CML 2001 Report:

Multi-Objective Genetic Algorithm and Its Application in Optimizing Parameters of Fixed-Structure Controller for HDDs

Bo Zhu and Masayoshi Tomizuka
Computer Mechanics Laboratory/ Mechanical Systems Control Laboratory
Department of Mechanical Engineering
University of California at Berkeley

Ho Lee and Lin Guo
Advanced Technology Group
Maxtor Corporation

September 2001

Abstract

This report presents a non-gradient based optimization method, named multiobjective genetic algorithm (MOGA), for the parameter optimization of fixed-structure controllers in hard disk drives (HDDs). The structure of track following controller is fixed to address costs and implementation issues in commercial HDDs. On the down side, the rank constraint destroys the convexity of search space. In addition to multiple design objectives and constraints, it makes the tuning of controller parameters a difficult non-convex multiobjective optimization problem (NCMOP), for which gradient based methods are likely to be trapped in the local optima. Genetic algorithms (GAs) are stochastic rule based global search method, simulating the nature evolution process. GAs carry out the parallel search by maintaining a population of candidate solutions and updating them through genetic operators. MOGA is a GA embedded with the concept of Pareto optimality and capable of solving large-scale NCMOP efficiently and reliably. A design example of tuning a track following controller is used to demonstrate the effectiveness of the proposed method.

Contents

1 Introduction

Given cost-limited hardware and computing resources, designing a positioning system for mass-produced hard disk drives (HDDs) is a challenging work. The controller needs to achieve acceptable performance in the presence of external disturbances and measurement noises. Furthermore, the plant parameters vary from drive to drive due to manufacturing tolerances and also change with time and temperature. The controller therefore needs to provide sufficient robustness against these variations. On the other hand, the usage of economical digital signal processor in servo system requires a low order design of controller. This is a typical optimization problem with multiple non-commensurable objectives and constraints. Recently, linear matrix inequalities(LMIs) and bilinear matrix inequality (BMIs) have received a great deal of attention in parameterizing multi-objective optimization problems (MOPs) [1] [2], although general design method guaranteeing the global optimization has not been obtained. LMIs or BMIs provide a unifying framework to setup multiple *quadratic* performance inequalities, which are then efficiently solved by convex optimization approaches, e.g. semi-definite programming (SDP). It has been shown that a number of multi-objective problems, including the popular H_∞ control synthesis problem and mixed H_2/H_∞ optimal control problem, can be parameterized as LMIs or BMIs [3]. However, for the practical control of systems like HDD, it is rather difficult to fit the problem into a solvable convex form because of numerous design specifications and constraints. Furthermore, an optimal solution from the free-order synthesis is a high order controller, which is not implementable in HDD even with order reduction techniques. In fact, it is more efficient to start with a controller structure that has good nominal properties, and optimize the parameters within that structure. Fixing the structure of controller minimizes the coding difficulties involved in design iterations and enhances the controllability of overall firmware. On the other hand, since it also inevitably loses the convexity of the search space, gradient-based methods [4] will have no guarantee of global convergence for such a nonconvex multiobjective optimization problem (NCMOP) and require lots of auxiliary conditions for parameterizing NCMOP, if possible.

This report proposes a non-gradient based solution to the design problem above by using a multi-objective genetic algorithm (MOGA), which is the combinations of genetic algorithms (GAs) and Pareto optimization. GAs are non-gradient based optimization methods that imitate the stochastic mechanisms of natural selection and genetic variation. Pareto optimality, which defines the optima based on vectorial performance, avoids the hassle of using weights before optimization and provides decision maker with multiple "trade-off" optimal solutions which reveal the characteristics of solution surface before a final solution is chosen. Finally, the effectiveness of the proposed method is shown by an example of tuning a track following controller for HDD.

This report is organized as following. Section 2 shows two ways of setting up optimization problem for the fixed-structure track following controller, along with a brief review of different optimization methods. Section 3 discusses a general GA in details and Section 3 extends it to MOGA. Section 4 shows a design example of tuning a track following controller. A summary is given in 5.1.

2 Problem Formulation

This report focuses on the tuning of fixed-structure track following controller. Figure 1 shows a discrete time model of HDD servo system in the track following mode. $P\left(z^{-1}\right)$ is the ZOH equivalence of continuous time plant dynamics, including power amplifier (PA), voice coil motor (VCM), and head-disk assembly (HDA). For convenience, disturbances and noises are lumped into three sources depending on their injecting points in the loop: torque disturbance d_t, position disturbance d_p, and measurement noise n_m. $C(z^{-1}, K)$ represents the fixed-structure low-order digital controller with m tunable parameters $K=\{k_1, ..., k_m\} \in \Omega_K \subset R^m$ where Ω_K is the non-convex search space with boundaries defined by designer. The following notation and terminology are used

$$\begin{array}{ll} S\left(z^{-1}, K\right) (= \frac{1}{1+P(z^{-1})C(z^{-1},K)}, & \text{sensitivity function;} \\ T\left(z^{-1}, K\right) = \frac{P\left(z^{-1}\right)C\left(z^{-1},K\right)}{1+P(z^{-1})C(z^{-1},K)}, & \text{complementary sensitivity function.} \end{array}$$

There are two frameworks that can be used to set up the optimization problem. No matter which framework readers choose to use, there are one fundamental issue they have to address: how should the global optimum be achieved efficiently and reliably with so many objectives and constraints.

2.1 Classical Control

The classical frequency domain loop-shaping techniques have been successfully used by servo engineers for decades, to the analysis and design of HDD servo systems. Setting up the optimization problem in this framework makes it ready for engineers to adopt proposed method into their practical applications. In track following mode, the major design objective is to minimize the *true* tracking error PE_t, which is, however, immeasurable. The position error signal (PES) being measured is the true position error PE_t contaminated by the measurement noise n_m. Fortunately, n_m is shown to be white [5] thus minimizing the energy of PES is equivalent to minimizing the energy of PE_t. Typical time domain specifications on the PES are $|PES(t)| \leqslant 12\%$ trackwidth and $3\sigma(PES) \leqslant 5\%$ trackwidth. The design also needs to satisfy some minimum relative stability margins. In this report, all constraints are related to the frequency domain requirements, such as the minimum phase margin PM_{MIN} which is a direct safeguard against time delay uncertainty, the minimum gain margin GM_{MIN} which is a direct safeguard against steady-state gain uncertainty, the minimum crossover frequency ω_{oMIN}, and the maximum peak of sensitivity function $S_{\infty MAX}$.

