**Proceedings of the 17th World Congress**
**The International Federation of Automatic Control**
**Seoul, Korea, July 6-11, 2008**

# A Unified Approach To Control Design Using A Cooperative Co-Evolutionary Bisection Algorithm

**A. O. Farag** *

* *e-mail: elgiloshi@yahoo.co.uk*

**Abstract:** This paper proposes a novel algorithm named as a Co-evolutionary Cooperative Bisection (CCB) algorithm as master tool for solving variety of control problems. The proposed algorithm is computationally attractive and has very favorable convergence properties. The numerical superiority of the CCB-algorithm is back to two main factors, the numerical efficiency of $\gamma$-Bisection algorithm, and the multi-population evolutionary algorithm proposed in this paper (*Cooperation + Parallelism*). To demonstrate the potential of the proposed algorithm a method for designing fixed-structure controllers that minimize an upper bound on the singular value $\mu$ is presented. The problem considered here is of significant practical interest, since many industrial controllers are constrained in order/structure and required to meet high robustness demands. Moreover, even if the controller order or structure are not restricted, the proposed algorithm has the advantage of being able to optimize over for stability multipliers and controllers simultaneously, thereby improving the chances of converging to the global minimum. Numerical examples given here confirm the computational efficiency and the excellent convergence properties of the proposed CCB-algorithm.

## 1. INTRODUCTION

Even though evolutionary algorithms have been with us for more than 60 years by now, significant research into the use of co-evolutionary algorithms did not really begin until the early 1990s. The notion of cooperative co-evolution (CCE) was first introduced and made popular in Potter and Jong [1994], Potter [1997]. They have shown that CCE-algorithms outperforms standard single population evolutionary algorithms when applied to static function optimization problems. Although cooperative co-evolution was first introduced to solve difficult optimization problems by means of problem decomposition, its performance for such tasks was not satisfying in some applications. One main reason for this, is the weak and not complete understanding of its dynamics. Among the recent attempts to understand the dynamic of co-evolution algorithms is the paper by Popovici and De Jong in 2005. One important result of their study is the observation that the rules controlling the performance of simple evolutionary algorithms (i.e. single population) do not transfer directly to co-evolutionary algorithms.

They have shown that the interaction between the collaboration scheme and landscape of the problem can lead to a conflicting behavior in case of CCE type algorithms. A similar conclusion on the influence of the collaboration scheme on the performance of CCE-algorithms was also highlighted in Wiegand et al. [2001]. That paper consist of an empirical study of collaboration methods based on a huge number of experimental results. Based on their very interesting study a number of important conclusions about the performance of CCE-algorithms have been learned. For example it is evident that using an optimistic approach is generally the best mechanism for collaboration credit assignment (i.e select the best). They also manage to drive some guidelines on the relation between the degree of nonlinearity of the problem landscape and the suitable number of collaborators as well as the selection pressure. Specifically their study has shown that for a simple problem that is linearly separable, a greedy approach to collaborator selection will do well, moreover, the number of collaborators may be limited (i.e. one or two).

In 2003 Wiegand has introduced the most complete and powerful analysis of co-operative co-evolutionary algorithms. In his work he has demonstrated the relationship between the notion of cooperative co-evolution and symmetric games in evolutionary game theory Wiegand et al. [2002a],Wiegand et al. [2002b]. This provided a natural way to study cooperative co-evolution from a dynamical systems perspective and provided many new insights.

Let us remember that CCE-algorithms attacks complex search spaces by breaking them into parts, evolving the parts separately, and then assembling the parts into a complete optimal solution. The critical question that need to be answered now is "what makes a good part". Wiegnad has established the importance of this question by pointing out that CCE-algorithms do not optimize when they use certain straightforward notions of what constitutes a good part. Moreover, he has shown that CCE-algorithms such as the one described by potter exhibit a behavior called *relative overgeneralization* Panait et al. [2004]. In fact, rather than finding complete objects which are optimal with respect to the problem, CCE-algorithms tend to find objects which are robust (e.g. see Wiegand and Potter [2006]) under a change of parts, the so called *robust resting balance*. His results has opened a new question: how can we move the CCE-algorithms out of local optimal solutions? (i.e. encourage it to find global optima).

In 2005 Bucci has presented one of the most successful attempts to answer the above question Bucci and Pollack [2005]. The main idea of his approach is to modify CCE-algorithms such that individuals are compared using Pareto dominance like technique in a similar manner to that used with Multi-objective Optimization Fonseca and Fleming [1995]. Another very interesting attempt to improve the dynamic behavior of CCE-algorithms was introduced in Panait [2006] and Panait et al. [2003]. The key step here is to bias CCEA to search for ideal collaborations rather than robust resting balance, this idea has produced a successful algorithms Panait et al. [2006].

This paper introduces an alternative approach to overcome problems associated with CCE-algorithms. The key idea of this approach is to modify the classical CCE-algorithm proposed in Potter and Jong [1994] by introducing *Parallelism* into the core of the CCE-algorithm. The motivation for this approach is based on the previous observations Cantu-Paz [1992] which had concluded that parallel evolutionary algorithms perform better than non-parallel one in terms of global convergence.

The new algorithm has been specifically developed to tackle complex control design problems. more precisely, *to develop reliable and numerically efficient robust-fixed-structure controller synthesis*. A major issue in control design is robustness against model uncertainty, in fact almost all practical situations involves some sort uncertainty that bother control engineers. Robust control is also useful to handle other difficult control problems, for instance the dynamic behavior of a nonlinear plant can be captured by a family of linearized models or by a Linear Parameter Varying LPV model Kajiwara et al. [1999], and then solved using linear control theory. Over the years many successful robust control design techniques have been proposed, for example $H_\infty$ techniques Zhou et al. [1996], Chilali and Gahinet [1996], $\mu$-synthesis Balas and Doyle [1994], Balas et al. [1993], and robust $H_2$ techniques Packard and Doyle [1987], How et al. [1994], Banjerdpongchai and How [2000] and many others.

Most existing robust control syntheses are formulated as non-convex optimization problems, which make them unattractive computation wise. For instance, $\mu$-synthesis which is one of the most widely used design techniques, is usually solved by searching over *controllers* and *multipliers*. Simultaneous Optimization over-both controllers and multipliers is a non-convex optimization problem which has no complete solution to date. A commonly used approach is to retain convexity by iterating once over controllers and once over multipliers, this technique may work, but in most cases converge to sub-optimal solutions.

