$\frac{1}{\sqrt{SP^k_{rank}}} - \frac{1}{\sqrt{SP^k_{rank} + SP^k_{Nctrl}}} < skip\_factor$$

then the same rank is assigned to all controllers in the subpopulation.

V. DESIGN EXAMPLES

In this section the algorithms presented in the previous section will be tested on three design examples. The EA used is the Genetic Algorithm Toolbox provided with MATLAB [6]. The following settings are used below, if not specified otherwise: initial range for each parameter between $[-1; 1]$; crossover rate = 0.8; mutation shrink = 1; mutation scale = 1; crossover type - intermediate; number of elite children = 2. All reported computation times are on Pentium IV, 3GHz processor. To make a fair evaluation of the advantage of the population bisection over the standard bisection technique the Hankel singular values are computed and the lower and upper boundaries for the norm are taken as in (6), except if stated otherwise.

A. Spinning Satellite

The first example is a spinning satellite problem, taken from [7] MATLAB (`ssic`). The physical plant is second order and has 2 inputs and 2 outputs. Shaping filters are added to the plant, which lead to an 8^{th} order generalized plant.

It is straightforward to design a full-order controller using standard tools like `hinfmix` in MATLAB. The resulting controller achieves a H_∞ norm of 70.5351, but has the same order as the generalized plant, i.e. 8^{th}.

To design a 2^{nd} order controller for this problem, we now apply the proposed evolutionary technique. The controller is represented in canonical form, as already discussed in section III-B and has 12 parameters to be optimized. For each optimization run 100 generations and a population size of 20 were used.

Starting with random populations of controllers, search runs with different values of the skip factor were performed (limit for the lower boundary and modified 1^{st} bisection point are implemented as discussed in Section IV-C). Ten optimization runs were done for each value of the skip factor. The average results from each of those set of runs are presented in Table I. The number in the first column is the skip factor used. The numbers in the "Bisections" column represent the percentage of calculations, that the population bisection does, compared to the standard bisection method. This value is obtained by bisecting the same population with both methods and than taking the ratio (i.e. 0% means that no bisections were done at all). The last two columns give the mean and the standard deviation (STD) of the runs.

The same results are graphically displayed in Figure 3. As can be seen, increasing the skip factor up to 0.5 improves the efficiency, in terms of performed bisections (reduced from 80% to 20%), while still having a very good performance. Reduction to 20% means, that the number of Hamiltonian constructions and eigenvalue problems solved by the population bisection method is five times fewer than that of the standard bisection method. A further increase of the skip factor beyond 0.5 will reduce the number of bisections, but only at the expense of a very large H_∞ norm.

TABLE I
RESULTS FOR THE SATELLITE PROBLEM

Skip factor	Bisections [% of stnd.]	H_∞ norm mean	H_∞ norm STD
0	79.27	70.6086	0.0460
0.1	67.62	70.5953	0.0574
0.2	46.06	70.6153	0.0526
0.3	32.86	70.5974	0.0541
0.4	28.57	70.6352	0.0651
0.5	22.55	70.5972	0.0360
0.6	17.25	71.0078	1.1755
0.7	10.09	72.3539	3.5015
0.8	0.00	216.6496	122.3274
0.9	0.00	202.9761	155.3374
1.0	0.00	634.4	1115.6

The improvement in terms of computation time depends on the realization of the algorithm and the programming language used (MATLAB, C or FORTRAN). Using MATLAB in interpreter mode and the `normhinf` command a run takes 14.39 sec. Of this time 5.94 sec. are spent for computing the bounds using Hankel singular values and 7.77 sec. performing bisection checks. Thus the bisection process constitutes 54% of the whole computation, while the boundary computation time takes 41%. In order to make a fair comparison we should mentioned that the H_∞ norm computation with `norminf` takes 9.61 sec.