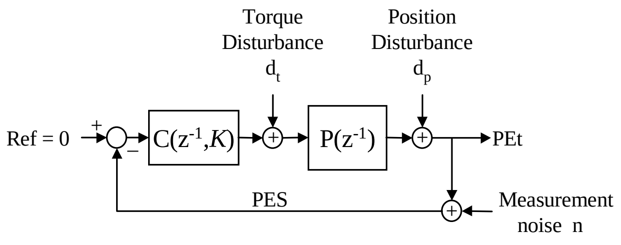


Figure 1: A typical blockdiagram of track following control

This is a constrained NCMOP. A design example is given in Section 5

2.2 Mixed H_2/H_∞ Control

Designing controller for track following is multi-objective output-feedback control problem involving noise rejection (H_2) and loop-shaping (H_∞). Since all disturbance, including d_t, nonrepeatable part of d_p, and n_m, can be modeled as white noises [6], the time domain regulation design (minimizing PES) can be formulated as H_2 control problem. Furthermore, Skogestad et al. [7] show the gain margin and phase margin are closely related to the peak values of $|S(j\omega)|$ and $|T(j\omega)|$. Thus we can use $\|S\|_\infty$ and $\|T\|_\infty$ instead of PM and GM as stability constraints. So all performance requirements in frequency domain can be formulated as H_∞ loop shaping problem. If we drop off the non-convex rank constraint due to fixing the structure (and thus order) of controller, the overall problem becomes a full-order H_2/H_∞ control synthesis problem and can be parameterized in the form of LMIs [8] which are solvable by widely available convex solvers like SDP. When the rank constraint is applied, it is a fixed-order (or more generally, reduced-order) control synthesis problem that is nonconvex and hard to solve. On one side, one can apply linearization or approximation algorithms to make it solvable by convex solvers [9] [10], but still there is no guarantee of finding a globally optimal solution. Also, the exactness of the LMI formulation is at the cost of introducing a large number of auxiliary variables which produces an order of magnitude increase in search space. This can be a problem for NCMOPs with a large number of tunable variables. On the other side, as suggested in this report, one can directly apply non-gradient based optimization methods, e.g., [11] shows a genetic method to solve LMIs.

2.3 Overview of Optimization Techniques

Among numerous search and optimization techniques that are potentially capable of solving non-convex problems, gradient-based, enumerative, and stochastic techniques are mostly used.

Gradient-based techniques use information about the slope of target function to dictate a direction of search where the optimum is thought to be lie. They can be classified into two groups: indirect and direct [12]. Indirect methods search local extreme by solving a set of usually nonlinear equations which result from setting the gradient of the objective functions equal to zero. On the other hand, direct methods seek local optima by jumping on the hyper-surface of target function and moving in a direction related to the local gradient. Both classes are local methods for they seek the optima only in a neighborhood of the current point. General speaking, gradient-based methods are more efficient than other methods in a local convex range. However, they require the continuity and derivative existence of target function and thus can not provide sufficient robustness to find the global optimum for non-convex optimization problems.

The idea behind enumerative schemes is pretty simple; within a finite search space, or a discretized infinite search space, the search algorithm evaluates objective function values at every point in the space, one at a time. This scheme can arrive at reasonably good solutions for search spaces of small sizes. But when confronted with search spaces of enormous size and wide variation from point to point in their precinct, such schemes must ultimately be discounted in the robustness race for one simple reason: lack of efficiency. Even the highly touted enumerative scheme dynamic programming may become exhausted on problems of moderate size and complexity, suffering from "the curse of dimensionality" [13].

Stochastic optimization algorithms have been recognized to be able to overcome the shortcomings of gradient-based and enumerative schemes. The common feature of these methods is that they only need evaluations of the objective function, not requiring its gradient or Hessian. So they are most suitable for problems that are very nonlinear or have a number of discontinuities. Random walks is the simplest version of stochastic optimization algorithms. Although it is capable of searching for the global optima, it is still lack of efficiency and expected to do no better than enumerative schemes. Some more advanced random methods like Genetic algorithms (GAs) and simulated annealing carry out initial search randomly, but the information gained from the search is utilized in guiding the next search.

3 General Genetic Algorithms

GAs are biologically inspired global searching methods, described in one sentence as "an individual with greater vitality has a better chance to survive in this highly competitive world." Since the initial idea is brought out by Holland [14] in 1975, GAs have attracted a great interests and quickly become a flagship among machine learning and function optimization. It is beyond the scope of this paper to review the vast literature associated with GAs, instead, this section addresses the basic ideas of GAs. The interested reader might consult Goldberg [12] and Michalewicz [15].

GAs overcome the aforementioned limitations of conventional searching algorithms through following aspects [15] [16].

- GAs use probabilistic transition rules to guide their search, but not deterministic rules.

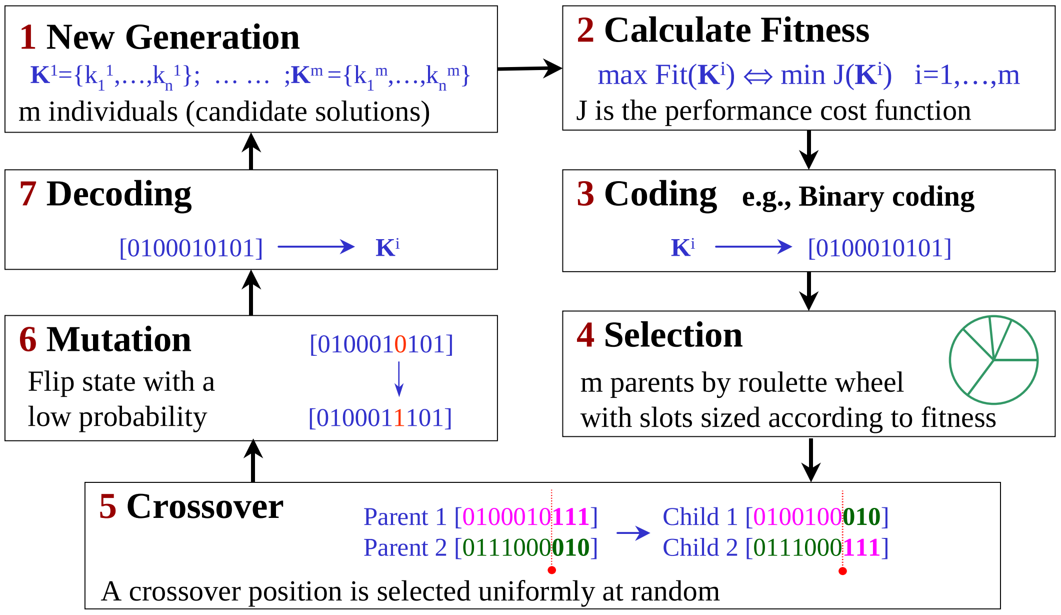

1 New Generation
K¹={k₁¹,…,kₙ¹}; … … ;Kᵐ={k₁ᵐ,…,kₙᵐ}
m individuals (candidate solutions)
2 Calculate Fitness
max Fit(Kⁱ) ⇔ min J(Kⁱ) i=1,…,m
J is the performance cost function
3 Coding e.g., Binary coding
Kⁱ → [0100010101]
4 Selection
m parents by roulette wheel
with slots sized according to fitness
5 Crossover
Parent 1 [0100010111]
Parent 2 [0111000010]
Child 1 [0100100010]
Child 2 [0111000111]
A crossover position is selected uniformly at random
6 Mutation
Flip state with a
low probability
[0100010101]
[0100011101]
7 Decoding
[0100010101] → Kⁱ

Figure 2: The flowchart of a simple genetic algorithm for parameter optimization. It is easier to exemplify the genetic operators by using the binary coding.