This paper proposes a novel Co-evolutionary Cooperative Bisection (CCB) algorithm to address the above problem. The proposed algorithm is based on a combined use of CCE-algorithms and Algebraic Riccati Equation (ARE) solvers. The work presented here is a continuation of a previous successful combination of genetic algorithms and ARE-solvers i.e. see Farag and Werner [2004], [2005a], and [2005b]. The newly developed technique is not only computationally faster than previous techniques, but also poses superior convergence properties as it will be shown in next sections.

The paper is organized as follows: Section 2 introduces a unified approach to control design based on a Riccati Equation Formulation then proposes the $\gamma$-bisection algorithm. The CCB-algorithm is proposed in section 3. Section 4 presents a new method for designing low order controllers that minimizes an upper bound on the singular value $\mu$. Conclusions are drawn in section 5.

## 2. A UNIFIED CONTROL DESIGN SYNTHESIS

Let $A(\theta,\beta)$, $Q(\theta,\beta,\gamma)$ and $R(\theta,\beta,\gamma)$ be real $n\times n$ matrices with $Q(\theta,\beta,\gamma)$ and $R(\theta,\beta,\gamma)$ are symmetric. Now consider the parameterized algebraic Riccati equation

$$A(\theta,\beta)^T P + PA(\theta,\beta) - (PB(\theta,\beta)+V(\theta,\beta,\gamma))R(\theta,\beta,\gamma)\cdot (B(\theta,\beta)^T P + V(\theta,\beta,\gamma)^T) + Q(\theta,\beta,\gamma) = 0 \quad (1)$$

where $B(\theta,\beta)$ and $V(\theta,\beta,\gamma)$ are real matrices of compatible dimensions. It is assumed here that all these matrices are continuous functions of a common parameter vectors $\theta \in R^{n_\theta}$, $\beta \in R^{n_\beta}$ and $\gamma \in R$.

Note: The importance of solving Equation 1 comes from the fact that several interesting control design synthesis can be brought to this specific form.

Finding vectors $\theta$, $\beta$, scalar $\gamma$ and a positive definite symmetric matrix $P \in R^{n\times n}$ that satisfy this equation is a non-convex problem. Note however that for any fixed value $\theta = \theta_0$, $\beta = \beta_o$, and $\gamma = \gamma_o$ the problem can be solved efficiently via Riccati solvers see Arnold and ALaub [1984]. The decision vectors $\theta$ and $\beta$ will usually represent controllers and multipliers respectively, while scalar $\gamma$ represents some performance measure that need be optimized.

The main goal of this section is to develop an efficient algorithm for solving the following problem:

**Problem 1:**

$$\min_{\theta,\,\beta,\,P} \gamma$$

subject to:

$$A(\theta,\beta)^T P + PA(\theta,\beta) - (PB(\theta,\beta)+V(\theta,\beta,\gamma))R(\theta,\beta,\gamma)\cdot (B(\theta,\beta)^T P + V(\theta,\beta,\gamma)^T) + Q(\theta,\beta,\gamma) = 0 \quad (2)$$

$$R(\theta,\beta,\gamma) < 0 \quad (3)$$

$$P = P^T > 0 \quad (4)$$

Now let us consider a simplified version of Problem-1.

**Problem 2:** For given $\theta = \theta_0$ and $\beta = \beta_o$,

$$\min_{P} \gamma$$

subject to:

$$A(\theta_0,\beta_0)^T P + PA(\theta_0,\beta_0) - (PB(\theta_0,\beta_0)+V(\theta_0,\beta_0,\gamma))R\cdot (\theta_0,\beta_0,\gamma)(B(\theta_0,\beta_0)^T P + V(\theta_0,\beta_0,\gamma)^T) + Q(\theta_0,\beta_0,\gamma) = 0 \quad (5)$$

$$R(\theta_0,\beta_0,\gamma) < 0 \quad (6)$$

$$P = P^T > 0 \quad (7)$$

This problem can be solved using the $\gamma$- Bisection Algorithm Boyd et al. [1989] shown in Figure 1.

Choose a suitable values for scalars $\gamma_U$ and $\gamma_L$
**repeat:**
$\gamma_0 = (\gamma_U + \gamma_L)/2$
If $R(\theta_0, \beta_0, \gamma_0) < 0$
$\gamma_U = \gamma_0$; If a solution to Equation-(5) exists;
$\gamma_L = \gamma_0$; Otherwise;
Otherwise
$\gamma_L = \gamma_0$;
**until** $(\gamma_U - \gamma_L \leq \varepsilon\gamma_L)$
$\gamma = (\gamma_U + \gamma_L)/2$
where $\varepsilon$ is a small number (i.e. $\varepsilon = 0.01$).

Fig. 1. The $\gamma$- Bisection Algorithm

Once this algorithm is terminated it will return some optimal $\gamma$ and a corresponding solution $P > 0$.

For convenience let us represent the computations performed using the above $\gamma$-Bisection algorithm by the mapping function $\mathcal{M}(\theta, \beta)$,

$$\gamma_0 = \mathcal{M}(\theta_0, \beta_0) \tag{8}$$

Note: this mapping exists as long as $A(\theta_0, \beta_0)$ is stable.

Let us observe that Problem-2 is solvable in a straight forward manner via the bisection algorithm, moreover, it is a special case of problem-1 ($\theta$ and $\beta$ are given). Thus, all needed to solve Problem-1 is to search for the vectors $\theta$ and $\beta$ using GA, while using $\gamma$ as an objective function computed through the mapping $\mathcal{M}(\cdot)$. A typical population structure to solve Problem-1 is given in Table 1 below.

| GA Population | | Bisection+ARE solvers | Fitness |
|---|---|---|---|
| Controllers | Multipliers | | |
| $\theta_1$ | $\beta_1$ | $\gamma_1 = \mathcal{M}(\theta_1, \beta_1)$ | $f(\gamma_1)$ |
| $\theta_2$ | $\beta_2$ | $\gamma_2 = \mathcal{M}(\theta_2, \beta_2)$ | $f(\gamma_2)$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\theta_{n_p}$ | $\beta_{n_p}$ | $\gamma_{n_p} = \mathcal{M}(\theta_{n_p}, \beta_{n_p})$ | $f(\gamma_{n_p})$ |

Table 1. GA Population Structure

Experience has shown that a standard GA may work well when applied to Problem-1, but only when the size of the decision vectors $\theta$ and $\beta$ are relatively small ($n_\theta \leq 10$ and $n_\beta \leq 5$). On the other hand if the problem size is big ($n_\theta + n_\beta > 15$), then GA has difficulties in solving Problem-1, and may require very long runs to converge to acceptable solutions. Interestingly, if $n_\beta = 0$, then GA remain efficient in solving Problem-1 even with large values of $n_\theta \approx 40$, which is bigger that the sum $n_\theta + n_\beta = 15$. In other words, the decision vector $\beta$ plays *a different role* when solving Problem-1. To clarify this claim let us consider Table 2 below.

| Random chromosomes | Objective value |
|---|---|
| (Controller A , Multiplier A) | 2.5 |
| (Controller B , Multiplier B) | 1.9 |
| (Controller C , Multiplier C) | 2.0 |

Table 2. Objective values of a typical run

This table shows a fictitious scenario that may happen when using the population structure shown on Table 1. Note that each chromosome codes both the controller and the multiplier variables ($\theta$, $\beta$). Based on this table it is clear that the pair (Controller B , Multiplier B) achieve the best fitness, so it has the highest chances of surviving in next generations.