The bisection algorithm, implemented in C language and compiled with `mex` to DLL executable for MATLAB, performs an average run for 9.32 sec (skip factor = 0). Of this time, 4.69 sec. are spent on computing the boundaries (i.e. Hankel SVD) and 4.40 sec. (or 47%) on performing bisection checks. Exchanging the boundaries (6) with the heuristic ones proposed in Section IV-C and setting $\Gamma = 0$ leads to average computation time of 4.37 sec., where 0.05 sec. are spent computing the bounds and 4.13 sec. performing bisections. At the same time the average norm is 70.6172 with STD 0.0594.

This time can be further reduced by using a non-zero skip factor. For example with a skip factor 0.4 the computation time is 1.05 sec. (0.05 for boundaries and 0.83 for bisection), while achieving an average H_∞ norm of 70.6150 and STD of 0.0195. It might be interesting for the reader to know that the controller design with the MATLAB function `hinfmix` for this problem takes 2.03 sec.

It should be noted that increasing the skip factor beyond $1 - 1/\sqrt{3} = 0.4226$ allows the algorithm to assign the same rank even to the first 3 individuals, which would remove the elitism strategy. If $skip\_factor \geq 1 - 1/\sqrt{pop\_size} \approx 1$, the

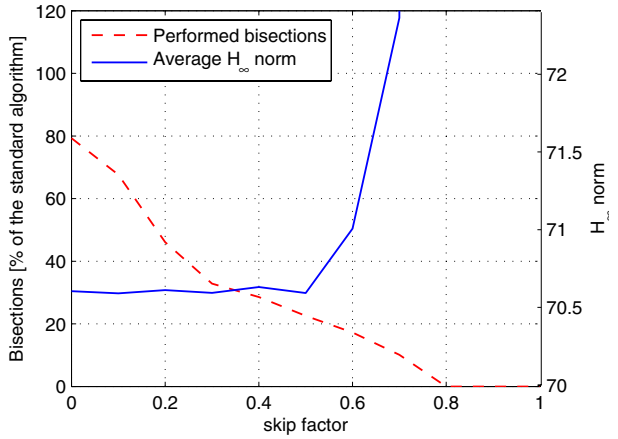

Fig. 3. Number of performed bisections and achieved H_∞ norm for different skip factor values for the spinning satellite problem

TABLE II

INFLUENCE OF THE MODIFICATIONS TO THE POPULATION BISECTION

Skip factor	Bisections [% of stnd.]	H_∞ norm mean	STD
0 $\star$	82.30	70.5567	0.0172
0 a	80.35	70.6238	0.0777
0 b	79.61	70.5933	0.0286
0	79.27	70.6086	0.0460
0.3 $\star$	68.62	70.5488	0.0112
0.3 a	64.46	70.5521	0.0108
0.3 b	33.43	70.6126	0.0294
0.3	32.86	70.5974	0.0541

same rank is assigned to all individuals, i.e. the algorithm is reduced to a purely random search.

To understand the way the modifications proposed in the previous section work, we now consider them one by one - see Table II. Their influence is shown for skip factors 0 and 0.3. The $\star$ indicates, that no modification is used. Those results are improved by adding a limit for the lower bound (case a) or by only using the modification for the first bisection point (case b). As can be seen from the table this has a much stronger effect on the computation savings. Finally, by combining those two approaches even better results are achieved. It is important to note that the reduction of the bisections down to 79.27% when skip factor is 0, is achieved without loss of information - the controllers are assigned the same fitness values as in the case when the precise H_∞ norm is computed.

B. Two Mass-Spring System

The two-mass-spring robust control design benchmark problem - also known as ACC benchmark problem - was proposed in [8]. The plant represents two carts, connected with a spring to each other. A control force is applied to the first cart and the position of the 2nd is to be controlled. It is assumed that the masses of the cart are equal to 1, but the spring constant is not precisely known ($0.5 \leq k \leq 2$),

TABLE III

RESULTS FOR THE TWO-MASS-SPRING PROBLEM

Skip factor	Bisections [% of stnd.]	H_∞ norm mean	STD
0	63.36	1.1891	0.1079
0.1	55.07	1.1593	0.0663
0.2	45.00	1.1506	0.0822
0.3	38.25	1.1792	0.1446
0.4	36.24	1.1651	0.0694
0.5	22.27	1.2674	0.0933
0.6	9.20	1.5408	0.3225
0.7	1.21	$1.79\,10^7$	$4.41\,10^7$

and a controller is to be designed that guarantees stability for the given range of values for the spring constant. Using an LFT representation of the uncertainty, this problem can be expressed in terms of a generalized plant (1), where the system matrices are