- GAs use objective function information, but not derivative or other auxiliary data.
- GAs maintain a population of potential solutions to search in a parallel manner, while all other methods process a single point of the search space. This is referred to as "Implicit Parallelism".

The basic building blocks of GAs are shown in Figure 2. GAs start with a population of randomly generated solutions. Each solution in this very first generation will be evaluated through cost functions that are defined according to design objectives. During the evaluation, we define a fitness function and evaluate the fitness for each candidate solution. The fitness is assigned in a way that maximizing the fitness is equivalent to minimizing the cost function. Then the population is processed and evaluated through various operators to generate a new population. The basic operators of GAs are *selection*, *crossover*, and *mutation,* as shown in step 4, 5, and 6 of Figure 2 respectively. This process is repeated until a global optimal point is reached. The following sections examine this process in details.

3.1 Coding: From Binary to Real

GAs work with a coding of parameters, but not parameters themselves. The main purpose of coding is to make it possible to imitate the natural evolution process in parameters space. The traditional way to illustrate GAs is to use *binary coding* (BC) . As shown in Figure 2-(3), each parameter is encoded into a string of bits. The individual bit is called a *gene*. The content of each gene is called an *allele*. The whole string of such genes for all parameters in a written sequence is called a *chromosome*. The choice of chromosome length depends on the accuracy requirements of the targeting optimization problem. GAs maintain a population of chromosomes or individuals in every generation, quite often these individuals are also called *candidate solutions*. The number of chromosomes in a population is called the population size, denoted as n_{pop}. These chromosomes will evolve from generation to generation through some genetic operators.

However, There are some drawbacks when applying BC to multidimensional, high-precision numerical problems. For example, for 80 variables with domains in the range [-100 100] where a precision of six digits after the decimal point is required, the minimum length of the binary chromosome is $27 \times 80 = 2160$. For such a problem GAs perform poorly because the BC generates a search space of about 2^{2160}.

Instead of using BC, this report uses real-coded (RC) encoding scheme, i.e. each candidate solution is represented as a float-point vector $K^j = [k_1^j, ..., k_m^j]$. At $(t)th$ generation, MOGA maintains a population of individuals, $\mathbf{K}(t) = [K^1(t), ..., K^j(t), ..., K^{n_{pop}}(t)]$,with a fixed population size n_{pop}. This real-coded (RC) representation has many advantages over the classical binary-coded (BC) representation which uses a bit string to code candidate solutions [15].

- The RC is faster, more consistent from run to run and provides a higher precision than BC,The precision of RC depends on the underlying machine, hence is generally much better than that of the BC. Although we can always enhance the precision of the RC by introducing more bits, this considerably slows down the algorithm.
- The RC is capable of representing quite large domains. On the other hand, the BC must sacrifice precision with an increase in domain size for a given fixed binary length. Thus the RC avoids Hamming Cliffs effect from which the BC suffers.
- The RC is conceptually closest to the problem space, so virtually no decoding is required. The RC also allows for an easier implementation of genetic operators.

3.2 Initial population

A simple GA is an iterative procedure starting with a randomly generated population of candidate solutions. Since one has little geometric knowledge of the search space of target problem before solving it, the uniform distribution is arguably the best initialization scheme, keeping in mind that the population size n_{pop} is the only factor in this method to balance the trade-off between initial diversity and computational complexity. If n_{pop} is too small, the chance that the chromosomes in the population cover the entire search space is low. This makes it difficult to obtain the global optimum solution and leads to a local optimum as a result of premature convergence. On the other hand, a population size that is too large decreases the rate of convergence. In the worst case scenario, it may lead to divergence. Hence the population size needs to be selected based on the size of search space. An example at the end of this section will clearly illustrate this relationship.

Individuals in the first generation are evaluated and assigned fitness values based on their relative performance. Through applying genetic operators, the initial population will evolve into a new population which contains a group of better solutions.

3.3 Cost Function and Constraint Handling

Cost functions are directly related to our design objectives and constraints. If we are to solve a minimizing problem, the objective function itself can be used as cost function because GAs are aiming at minimizing cost functions. On the other hand, to define the cost function for maximization problems, one usually just takes the reciprocal or flips the sign of objective functions. Since GA is a population-based probabilistic optimization method and does not require the convexity of search space, the cost function $J(K)$ can be freely chosen for the convenience of designer. Although the optimization problem of fixed-structure controller can be formulated as solving LMIs, limiting the objective functions in quadratic form does not fully explore the potential of GAs. Some practical design problem is very difficult or even impossible to be parametrized in standard frameworks.

Constraints can often be seen as hard objectives, which need to be satisfied before the optimization of the remaining objectives takes place[17]. Constraints are usually expressed in the following type of inequality

$$f(K) \leqslant g \tag{1}$$

where $f(K)$ is a real-valued function of variable set K, and g is a constant value. The inequality may also be strict ($<$ instead of $\leqslant$). Equality constraints of the type $f(x) = g$ can be formulated as particular case of inequality constraints. The simplest approach to handing constraints has been to assign infeasible individuals an arbitrarily high cost (or low fitness). Certain types of constraints, such as bounds on the search space, can be handled by mapping the search space so as to minimize the number of infeasible solutions it contains and designing the genetic operators carefully in order to minimize the production of infeasible offspring from feasible parents [18]. In the case where no feasible individuals are known or cannot easily be found, but evaluation of individuals are relatively easy, a penalty function is used to impose penalty onto infeasible individuals depending on the extent to which they violate the constraints. Then cost function for the constraint (1) is simply

$$J(K) = Q[f(K) - g] \tag{2}$$

where the penalty function

$$Q[a] = \begin{cases} a & a > 0 \\ 0 & a \leqslant 0 \end{cases} . \tag{3}$$

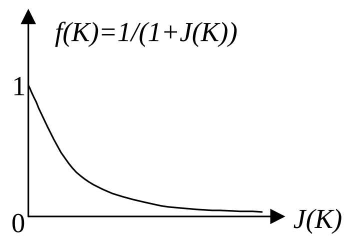


Figure 3: An example of fitness assignment for a minimizing problem with positive semidefinite objective function.

For example, the cost function $J_{S_\infty}(K) = Q[S_\infty(K) - 6]$ is to enforce the constraint of making the peak value of sensitivity function less than $6dB$.

3.4 Fitness

After the function evaluation, a real positive number called fitness is assigned to each candidate solution in the way that maximizing the fitness is equivalent to minimizing the cost function, i.e.,

$$max[f(K)] \Longleftrightarrow min[J(K)] \Longleftrightarrow \text{achieve objective and/or satisfy constraint.} \qquad (4)$$

The fitness serves as the surviving probability of each individual during genetic operations as described in the next section. An individual with a bigger fitness value implies higher quality with respect to design objectives, and a better chance to survive in genetic operations. The programmer is allowed to use any fitness function that adheres to the relationship in (4). This advantage over other optimization methods makes GAs more attractive when dealing with practical engineering optimization problems. (5) and Figure 3 show an example of defining fitness for a minimizing problem with positive semidefinite objective function. $f(K)$ gets the biggest value of 1 (for a possibility may not be greater than 1) when minimum $J(K) = 0$, and converges to zero as $J(K)$ increases.

$$f(K) = \frac{1}{1 + J(K)} \qquad (5)$$

3.5 Genetic Operators: Selection, Crossover, and Mutation

The new generation is generated from the current one by examining the fitness of all candidate solutions and applying the genetic operators, which are *selection*, *crossover*, and *mutation*. Through the genetic operations, the survival of the fittest means transferring the highly fit chromosomes to the next generation of chromosomes and combining different chromosomes to explore new search points.