Let us now disassemble these pairs and compute the objective function of every possible combination as shown in Table 3,

| | Multiplier A | Multiplier B | Multiplier C |
|---|---|---|---|
| Controller A | **2.5** | 7.2 | 1.2 |
| Controller B | 4.1 | **1.9** | 3.0 |
| Controller C | 1.8 | 1.5 | **2.0** |

Table 3. Fitness values of the disassembled chromosomes

The question now is how to assign fitness? There are many ways of doing that for instant a fairer fitness assignment scheme for controllers (A,B,C) would be (1.2,1.9,1.5) respectively (minimum of each row). Similarly, a fairer fitness assignment for multipliers (A,B,C) would be (1.8,1.5,1.2) respectively (minimum of each column). It is interesting to observe that the best combined fitness is (controller A + Multiplier C), so the controller A which has the worst fitness on Table 2 (i.e. very low chances of surviving) deliver the best fitness when combined with multiplier C.

So, if a standard single population GA is used to solve this problem with a population structure shown in Table 1 the controller A is combined with the Multiplier A which give the worst fitness and will be removed from future generations. This possibility explains why the convergence of GA is much worst when a new *influential* player is introduced i.e. *Multipliers*. The way GA is dealing with the problem at the moment is to think in terms of a combined fitness which may work, but for sure is not the best option. One possibility is to consider all possible combinations, then to compute the fitness as shown above (i.e rows for controller and column for multipliers), but this option will increase the computational duty drastically (the number of fitness evaluations in each step is squared).

The option taken here is to separate controllers and multipliers in different populations (i.e. species), such that the fitness of all controllers are computed with a single representative multiplier $\beta^*$, similarly, all multipliers are tested against a single representative controller $\theta^*$. Which means that the controllers evolve in a separate population while *Cooperating* with the multipliers population and vice versa.

The above discussion opens the door to new directions in which complex control design problems can be tackled, namely, to use Co-evolutionary Cooperative algorithms. The next section presents inner structure of the CCB-algorithm.

## 3. A COOPERATIVE CO-EVOLUTIONARY BISECTION ALGORITHM

Let us start by introducing the following shorthand notations:

$P_\theta^i$ : $i^{th}$ controllers population.
$P_\beta^j$ : $j^{th}$ multipliers population.
$\theta_i^*$ : fittest controller in $P_\theta^i$.
$\beta_j^*$ : fittest controller in $P_\beta^j$.
$n_p$ : Populations size (same for all).
$n_K$ : number of controllers populations.
$n_D$ : number of multipliers populations.
$n_g$ : Migration interval.

**Initialization** Set $k = 1$, and generate random controllers/multipliers populations ($P_\theta$, $P_\beta$).

**Warming-Up-phase**

(1) Compute the fitness of $P_\theta^i$ $(i = 1, ..., n_K)$, using some $\beta_0$.
(2) If stabilizing controllers are found in all populations, then save the best controllers $\{\theta_1^*, \theta_2^*, ..., \theta_{n_K}^*\}$ and terminate this phase; otherwise continue.
(3) Evolve $P_\theta^i$, $(i = 1, ..., n_K)$, and go-back to step-1.

**Process-Multipliers**

(1) Compute the fitness of $P_\beta^j$ using a random $\theta_i^*$ selected from the set $\{\theta_1^*, \theta_2^*, ..., \theta_{n_K}^*\}$ for $(j = 1, ..., n_D)$.
(2) Update the set $\{\beta_1^*, \beta_2^*, ..., \beta_{n_D}^*\}$.
(3) Apply Migration (every $n_g$ cycles, see Remark 4).
(4) Evolve $P_\beta^j$, $(j = 1, ..., n_D)$.

**Process-Controllers**

(1) Compute the fitness of $P_\theta^i$ using a random $\beta_j^*$ selected from the set $\{\beta_1^*, \beta_2^*, ..., \beta_{n_D}^*\}$ for $(i = 1, ..., n_K)$.
(2) Update the set $\{\theta_1^*, \theta_2^*, ..., \theta_{n_K}^*\}$.
(3) Apply Migration (every $n_g$ cycles, see Remark 4).
(4) Evolve $P_\theta^i$, $(i = 1, ..., n_K)$.

**Termination-Criteria** If the maximum number of iterations is reached then *stop*; otherwise set $k = k + 1$, go to *Process-Multipliers*.

Figure 2 shows the population structure associated with the above algorithm.

**Remark 1:** In the warming up phase the algorithms searches for stabilizing controllers, i.e. the controller parameters (i.e. $\theta$), so at this stage the multipliers populations (i.e. $P_\beta^j$, $j = 1, ..., n_D$) are not evolved. The fitness of each controller is computed by assuming identity multipliers ($\beta_0$ =Identity) and according to the following objective function:

$$f(\theta_i) = \begin{cases} \mathcal{M}(\theta_i, \beta_0), & \text{if } \bar{A} \text{ is stable} \\ \kappa(\bar{A}(\theta_i)) + \pi_P, & \text{if } \bar{A} \text{ is unstable} \end{cases}$$

where $\kappa(\bar{A})$ stands for maximum real part of the eigenvalues of $\bar{A}$, and $\pi_P$ is a penalty for destabilizing controllers.

The use of the penalty $\pi_P$ means that the algorithm searches for stabilizing controllers at early generations, and switch to the task of $\gamma$-minimization at later generations. This claim is based on the fact that at later stages the mutation operator is restricted (older generations has less mutation i.e. nonuniform mutation). This technique of handling hard constrains is known as *Death penalty* refer to Morales and Quezada [1998] and Michalewicz [1996] for further details.