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1.25 & 1.25 & 0 & 0 \\ 1.25 & -1.25 & 0 & 0 \end{bmatrix}; \; B_w = \begin{bmatrix} 0 \\ 0 \\ -0.75 \\ 0.75 \end{bmatrix}; \; B_u = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$C_z = \begin{bmatrix} 1 & -1 & 0 & 0 \end{bmatrix}; \quad C_y = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$$

$$D_{zw} = 0; \quad D_{zu} = 0; \quad D_{yw} = 0.$$

Here w and z represent the input-output channel which is closed by feedback through an uncertain matrix Δ with norm less than 1, i.e. $w = \Delta z$.

Designing a 4th order controller using standard tools one can achieve a H_∞ norm of 0.6014. Now assume that we want to design a 2nd order controller, using the proposed evolutionary approach. Since the plant is SISO this leads to 5 decision variables. The population size was set again to 20, but the number of generations was increased to 1000 to achieve good reproducibility of the results. Average results for 10 runs for skip factors between 0 and 0.7 are presented in Table III and in Figure 4. Again, up to a skip factor of about 0.5 one can achieve significant computation savings without big loss in performance.

The computation time for one optimization run when using the standard bisection (C implementation) is 49.75 sec. and with population bisection (skip factor 0) this time is 44.86 sec. (26.82 sec. are spent on computing the boundaries and 21.16 sec. for bisection checks). Using the proposed cheaper bounds the overall computation time is reduced to 16.68. For skip factor 0.4 the time is 9.60 sec. (0.03 spent on bound. computation and 8.02 on bisection checks), while still achieving good results (average H_∞ norm 1.2218, STD 0.0963).

A randomly picked controller among the ones designed with skip factor of 0.4 achieves a H_∞ norm of 1.13. The state space model is ($B_K = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$)

$$A_K = \begin{bmatrix} 0 & -3.057 \\ 1 & -1.376 \end{bmatrix}; \; C_K = \begin{bmatrix} -4.102 \\ -3.096 \end{bmatrix}^T; \; D_K = 2.778$$

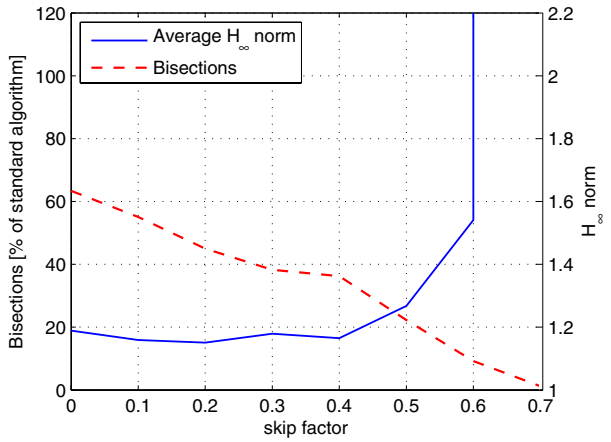

Fig. 4. Number of performed bisections and achieved H_∞ norm for different skip factor values for the two-mass-spring problem

Due to the inherent conservatism of this robust design technique, the controller robustly stabilizes the plant actually for the range $0.34 \leq k \leq \infty$.

C. Two-Link Flexible Manipulator

The last example is taken from [10]. The system represents a two-link planar robot manipulator. Due to the varying geometry the inertia of the system changes during operation, which causes also changes in the system response. Requirements in terms of robust stability under those variations and command tracking performance are given in terms of shaping filters. A full-order H_∞ controller designed for the resulting generalized plant model has 14^{th} order and achieves a H_∞ norm of 3.82.