Selection The roulette-wheel-like *selection* is more likely to pick up candidates with higher fitness values into a mating pool. Each candidate solution corresponds to a slice in the weighted roulette wheel (Figure 2-(4)). The relative size of the slice equals to the fitness percentage of corresponding solution in the total fitness. The total fitness is obtained by summing the fitness over all individuals in the current generation. Every spin of the weighted roulette wheel yields one reproduction candidate for parent pool. Since the time to stop is a uniform random variable, the possibility of any individual being picked up is proportional to its fitness. This spinning process needs to be repeated until enough parents are generated.

Crossover The selected solutions in parent pool are then processed by applying *crossover* which pairs up the parents and exchanges portion of their segments pairwisely. The step 5 of Figure 2 shows a single point crossover for BC chromosomes. A *crossover point* is selected randomly as an integer between 1 and the chromosome length. Here this crossover point is 7. Then children Child 1 and Child 2 are generated by exchanging the segments after 7 between Parent 1 and Parent 2.

Besides the crossover point, another parameter called *crossover probability* P_c, is to control the number of crossovers by acting as a decision variable before performing the crossover. A real number is uniformly generated in the range [0 1]. If this number is less than P_c, a crossover is performed, otherwise, Child 1 and Child 2 are simply direct copies of Parent 1 and Parent 2.

Mutation Nature uses large population sizes to keep her diversity. However, it is expensive to keep a very large population in GAs. Instead, *Mutation* performs a slight perturbation to the resulting solutions with a very low probability. As illustrated in the step 6 of Figure 2, the 7th gene is bit This is an effective way to preserve the diversity with a limited population size.

In this report, a special dynamic mutation operator is used aiming at both improving single-element tuning and reducing the disadvantage of random mutation in the RC implementation. We call it a *non-uniform mutation [15]*. If $K^j(t)= [k_1^j(t), ., k_i^j(t).., k_m^j(t)]$ is a chromosome in (t)th generation and the element k_i^j is selected for applying non-uniform mutation, the result is a vector $K^{j'}(t) = [k_1^j(t), ., k_i^{j'}(t).., k_m^j(t)]$, where

$$k_i^{j'} = \begin{cases} k_i^j + \Phi(t, UB - k_i^j) & \text{if a random digit is 0} \\ k_i^j - \Phi(t, k_i^j - LB) & \text{if a random digit is 1} \end{cases}$$

and LB and UB are lower and upper domain bounds of the variable k_i^j. The function $\Phi(t, z)$ is chosen such that it returns a value in the range $[0, z]$ and the probability of $\Phi(t, z)$ being close to 0 increases as t increases. This property causes this operator to search the space uniformly initially and very locally at later stages; thus increasing the probability of generating the new point closer to the original one rather than a random choice. According to this requirement, the following function is used in this report:

$$\Phi(t,z) = z \cdot \left(1 - r^{(1-\frac{t}{n_{gen}})^b}\right)$$

where $r \in [0,1]$ is a random number, n_{gen} is the maximal generation number, and b is a system parameter determining the degree of dependency on iteration number. As shown in Figure 4 and Figure 5, the bigger b is, the faster the probability density of $\Phi(t,z)$ become concentrated on zero with the increase of t.

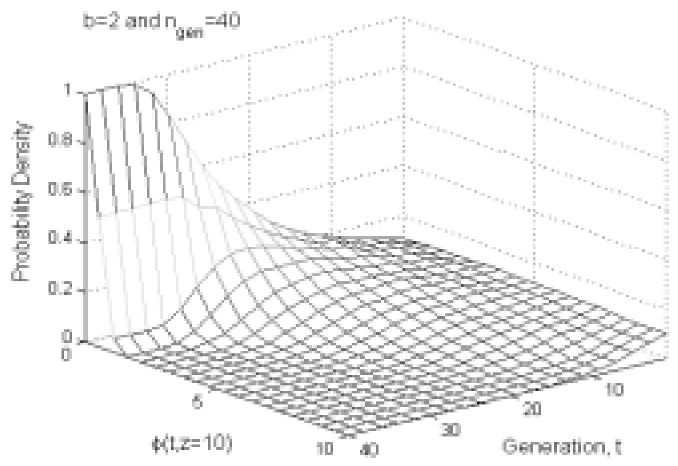


Figure 4: The probability density of $\phi(x,z)$ with respect to evolving generation t when b=2.

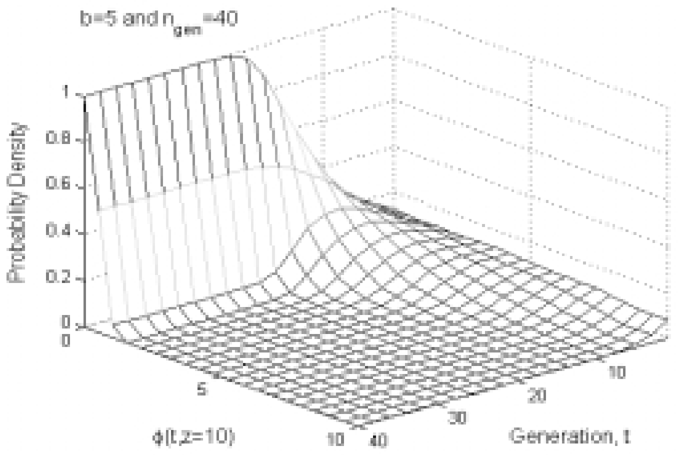


Figure 5: The probability density of $\phi(x,z)$ with respect to evolving generation t when b=5.

Table 1: The computational cost and solution quality v.s. the number of generations and population size

n_{gen}	5	10	15	30	50	100
n_{pop}	600	300	200	100	60	30
CPU time (s)	50.86	25.16	18.46	12.53	10.77	10
$max(x_{i\_opt})$	0.15	0.0015	0.001	0.016	0.00027	0.00047

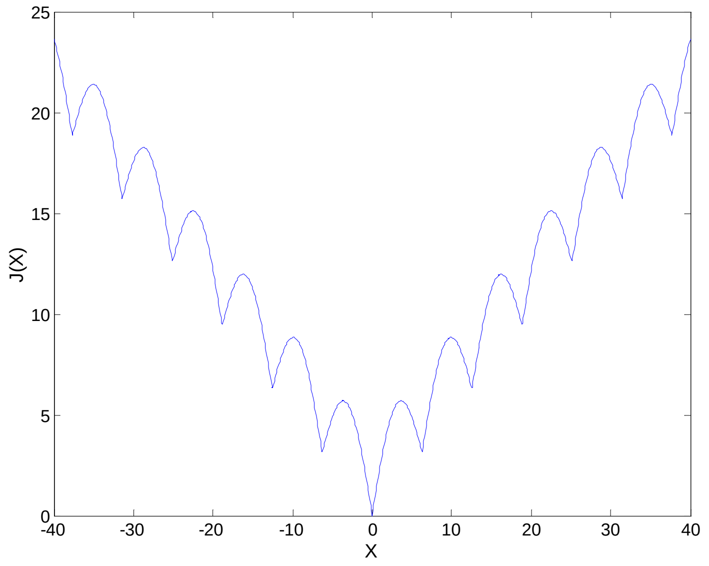


Figure 6: The illustration of the non-convex optimization example in (6) when n=1.