Thus, with the help of the death penalty $\pi_P$ the controller populations $P_\theta^i$, $i = 1, ..., n_K$ are filled with stabilizing controllers after few iterations. Note that this phase is terminated as soon as there is at least one stabilizing controller in each population, in other words the set $\{\theta_1^*, \theta_2^*, ..., \theta_{n_K}^*\}$ is filled.

**Remark 2:** In the *Process Multipliers* step the fitness of each multiplier is computed according to:

$$f(\beta_j) = \mathcal{M}(\theta_i^*, \beta_j)$$

Note that the entire $j^{th}$ multipliers population use a unique random $\theta_i^*$.

**Remark 3:** In the *Process Controllers* step the fitness of each controller is computed according to:

$$f(\theta_i) = \begin{cases} \mathcal{M}(\theta_i, \beta_j^*), & \text{if } \bar{A} \text{ is stable} \\ \kappa(\bar{A}(\theta_i)) + \pi_P, & \text{if } \bar{A} \text{ is unstable} \end{cases}$$

Note that the entire $i^{th}$ controllers population use a unique random $\beta_i^*$.

**Remark 4:** *Migration* is implemented in standard manner, every $n_g$ generations with a complete-net-topology see Cantu-Paz [2001].

The real potential of this algorithm will be demonstrated now by applying it to some interesting control problems.

## 4. $\mu$-SYNTHESIS BASED ON THE CCB-ALGORITHM

The CCEB developed in the previous section is used here to construct a new method for designing fixed structure controllers that minimize an upper bound on the structured singular value $\mu$. A key step in the proposed design procedure is to show that this problem can be formulated as Problem-1 considered in the previous section.

### *4.1 Problem Formulation*

Consider the generalized plant $P(s)$ shown in Figure 3 with a feedback controller $K(s)$ and a stable perturbation $\Delta(s)$ that satisfies $\|\Delta(s)\|_\infty < 1$ and has a given structure $\mathbf{\Delta}$. Partition $P(s)$ as

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$$

and let $M(s)$ denote the lower fractional transformation

$$M = P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21}$$

A typical design problem with a given performance and robustness specifications, involves the search for a controller that minimizes

$$\sup_\omega \mu_\Delta(M(j\omega))$$

where $\mu_\Delta$ is the structured singular value

$$\mu_{\mathbf{\Delta}}(M) := \frac{1}{\min\{\bar{\sigma}(\Delta) : \Delta \in \mathbf{\Delta}, \det(I - M\Delta) = 0\}} \quad (9)$$

A widely used approach - known as D-K iteration - to "solve" this problem is to minimize an upper bound on

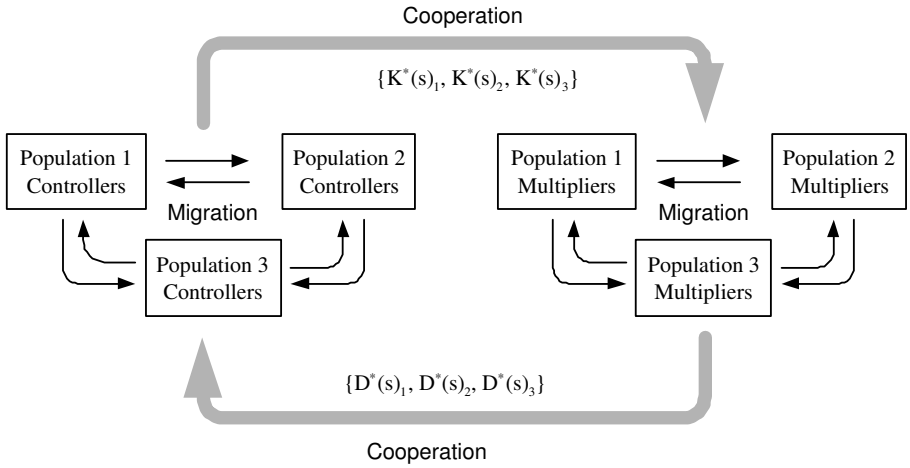


Fig. 2. The CCB-Algorithm

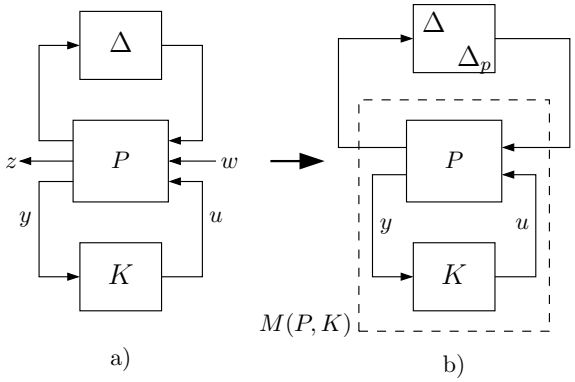


Fig. 3. Generalized plant

$\mu_\Delta$: introduce stable and minimum phase scaling matrices $D(s)$ that satisfy $D(s)\Delta(s) = \Delta(s)D(s)$, and try to find a controller that solves

$$\min_K \inf_D \|DMD^{-1}\|_\infty \tag{10}$$

by iteratively solving for $K$ and $D$. Solving for $K$ with $D$ fixed is a standard $H_\infty$ problem; on the other hand, when $K$ is fixed one searches for $D(j\omega)$ point-wise on a frequency grid, and tries to find a transfer matrix $D(s)$ that fits the resulting magnitude curve (e.g. see Lewin and Parag [2003]).

A major drawback of this approach is the fact that the order of the controller equals the order of the generalized plant plus twice the order of the scaling $D(s)$, which may lead to high controller orders. Moreover, in this framework it is not possible to impose constraints on the controller structure. The following section proposes an alternative approach for solving the above problem, that avoids these drawbacks.

### *4.2 Control Synthesis*

Let a state space realization of the generalized plant $P(s)$ in Figure (3) be

$$\begin{aligned} \dot{x} &= Ax + B_w w + Bu \\ z &= C_z x + D_{zw} w + D_z u \\ y &= Cx + D_w w \end{aligned}$$

where $x \in R^n$ is the state vector, $u \in R^{n_u}$ is the control input, $w \in R^{n_w}$ represents external inputs, $y \in R^{n_y}$ is the measured output, and $z \in R^{n_z}$ represents the outputs to be controlled. The matrices $C_z$, $D_{zw}$, $B_w$, $D_z$ and $D_w$ are tuning parameters to be chosen to meet some design requirements. Similarly, let

$$\begin{aligned} \dot{\zeta}(t) &= A_K \zeta(t) + B_K y(t) \\ u(t) &= C_K \zeta(t) + D_K y(t) \end{aligned} \tag{11}$$

be a state space realization of the controller $K(s)$, where $\zeta(t) \in R^{n_c}$ is the controller state vector, and $n_c \leq n$ is the order of the controller.