The results for different skip factor values when designing a second order controller are presented in Table IV and in Figure 5. For each skip value 5 runs are performed and the average results are presented. Each optimization is performed with a population size of 50 controllers and 2000 generations. In contrast to the previous examples here the proposed cheaper bounds are used for all optimization runs ($\Gamma = 0$). Because the ratio of number of bisections between standard and population bisection method is not a meaningful comparison, here the computation time is given. For the standard method and bounds according to (6) the time is 833 seconds.

VI. CONCLUSIONS

A new approach to low order controller design is proposed and tested on 3 design problems. The method is based on combining a evolutionary search with an efficient bisection approach. The examples illustrate that by using computationally cheap boundaries and with small loss of information the overall computation time can be reduced by a factor of 10.

Design problems with a large number of decision variables and the initialization of the controller search are issues of current research.

TABLE IV

RESULTS FOR THE TWO-LINK FLEXIBLE MANIPULATOR PROBLEM

Skip factor	Time [sec]	H_∞ norm mean	H_∞ norm STD
0	432.79	9.9492	2.2616
0.1	344.93	9.7795	1.8187
0.2	298.68	10.1905	1.1601
0.3	227.65	9.0235	0.3303
0.4	122.67	10.4607	3.0890
0.5	85.47	9.4452	0.5784
0.6	67.36	9.3681	0.5942
0.7	67.18	12.6639	3.1250
0.8	44.19	13.1278	2.0433
0.9	22.73	1795.65	1780.50
1.0	24.49	9479.72	19862.77

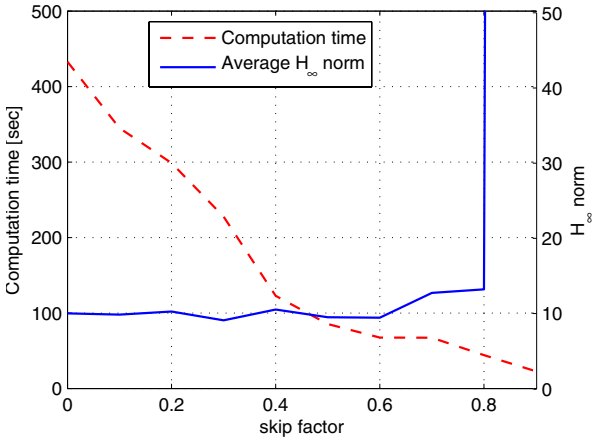

Fig. 5. Number of performed bisections and achieved H_∞ norm for different skip factor values for the two-link flexible manipulator problem

REFERENCES

[1] R. Skelton, T. Iwasaki and K. Grigoriadis, "A unified approach to linear control design", Taylor & Francis, London, UK, 1997

[2] A. Farag and H. Werner, "A Riccati - Genetic Algorithms Approach to Fixed-Structure Controller Synthesis", proc. of *American Control Conference*, Boston, USA, 2004

[3] S. Boyd, V. Balakrishnan and P. Kabamba, "On Computing the H_∞ norm of a tranfer matrix", proc. of *Amarican Control Conference*, Atlanta, USA, 1988, pp.396–397.

[4] Zbiniew Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Springer, 1996

[5] Akira Oyama, "Wing Design Using Evolutionary Algorithms", PhD thesis, Department of Space Engineering, Tohoku University, 2000

[6] "Genetic Algorithm and Direct Search Toolbox - for use with MATLAB", *Users Guide*, The MathWorks, 2004

[7] G. Balas, J. Doyle, K. Glover, A. Packard and R. Smith, "μ-Analysis and Synthesis Toolbox", The MathWorks, 1998, pp.6.14–6.18

[8] Bong Wie, Dennis Bernstein. "Benchmark Problems for Robust Controller Design", *Journal of Guidance Control and Dynamics*, vol.15, No.5, 1992

[9] Peter Thompson. "Classical/H_2 Solution for a Robust Controller Design Benchmark Problems", *Journal of Guidance Control and Dynamics*, vol.18, No.1, 1995

[10] P. Apkarian, G. Becker, P. Gahinet and H. Kajiwara,"LMI Techniques in Control Engineering from Theory to Practice", Workshop notes CDC, Kobe, Japan, 1996