3.6 An Example of Optimization by GA

The following example (6) shows the effectiveness of GA in solving non-convex optimization problems. The global minimum of this example, $J(x_i) = 0$, is apparently achieved at origin $x_i = 0$, as shown in Figure 6 ($n = 1$). The gradient-based methods are likely to be trapped in numerous local minima at $x_i = 2k\pi, k = 0, 1, ...$.

$$\begin{aligned} \underset{x_i}{min}\,[J(x_i)] &= \underset{x_i}{min} \sum_{i=1}^{n} \left(\left| \frac{x_i}{2} \right| + 4 \left| \sin(\frac{x_i}{2}) \right| \right) \\ x_i &\in [-40, 40];\ n = 8 \end{aligned} \tag{6}$$

In De Jong's [19] study of genetic algorithms in function optimization, a series of parametric studies across a five function suite of problems suggested that good GA performance requires the choice of a moderate population size n_{pop}, a high crossover probability P_c, and a low mutation probability P_m which is inversely proportional to the population size n_{pop}.

To deliver the best performance of GA with a given computational power, the right choice of the number of generations and the population size is crucial. Figure 7 illustrates the interaction of

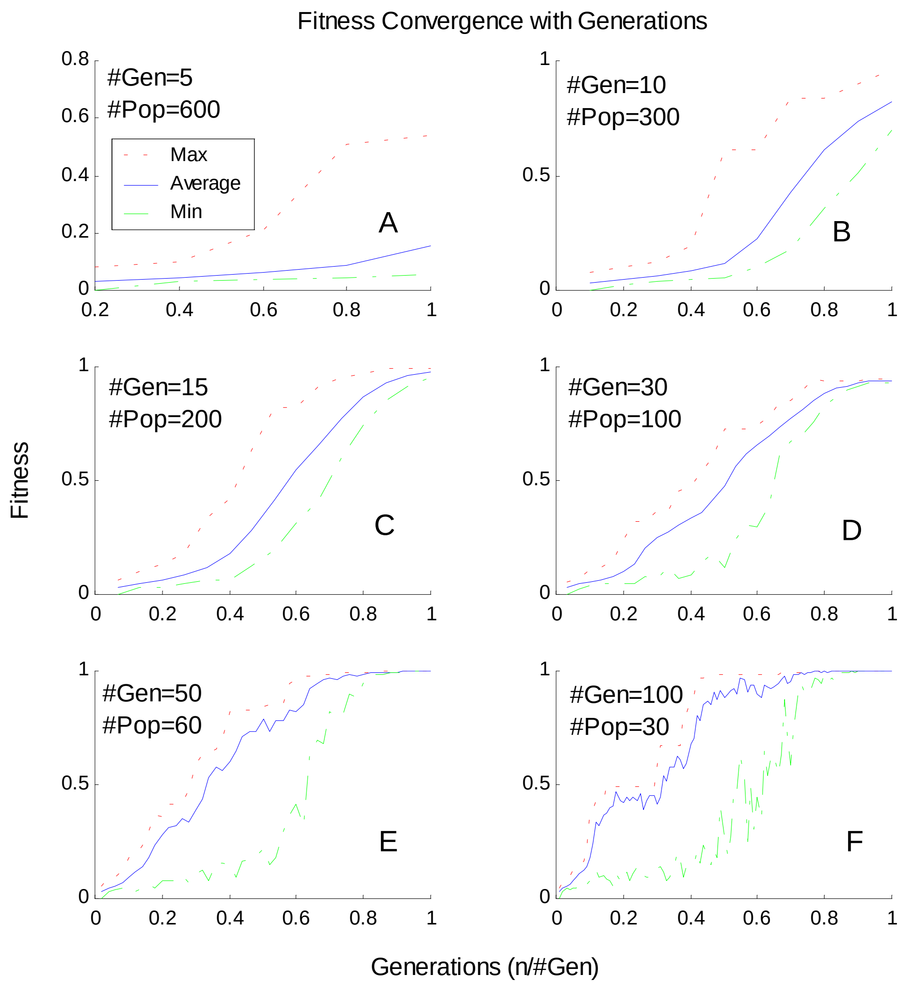


Figure 7: The right choice of the populaton size in one generation and the number of generations is very important in GAs. All trials have a total of 3000 samples.

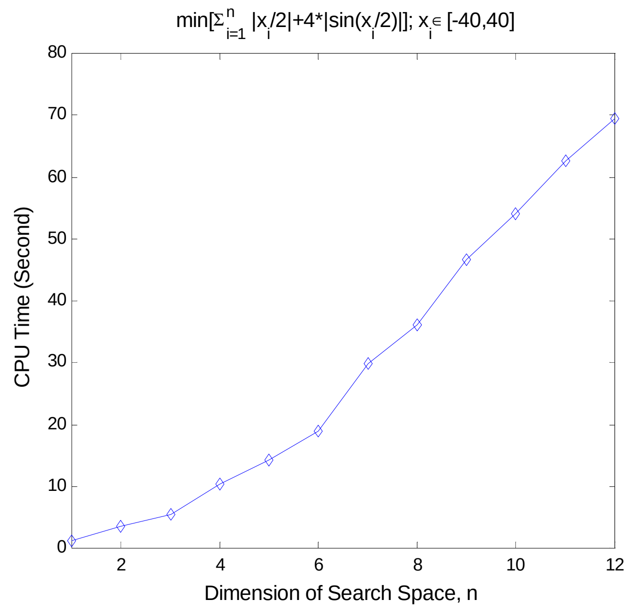


Figure 8: Upong applying GA to solve the given example, the relationship between the time to convergence ($|x_{i\_opt}| < 10^{-3}$) and the deminsion of search space is almost linear.

these two quantities on the fitness convergence and Table 1 shows their impact on the computational cost and solution quality. For a fair comparison, all six cases have the same total number of 3000 samples and the time to convergence for all cases is defined as when variables of the best solution converge to a hyper-ball centered at origin with a radius of 10^{-3}. We can see from Figure 7 that if the population size is too small, it may leads to the premature convergence problem because of the lack of diversity. Graphically it appears as a plateau in the average fitness curve before it converges to 1. A fitness of 1 generally corresponds to the global optimality (Figure 7-F). Other graphical evidences of the lack of diversity include the existing of very rough curves, and that the average fitness curve is much closer to the maximum fitness curve than to the minimum fitness curve. As the increasing of population size, however, the convergence speed is lowering and the computational cost is increasing. As shown in Table 1 and Figure 7-A, using too large population size not only wastes computational resource but also lowers the quality of solution. All simulation are performed on a AMD 1GHz PC with 256MB memory.