Now, let

$$P_D(s) = D_L(s)P(s)D_R(s) \tag{12}$$

denote the augmented generalized plant shown in Figure 4, where

$$D_L(s) = \mathrm{diag}(D(s), I_{n_y}), \quad D_R(s) = \mathrm{diag}(D^{-1}(s), I_{n_u}).$$

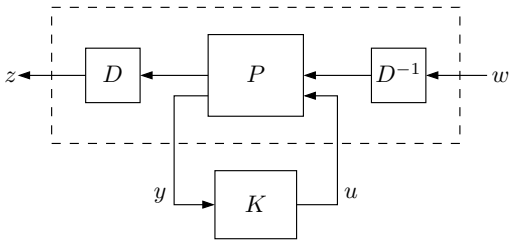


Fig. 4. Closed loop with D-scaling

Moreover, let

$$\begin{aligned} \dot{x} &= \hat{A}x + \hat{B}_w w + Bu \\ z &= \hat{C}_z x + \hat{D}_{zw} w + \hat{D}_z u \\ y &= Cx + \hat{D}_w w \end{aligned}$$

be a state space model of $P_D(s)$. A realization of the nominal closed-loop transfer matrix $M(s)$ is then

$$\begin{aligned} \dot{x}_{cl} &= \bar{A}x_{cl} + \bar{B}w \\ z &= \bar{C}x_{cl} + \bar{D}w \end{aligned}$$

where

$$\bar{A} = \begin{bmatrix} \hat{A} + BD_K C & BC_K \\ B_K C & A_K \end{bmatrix}, \; \bar{B} = \begin{bmatrix} \hat{B}_w + BD_K \hat{D}_w \\ B_K \hat{D}_w \end{bmatrix}, \tag{13}$$

$$\bar{C} = \left[ \hat{C}_z + \hat{D}_z D_K C \;\; \hat{D}_z C_K \right], \;\; \bar{D} = \hat{D}_{zw} + \hat{D}_z D_K \hat{D}_w \tag{14}$$

Using the real bounded lemma Boyd et al. [1994], we have the following result.

*Theorem 4.1.* The matrix $\bar{A}$ is stable and $\|M(s)\|_\infty < \gamma$ if and only if there exists a matrix $X = X^T > 0$ that satisfies

$$\begin{bmatrix} \bar{A}^T X + X\bar{A} + \gamma^{-1}\bar{C}^T\bar{C} & X\bar{B} + \gamma^{-1}\bar{C}^T\bar{D} \\ \bar{B}^T X + \gamma^{-1}\bar{D}^T\bar{C} & -\gamma I + \gamma^{-1}\bar{D}^T\bar{D} \end{bmatrix} \leq 0 \tag{15}$$

Using the Schur complement, inequality (15) holds if and only if

$$\bar{A}^T X + X\bar{A} + \gamma^{-1}\bar{C}^T\bar{C} - (X\bar{B} + \gamma^{-1}\bar{C}^T\bar{D})R^{-1}(\bar{B}^T X + \gamma^{-1}\bar{D}^T\bar{C}) \leq 0 \tag{16}$$

and

$$R = -\gamma I + \gamma^{-1}\bar{D}^T\bar{D} \leq 0 \tag{17}$$

When this result is applied to the minimization problem (10), the solution $X$ will be on the boundary; therefore one can replace the inequality by an equation and consider

$$\bar{A}(\theta,\beta)^T X + X\bar{A}(\theta,\beta) + \gamma^{-1}\bar{C}^T(\theta,\beta)\bar{C}(\theta,\beta) - (X\bar{B}(\theta,\beta) + V(\theta,\beta,\gamma))R(\theta,\beta,\gamma)^{-1}(\bar{B}^T(\theta,\beta)X + V(\theta,\beta,\gamma)^T) = 0 \tag{18}$$

instead of (16), where

$$V(\theta,\beta,\gamma) = \gamma^{-1}\bar{C}^T(\theta,\beta)\bar{D}(\theta,\beta),$$

$\theta$, $\beta$ are vectors containing the controller $K(s)$ and scaling matrix $D(s)$ (i.e. multipliers) variables respectively.

Thus, the condition $\|DMD^{-1}\|_\infty < \gamma$ is met upon finding $\theta$, $\beta$ and $X = X^T > 0$ such equation (18-17) is satisfied. This problem takes the from of Problem-1, so it can be solved in a straight forward manner using the CCB algorithm.

*Note that using the CCB algorithms avoids the frequency gridding and transfer function fitting required in standard D-K iteration (see examples below). Moreover, controller order and structure can be chosen freely.*

### *4.3 Spinning Satellite Example*

This section applies the design technique developed in the previous section to a design example - control of a spinning satellite - taken from the Tutorial of the $\mu$ Analysis and Synthesis Toolbox Balas et al. [1993] .

The state space model of the satellite is given as

$$\begin{bmatrix} \dot{\beta}_x \\ \dot{\beta}_y \end{bmatrix} = \begin{bmatrix} 0 & 10 \\ -10 & 0 \end{bmatrix} \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 & 10 \\ -10 & 0 \end{bmatrix} \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix}$$

Actuator uncertainty in both input channels is modelled as multiplicative input uncertainty, with weights

$$W_{\delta_1} = \frac{10(s+4)}{s+200} \;\; ; \;\; W_{\delta_2} = \frac{10(s+24)}{3(s+200)}$$

for input 1 and 2, respectively. Both sensors measurements are assumed to be noisy, this is modelled by the sensor noise weight $W_n = w_n I_2$, where

$$w_n = \frac{12(s+25)}{5(s+6000)}.$$

Finally, the performance objective of achieving a tracking error of no more than 1% is expressed by a performance weighting filter $W_p = w_p I_2$, where

$$w_p = \frac{s+4}{2(s+0.02)}.$$

Performance and robustness specifications are expressed in the form of a $\mu$ synthesis problem by introducing the augmented uncertainty structure

$$\mathbf{\Delta}_a = \{diag(\Delta, \Delta_p) \; : \; \Delta \in \mathbf{\Delta}, \; \Delta_p \in \mathcal{C}^{(4\times 2)}\}$$

where

$$\mathbf{\Delta} = \{ \begin{bmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{bmatrix} \; : \; \delta_i \in \mathcal{C}\}$$

The corresponding scaling $D(s)$ must then have the form

$$D(s) = \{diag(D_{11}(s), D_{22}(s), I)\}$$

A controller then meets the performance and robustness requirements when the peak value of $\mu_{\Delta_a}$ is less than 1.

*Low Order Controller Using CCB* Now, we will use the CCB algorithm developed in the previous section to design a second order controller that meets all robustness and performance requirements. The chosen controller structure is given below:

$$A_k = \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix}, \; B_k = \begin{bmatrix} b_{11} & 0 \\ 0 & b_{22} \end{bmatrix},$$
$$C_k = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}, \; D_k = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$$

which means that the decision vector $\theta$ contains 12-decision variables, namely,
$\{a_{11}, a_{22}, b_{11}, b_{22}, c_{11}, c_{12}, c_{21}, c_{22}, d_{11}, d_{12}, d_{21}, d_{22}\}$.

The multiplier matrix $D(s)$ is selected as:

$$D(s) = \begin{bmatrix} \frac{d_1 s^2 + d_2 s + d_3}{s^2 + d_4 s + d_5} & 0 & 0 \\ 0 & \frac{d_6 s^2 + d_7 s + d_8}{s^2 + d_9 s + d_{10}} & 0 \\ 0 & 0 & I \end{bmatrix}$$

which implies that the decision vector $\beta$ contains 10 decision variables, namely, $\{d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}$.

When using standard GA to solve this problem each chromosome contains 22 variables ($\theta + \beta$). For given values of $\theta$ and $\beta$ the objective function $\gamma$ is computed using the mapping $\mathcal{M}(.)$. This $\gamma$ is an upper bound on the structured singular $\mu$ that must be minimized to meet the design requirements.

To understand how each part of the CCB algorithm function four techniques (options) are compared here: CCB

without cooperation (CCB-P), CCB with only two populations (CCB-C) (i.e one for controllers and one for multipliers), standard GA (single population), and the full CCB algorithm. The main objective here is to understand the individual effect of *cooperation* and *parallelism*. Moreover, to ensure fairness; all algorithms are forced to use the same number of function evaluations $N_{max}$ (i.e. number of times $\mathcal{M}(.)$ is called). This restriction is important due to the fact that CCB-algorithm is a multi-population algorithm and the observation that fitness evaluation is the most time consuming operation. Furthermore, all tested algorithms use identical evolution operations (Nonuniform mutation and Arithmetical Crossover see Michalewicz [1996]).

CCB algorithm is a heuristic algorithm that may return a different solution every time it is used. To reduce the effect of this heuristic behavior when comparing the above techniques a number of runs are performed, then four performance measures are computed to check performance. The proposed performance measures are: average-$\gamma$, best-$\gamma$, worst-$\gamma$ and Standard Deviation (SD). It is clear that smaller values of all these quantities are more favorable (for this example), for instance smaller SD implies better *Consistency* in achieving a given level of performance. Table 4 shows the results obtained in ten different runs (All computations were performed on a standard desktop computer, Pentium-IV 2.0G, 512MB Ram, the average computation time for $10,000$ function evaluations is $\approx 4$ minutes).

| Iteration # | GA $n_p=30$ $N=1000$ $z_m=1$ | CCB-P $n_p=30$ $N=250$ $z_m=4$ | CCB-C $n_p=20$ $N=750$ $z_m=2$ | CCB $n_p=20$ $N=150$ $z_m=10$ |
|---|---|---|---|---|
| 1 | 5.293 | 2.920 | 1.151 | 1.070 |
| 2 | 2.593 | 2.919 | 1.000 | 1.149 |
| 3 | 2.593 | 1.142 | 1.119 | 1.246 |
| 4 | 1.843 | 4.678 | 2.691 | 1.057 |
| 5 | 0.977 | 1.088 | 1.227 | 0.937 |
| 6 | 1.256 | 2.920 | 1.241 | 1.708 |
| 7 | 2.698 | 5.385 | 1.008 | 1.129 |
| 8 | 2.594 | 1.221 | 0.976 | 1.274 |
| 9 | 2.593 | 1.070 | 2.565 | 1.313 |
| 10 | 1.152 | 1.117 | 1.498 | 1.780 |
| $\gamma_{ave}$ | 2.359 | 2.446 | 1.448 | 1.266 |
| $\gamma_{worst}$ | 5.293 | 5.385 | 2.691 | 1.780 |
| $\gamma_{best}$ | 0.977 | 1.070 | 0.976 | 0.937 |
| $\sigma_\gamma$ | 1.238 | 1.599 | 0.641 | 0.276 |

Table 4. Results of ten different runs, with a maximum number of 30,000 function evaluations ($z_m$ : number populations).

Based on the results shown on Table 4 the CCB algorithm has a much better performance compared to standard GA in every aspect. For example, the average $\gamma$ obtained with CCB is 1.266 which is roughly 50% less than the one obtained with GA (i.e. 2.359), moreover, CCB algorithm is much more consistent in achieving this performance level (i.e. $\sigma_\gamma = 0.276$) which is approximately 4.5 times less than that of standard GA. Another clear advantage of the CCB algorithm is the fact that its worst expected performance is not far from average ($\gamma_{worst} = 1.780$), in contrast GA has a much worst behavior ($\gamma_{worst} = 5.385$). It is worth nothing that smaller values of SD do not necessarily mean smaller values of $\gamma_{worst}$.

Another important observation that can be drawn from Table 4, is the fact *Cooperation* plays an important role on the performance of proposed algorithm (i.e. CCB), this confirms the claims made in section 2. More precisely, the use of one controller to compute the fitness of the whole multiplier population and vice versa do improve convergence. On the other hand parallelism (PGA) has no clear advantage when add to pure GA, but it improves the performance when combined with a cooperative algorithm (compar column 3 and column 5 in Table 4).

The above claims can further be clarified by considering Figure 5, which shows the variation of $\gamma_{ave}$, $\gamma_{best}$ and $\gamma_{worst}$ against the maximum number of function evaluations $N_{max}$. Figure 5.a shows that as $N_{max}$ increases $\gamma_{ave}$ improves and the gap $\gamma_{worst} - \gamma_{best}$ decreases. While Figure 5.b shows the variation of the same quantities when (CCB-P) is used, which is just standard GA plus parallelism (i.e use of multi-populations), unfortunately, no clear improvement is observed compared to standard GA. In contrast, incorporating cooperation Figure 5.c (CCB-C) improves $\gamma_{ave}$, and reduces the gap $\gamma_{worst} - \gamma_{best}$ drastically. Finally, the performance of the full CCB algorithm which uses both parallelism and cooperation is shown in Figure 5.d, clearly this algorithm outperforms all others in terms of the achieved $\gamma_{ave}$ and the gap $\gamma_{worst} - \gamma_{best}$.