Figure 8 shows the relationship between the computational cost (the time for all Pareto solutions converge to a small hyper-ball around the origin) and the dimension of search space. Unlike the enumerative or LMI+approximation+SDP methods, the CPU time representing the overall computational complexity does not increase exponentially, but almost linearly relative to the increasing number of tunable variables. This is truly attractive for a NCMOP with large number of tunable parameters.

We would like to point out that there is no hard and bound restrictions on what operator and

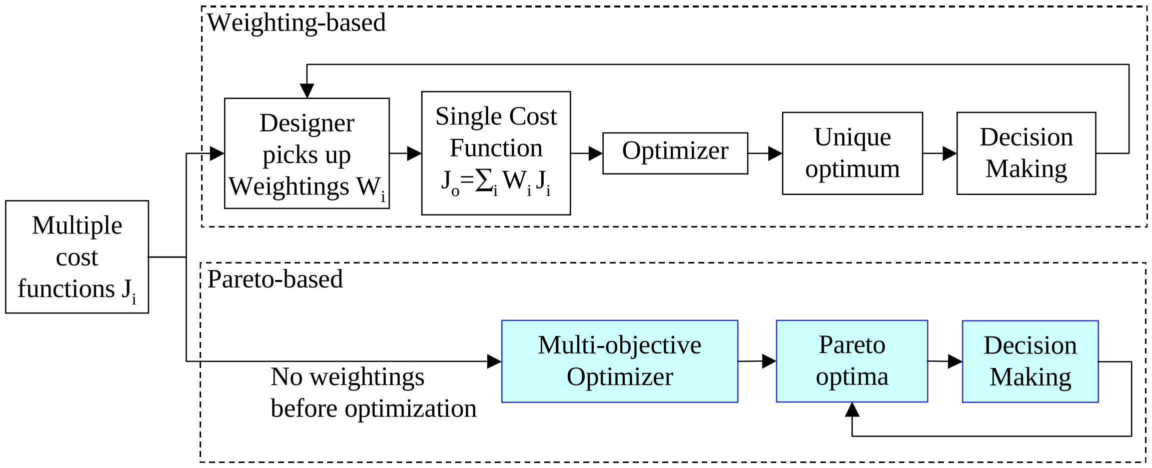

Weighting-based
Designer picks up Weightings W_i
Single Cost Function $J_o=\Sigma_i W_i J_i$
Optimizer
Unique optimum
Decision Making
Multiple cost functions J_i
Pareto-based
Multi-objective Optimizer
Pareto optima
Decision Making
No weightings before optimization

Figure 9: Different methods of dealing with multiobjective optimization problems. The upper part is the conventional weighting method and the lower part is the Pareto-based approach.

strategies a programmer has to use, so one has the freedom to choose the operation and strategies in any combination according to the characteristics of target problem.

4 Multi-Objective Genetic Algorithms

This section proposes a Multi-Objective Genetic Algorithm (MOGA), which extends the general GAs to handle multi-objective optimization problems, especially ones with non-commensurable objectives. Several multiobjective genetic algorithms (MOGAs), which are GAs combined with the concept of Pareto optimality, have been proposed [12] [20]. A good review of MOGAs is in [21]. Many researchers [22] [23] [11] have successfully applied MOGA to solve MOPs.

4.1 Multi-Objective Optimization (MOO)

Practical control design problems are often characterized by serval non-commensurable and often competing measures of performance, or so-called objectives. If any of objectives are competing, there is no unique solution. One way of dealing with multiple objectives is to combine them into a single objective by using weights (the upper part of Figure 9). In most cases, however, the best combination of weights is not known in advance due to the lack of knowledge of target problem. The whole iterative process involves many try-and-errors. One has to play with weights and repeat optimization for many times before getting insight into the interaction among objectives. This situation will get worse with increasing number of competing objectives.

Furthermore, the solution trade-off boundary may be nonconcurrent so that certain solutions are not accessible [24]. This can be illustrated geometrically by the following example with two objectives. In the objective function space a line, L, $wJ(K) = b$ is drawn. The minimization of b in

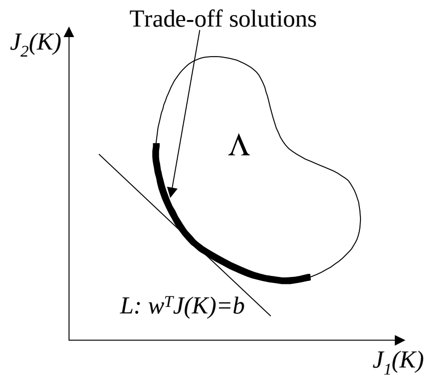


Figure 10: The convex trade-off boundary is accessible by varying weights.

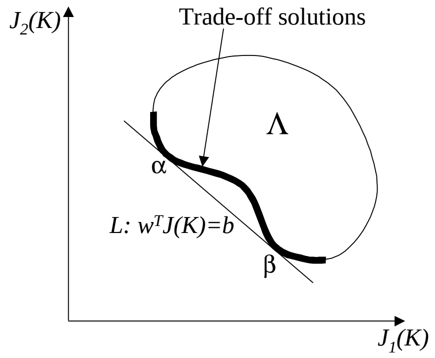


Figure 11: Some trade-off solutions on the nonconvex boundary are not accessible by varying w.

domain Λ can be interpreted as finding the value of b for which L just touches the boundary of Λ as it proceeds outwards from the origin. Selection of weights $w = [w_1, w_2]$, therefore, defines the slope of L, which in turn leads to the solution point where L touches the boundary of Λ. For a Λ with convex lower boundary as shown in Figure 10, all optimal solutions are reachable by varying w. However, accessibility cannot be guaranteed for Λ with a nonconvex lower boundary. For example, in Figure 11, all solutions between the point α and β are not accessible.

4.2 Pareto Optimality and Pareto Ranking

Pareto optimality, developed by Vilfredo Pareto (1848-1923, an Italian sociologist), is the most widely accepted criterion of economic efficiency. A state of a given system is Pareto optimal, and thus efficient, if and only if there is no feasible alternative state of that system in which at least one is better off and no one is worse off. And, for purposes of this criterion, a person is 'better off' with some alternative A rather than B if and only if this person *prefers* A to B [25]. The concept of Pareto optimality requires no weights *before* optimization and makes it possible to provide multiple optimal solutions to decision maker.

Definition 1 (Dominating Solution) *The vector* $\mathbf{J}(K^1)$=$\{J_1(K^1),...,J_n(K^1)\}$ *is said to* dominate *vector* $\mathbf{J}(K^2) = \{J_1(K^2),...,J_n(K^2)\}$ *if and only if* $\mathbf{J}(K^1)$ *is partially less than* $\mathbf{J}(K^2)$, *denoted as* $\mathbf{J}(K^1) <_p \mathbf{J}(K^2)$, *more precisely*

$$(\forall i)J_i(K^1) \leqslant J_i(K^2) \wedge (\exists i)J_i(K^1) < J_i(K^2). \tag{7}$$

Definition 2 (Pareto Optimality) *A solution* K^1 *is* Pareto optimal *if and only if there is no* $K^2 \in \Omega_K$ *such that* $\mathbf{J}(K^2) <_p \mathbf{J}(K^1)$. *Pareto optimal solutions* $\mathbf{K}_p$ *are also called* non-dominated set *or* non-inferior set, *which are a set of* K^j *such that*

$$\begin{aligned}(\mathbf{J}(K^i) &\nless {}_p\mathbf{J}(K^j)) \wedge (\mathbf{J}(K^j) \nless_p \mathbf{J}(K^i)), \\ K^i &\in \mathbf{K}_p, K^j \in \mathbf{K}_p, \forall i \neq j.\end{aligned} \tag{8}$$

In the other words, the Pareto set is optimal in the sense that no improvement can be achieved in any objective without degradation in others. An illustration example is shown in Figure 13, the solutions are evaluated based on three performance indexes respectively. For each index, the smaller the value, the better the solution. It is easy to see that solution A and B are better than all other solutions in all aspects. However, A performs better than B in terms of J_1 and J_2 but worse than B in term of J_3. According to the definition of Pareto optimality, we say A and B are the equally best solutions and in the same Pareto set. Similarly, C and D are equally good solutions in the second best set. Both dominate solution E but are dominated by A and B. The overall Pareto ranking with respect to $\mathbf{J} = \{J_1, J_2, J_3\}$ can be expressed as

$$\mathbf{J}\{A, B\} <_p \mathbf{J}\{C, D\} <_p \mathbf{J}\{E\} \tag{9}$$

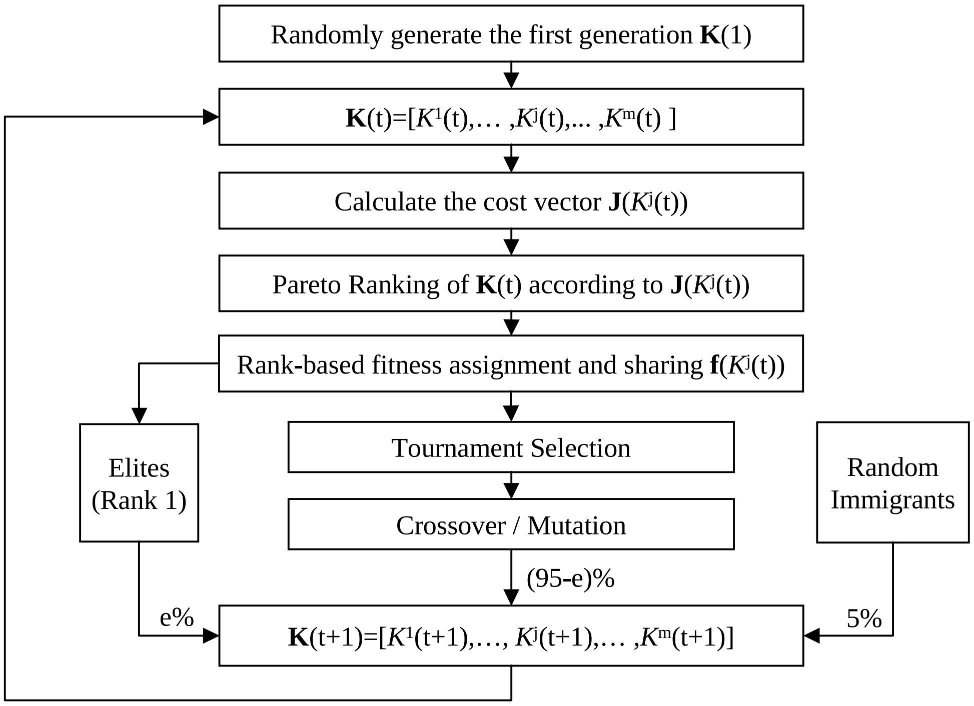
The flow chart of the MOGA used in this report is shown in Figure 12. The details are described below. A summary of this MOGA is given at the end of this section.

All K^j in the Pareto optimal set have the similar vectorial performance and thus are so-called "the equally best solutions" among the current generation $\mathbf{K} = [K^1, ..., K^{n_{pop}}]$. Therefore they are assigned the same rank of 1. The final solution of a MOP depends only on the vectorial performance and on the preferences of the decision maker, and not on any subsequent optimization [26]. Based on (7) and (8), a Pareto ranking scheme similar to [27] is proposed for MOGA as following,

1. Sort $\mathbf{K} = [K^1, ..., K^j, ..., K^{n_{pop}}]$ from the least to the largest according to $||\mathbf{J}(K^j)||_1 = \sum_{i=1}^{n} J_i(K^j)$. The sorted vector is denoted as $\mathbf{K}_s$. Let RNK=1.

2. Use the first entry of $\mathbf{K}_s$, K_s^1, as criterion. Take out any K_s^j from $\mathbf{K}_s$ and put it into a dominated vector Φ_d if $\mathbf{J}(K_s^1) <_p \mathbf{J}(K_s^j)$, $j = 2, ..., length(\mathbf{K}_s)$.

3. Move out K_s^1 from $\mathbf{K}_s$ and put it into a Pareto optimal set $\mathbf{K}_p$.



Randomly generate the first generation **K**(1)
K(t)=[*K*¹(t),… ,*K*ʲ(t),... ,*K*ᵐ(t)]
Calculate the cost vector **J**(*K*ʲ(t))
Pareto Ranking of **K**(t) according to **J**(*K*ʲ(t))
Rank-based fitness assignment and sharing **f**(*K*ʲ(t))
Elites (Rank 1)
Tournament Selection
Crossover / Mutation
Random Immigrants
(95-e)%
e%
5%
K(t+1)=[*K*¹(t+1),…, *K*ʲ(t+1),… ,*K*ᵐ(t+1)]

Figure 12: The flow chart of MOGA with elitism and random imigrants.

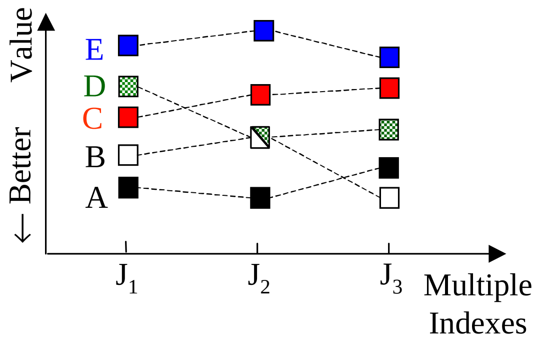


Figure 13:

4. Let the remaining candidate solutions in $\mathbf{K}_s$ to form a new $\mathbf{K}_s$, then repeat 2 to 4 until all dominated solutions are removed.

5. Assign all entries in $\mathbf{K}_p$ with the same rank, RNK. Empty $\mathbf{K}_p$.

6. Replace $\mathbf{K}_s$ with Φ_d, i.e. $\mathbf{K}_s = \Phi_d$.

7. $RNK = RNK + 1$ and repeat 2 to 7 until the entire population is ranked.