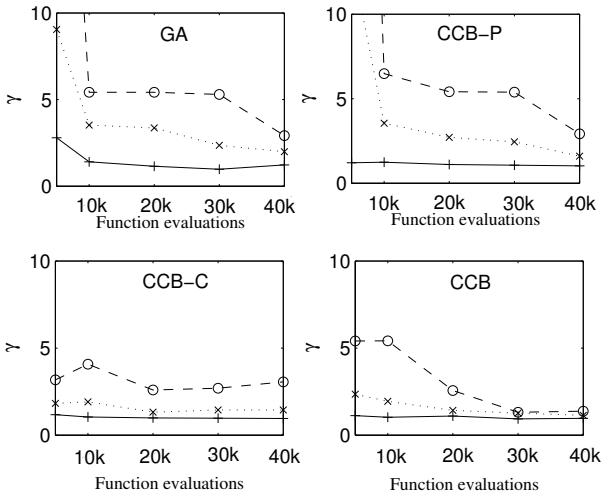


Fig. 5. The variation of $\gamma_{ave}$, $\gamma_{best}$ and $\gamma_{worst}$ against $N_{max}$ ($k$ : $kilo = 1000$)

To better understand the above results let us observe that when using GA for solving optimization problems then the minimum number of iterations required to converge to acceptable solutions, increases as the number of decision variable increases. Let us assume now that for a given problem GA requires 100 iteration with a population size of 20 to converge to acceptable solutions. Now, if more than one population is used, for instance 4 populations, then to keep $N_{max} = 100 \times 20$ fixed the number of iterations must be reduced to 25, but this is to small for GA to work at all. This explains why the use of parallelism does not improve the convergence properties of GA, in other words the reason for that is the limited number of function evaluations (5k-40k) and the complexity of the search space.

The remaining question now is why parallelism helps when combined with cooperation even with a small $N_{max}$ to clarify this point we need to consider problem solved in this section. Let $N_{max} = 30k$ and $n_p = 30$ which gives $N_{itr} = 1000$ if pure GA is used, but if CCB-P is used with the same $n_p$ and 5 populations then each population is allowed to evolve for only 200 iterations which is apparently not enough to search for 22-decision variable $(\theta, \beta)$. Now, when solving the problem in cooperative manner there will be 10 populations (5 sub-populations for controllers $\theta$'s, and 5 sub-populations for multipliers $\beta$'s), in this case $N_{itr} = 100$, but the search space is much smaller (i.e each subpopulation searches for either 12-controller variables or 10-multipliers variables ), consequently, this small number of iterations $N_{itr} = 100$ is sufficient to show *the power of parallelism*. Interestingly, if the maximum number of function evaluations is further decreased (say 5k), then parallelism will not work in either cases (cooperative or noncooperative), this fact is clear in Figure 5, and can be further clarified with the help of Figure 6. This figure shows the variation of SD against $N_{max}$, and confirms that the CCB algorithm has the best consistency (i.e smallest $\sigma_\gamma$) for $N_{max} = 10k - 40k$, but with $N_{max} = 5k$ CCB-C outperform CCB, which means that parallelism has a negative effect if $N_{max}$ is not sufficiently large.

There are several parameters influencing the performance of parallel GA, in particular *migration rate, number of migrants, and the migration strategy (or scheme)*. Suitable values for these parameters can be estimated by performing a sufficient number of experiments. Four possible migration strategies (i.e. for transferring and receiving migrants) were considered here, *random → random, random → worst, best → random and best → worst*. For example, random → random scheme means that each population export a *random* set of migrants, on the other hand *a random* set of individuals are removed from it to receive new migrants. It has been found that random → random scheme achieves the best $SD$, moreover, based on this experimental study a hybrid scheme (proposed here) was found to perform the best. This scheme uses *best → random* at early generations, then switches to *best → worst* scheme at later generations (e.g. after 50% of total number of iteration). The motivation for this hybrid scheme is based on the observation that at earlier generations removing the worst individuals may lead to premature convergency towards local minima. On the other hand at later generations all populations are already converged to some solutions so is its more important at this stage to share best solutions between these populations. Figure 7 shows the performance of some tested schemes including the hybrid scheme, this figure confirms the superiority of the proposed scheme over other schemes, all results developed here are based on this hybrid-scheme.

Considering other migration parameters, it was found that a suitable migration interval may be chosen as $0.05 - 0.15 \times N_{iter}$, a suitable number of migrants (transmitted) can be selected as $N_p/(z_m - 1)/2$ which implies that each subpopulation receives $N_p/2$ migrants (i.e. complete net topology). For example if $N_p = 20$ with three populations, then each population receives ten migrants (5 migrants from each other population).

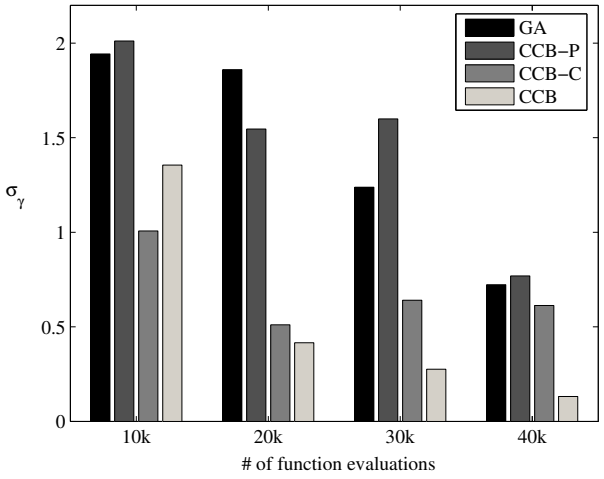


Fig. 6. SD versus $N_{max}$ for GA and CCB.

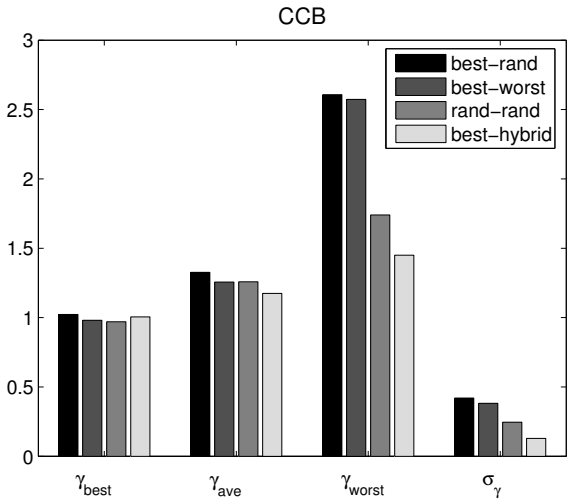


Fig. 7. The influence of migration strategy on $\sigma_\gamma$.

## 5. CONCLUSION

This paper proposes a novel algorithm named as a Co-evolutionary Cooperative Bisection (CCB) algorithm as master tool for solving variety of control problems. The proposed algorithm is computationally attractive and has very favorable convergence properties. The numerical superiority of the CCB algorithm is back to two main factors, the numerical efficiency of $\gamma$-Bisection algorithm, and the multi-population structure proposed in this paper (*Cooperation* and *Parallelism*).

To demonstrate the potential of the proposed algorithm a method for designing fixed-structure controllers that minimize an upper bound on the singular value $\mu$ has been developed. The problem considered here is of significant practical interest, since many engineering control problems involve hard constraints on the controller order or structure. Moreover, even if the controller order or structure are not restricted, the proposed algorithm has the advantage of being able to optimize over the multiplier $D(s)$ and controllers $K(s)$ simultaneously, thereby improving the chances of converging to the global minimum. Numerical

examples given here confirm the computational efficiency and the excellent convergence properties of the CCB-algorithm.

## REFERENCES

W.F. Arnold and J. ALaub. Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations. In *Proc. IEEE*, pages 1746–1754, 1984.

G.J. Balas and J.C. Doyle. Robust Control of Flexible Modes in the Controller Crossover Region. *AIAA Journal of Guidance, Dynamics and Control*, 17(2):370–377, 1994.

G.J. Balas, J.C. Doyle, K. Glover, A. Pack, and R. Smith. *$\mu$-Analysis and Synthesis Toolbox*. The Math Works, 1993.

D. Banjerdpongchai and J. P. How. Parametric robust $H_2$ control design using iterative linear matrix inequalities synthesis. *AIAA Journal of Guidance Control and Dynamics*, 23(1):138–142, 2000.

S. Boyd, V. Balakrishnan, and P Kabamba. A Bisection Method for Computing the $h_\infty$ Norm of Transfer Matrix and Related Problems. *IEEE Math Control Signal Systems*, 2:207–219, 1989.

S.P. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in Systems and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, USA, 1994.

A. Bucci and J. Pollack. on Identifying Global Optima In Cooperative Coevolution. In *the 2005 Genetic and Evolutionary Computation Conference*, pages 539–544, 2005.

E Cantu-Paz. A Survey of Parallel Genetic Algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10 (2):141–171, 1992.

E Cantu-Paz. Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms. *Journal of Heuristics*, 7(4):311–334, 2001.

M. Chilali and P. Gahinet. $H_\infty$-design with pole placement constraints: An LMI approach. *IEEE Trans. Automatic Control*, 41(3):358–367, 1996.

A. Farag and H. Werner. A Ricatti - Genetic Algorithms Approach to Fixed-Structure Controller Synthesis. In *Proc. American Control Conference ACC*, 2004.

A. Farag and H. Werner. Decentralized Control of Winding Systems: a Hybrid Evolutionary-Algebraic Approach. In *Proc. 16th IFACWorld Congress*, Prague, 2005a.

A. Farag and H. Werner. Fixed-Order Control of Active Suspension : a Hybrid Evolutionary-Algebraic Approach. In *Proc. 16th IFACWorld Congress*, Prague, 2005b.

C.M. Fonseca and P.J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

J.P. How, S.R. Hall, and W.M. Haddad. Robust controllers for the middeck active control experiment using Popov controller synthesis. *IEEE Trans. Control Systems Technology*, 2(2):73–86, 1994.

H. Kajiwara, Pierre Apkarian, and Pascal Gahinet. LPV Techniques for Control of an Inverted Pendulum. *IEEE Control Systems Magazine*, 19:44–54, 1999.

D. R. Lewin and A. Parag. A constrained genetic algorithm for decentralized control system structure selection and optimization. *Automatica*, 39:1801–1807, 2003.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 1996.

Angel Kuri Morales and Carlos Villegas Quezada. Univeral Eclectic Genetic Algorithm for Constrained Optimization. In *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing*, pages 518–522, 1998.

A. Packard and J. Doyle. Robust control with an $H_2$ performance objective. In *Proc. American Control Conference*, pages 2141–2146, 1987.

L. Panait, R.P. Wiegand, and S. Luke. Improving Coevolutionary Search for Optimal Multiagent Behaviors. In *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 653–658, 2003.

L. Panait, S. Luke, and R. P. Wiegand. Biasing Coevolutionary Search for Optimal Multiagent Behaviors. *IEEE Transactions on Evolutionary Computation*, 2006.

Liviu Panait. *The Analysis and Design of Concurrent Learning Algorithms for Cooperative Multiagent Systems*. PhD thesis, George Mason University, Fairfax, Virginia, 2006.

Liviu Panait, R. Paul Wiegand, and Sean Luke. A Visual Demonstration of Convergence Properties of Cooperative Coevolution. In *Schwefel*, pages 892–901, 2004.

Elena Popovici and Kenneth De Jong. Understanding Cooperative Co-evolutionary Dynamics via Simple Fitness Landscapes. In *Genetic and Evolutionary Computation Conference GECCO 2005*, pages 507–514, 2005.

M. Potter. *The Design and Analysis of a Computational Model of Cooperative CoEvolution*. PhD thesis, George Mason University, Fairfax, Virginia, 1997.

Mitchell A. Potter and Kenneth A. De Jong. A Cooperative Coevolutionary Approach to Function Optimization. In *In Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pages 249–257, 1994.

R. P. Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 2003.

R. P. Wiegand and M. A. Potter. Robustness in Cooperative Coevolution. In *Genetic and Evolutionary Computation Conference GECCO 2006*, pages 369–376, 2006.

R. P. Wiegand, W. Liles, and K. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Genetic and Evolutionary Computation Conference GECCO 2001*, pages 1235–1242, 2001.

R. P. Wiegand, W. Liles, and K. De Jong. Analyzing Cooperative Coevolution with Evolutionary Game Theory. In *In Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1600–1605, 2002a.

R. P. Wiegand, W. Liles, and K. De Jong. Modeling Variation in Cooperative Coevolution Using Evolutionary Game Theory. In *Foundations of Genetic Algorithms*, pages 231–248, 2002b.

K. Zhou, C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, USA, 1996.