4.3 Rank-based Fitness Assignment

Every candidate solution K is assigned a fitness value $f(K)$ which is the measurement of solution quality. For a candidate solution in MOGA, the smaller the rank number, the better the vectorial performance. By selection, the MOGA is biased to the solution with higher fitness value. Therefore the fitness assignment is such a mapping that maximizing the fitness $f(K)$ is equivalent to minimizing the cost vector $J(K)$, i.e. maximizing the vectorial perform of K. In this paper, a simple exponential mapping is used

$$f^{'}(K) = \frac{1}{rank(K(t))} \tag{10}$$

4.4 Intra-Rank Fitness Sharing

Although all "equally good" solutions are assigned the same fitness, their actual choice to be selected as parents may differ due to the random nature of selection. This imbalance can be accumulated with the evolutions such that the population drifting towards an arbitrary region of the trade-off surface, a phenomenon known as *genetic drift* [28]. Various population diversity mechanisms have been proposed that make a GA to maintain a diverse population of individuals through its search. These mechanism allow GAs to identify multiple optima in a multimodal objective domain. *Intra-Rank Fitness sharing* is one of such mechanisms to counteract the effect of the genetic drift by re-distributing the fitness among the candidate solutions with the same rank [26]. The sharing function penalizes the fitness of individuals in popular neighborhoods and is in favor of more remote individuals. In this report, the sharing function is defined as

$$f(K^j) = \frac{D(K^j)}{\sum_j D(K^j)} \sum_j f^{'}(K^j) \tag{11}$$

where $D(K^j(t))$ is the *mutual similarity distance* which is the summation of similarity distance between $K^j(t)$ and any solution $K^i(t)$, $i \neq j$, with the same rank in the current generation. $f^{'}(K^j)$ is the original fitness value for K^j and $f(K^j)$ is the fitness after sharing. The similarity distance can be defined either in the objective space (phenotype) [29] or in the solution space (genotype) [30]. A genotype similarity distance is usually objective independent metric, such as the Hamming distance between two candidate solutions. It is used if there is little knowledge about the objective space. A phenotype metric is a more meaningful distance measurement quantifying the similarity of two solutions in the objective space.

4.5 Elitism and random migrants

To increase the converging rate of MOGA, the elite (individuals with the Rank 1) of current generation, which is $e\%$ of total population, are directly copied into the new generation. $(95 - e)\%$ individuals are generated from selection/crossover/mutation process. As a complementary mechanism of mutation, the remaining 5% are generated randomly to preserve the population diversity.

The MOGA used in this paper is summarized as follows.

1. Determine the search space of K, $\Omega_K \subset R^m$, which is the range of K that stabilizes the nominal plant $P_0(z^{-1})$ by Jury Criterion [31].

2. The MOGA randomly and uniformly generates the first generation with n_{pop} individuals (candidate solutions), $[K^1(1), ..., K^j(1), ..., K^{n_{pop}}(1)] \in \Omega_K$. Each individual is represented by RC.

3. For each individual $K^j(t)$ in the current $(t)th$ generation, calculate the cost function vector $\mathbf{J}(K^j(t)) = \{J_1(K^j(t)), ..., J_i(K^j(t)), ..., J_n(K^j(t))\}$.

4. Pareto ranking of $[K^1(t), ..., K^j(t), ..., K^{n_{pop}}(t)]$ according to $\mathbf{J}(K^j(t))$.

5. Assign fitness to $K^j(t)$ based on its ranking , $\mathbf{f}'(K^j(t)) = \frac{1}{rank(K^j(t))}$, and apply fitness sharing, $\mathbf{f}(K^j(t)) = \frac{D(K^j)}{\sum_j D(K^j)} \sum_j \mathbf{f}'(K^j(t))$.

6. Directly migrate the elite, individuals with rank 1, to the $(t+1)th$ generation. This makes up $e\%$ of total population n_{pop}, where $e\%$ is up to 40%.

7. Apply the *tournament selection* [12] to generate $n_{pop} \cdot (95 - e)\%$ parents from the $(t)th$ generation. A linear crossover is used to produce $n_{pop} \cdot (95 - e)\%$ new individuals from these parents. Apply *mutation* to these new individuals.

8. Randomly generate 5% of total n_{pop} individuals in the search space Ω_K for the $(t+1)th$ generation.

9. Set $[K^1(t+1), ..., K^j(t+1), ..., K^{n_{pop}}(t+1)]$ as the current generation. Should this new generation achieve the optimization goal, stop the MOGA; otherwise go to step 3.

Table 2: MOGA configerations

$n_{pop} = 200$	$Pr(crossover) = 1$	$e\% \leqslant 30\%$
$n_{gen} = 60$	$Pr(mutation) = 0.02$	

Table 3: Design objectives before and after MOGA optimization

	Before Optimization		After Optimization	
	Simultaion	Experiment	Simulation	Experiment
$mean(\sigma_{PES})$	5.240	5.219	5.032	5.011
$var(\sigma_{PES})$	2.253	2.311	2.095	2.154
GM	$6.1dB$	$6.3dB$	$6.1dB$	$6.2dB$
PM	$42.1°$	$40.4°$	$41.0°$	$39.7°$
ω_o	$540Hz$	$544Hz$	$552Hz$	$557Hz$
S_∞	$5.7dB$	$5.5dB$	$5.4dB$	$5.2dB$

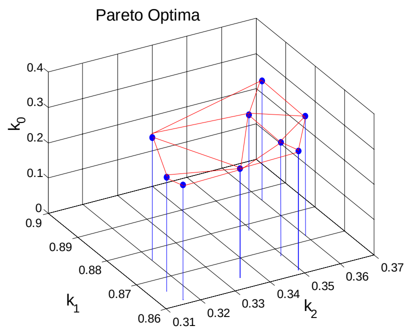


Figure 14: The Pareto optimal solutions by MOGA

5 Design Example

The disk drives used in our experiments have a recording density of 19,300 TPI and a sampling rate of 10,800Hz. The track-following controller for these drives is an extended PID controller in the following form

$$C(z^{-1}) = k_0 \cdot \mathbf{F}(z^{-1}; k_1, k_2, k_I) \tag{12}$$

where k_0 is the proportional gain, $\mathbf{F}(\cdot)$ is a function of z^{-1}, and constant parameters k_1, k_2, and k_I. A digital notch filter is added to attenuate the adverse effect due to the suspension resonance at 2700Hz. The integrator gain, K_I, is predetermined and hence not a target of optimization. So the tunable parameter set is $K = [k_0, k_1, k_2]$. The original controller used by these drives was $[k_0, k_1, k_2] = [0.243, 0.850, 0.261]$, which was hand-tuned by experienced engineers in the previous design cycle. PES of those drives were collected to build the statistical disturbance model. The search space $\Omega_K = [(0.01 : 0.98), (-0.8 : 0.99), (-0.8 : 0.99)]$ is the range of K that stabilizes the nominal plant. The gradient method used in [4] had always failed to converge in this range. To make our method more practical for most servo engineers in HDD industry, we follow the classical control framework. Since we are designing the controller for the mass produced HDDs with plant variations and disturbance uncertainties, σ_{PES} is not a constant but rather a random variable. It is more appropriate to assess the average performance $mean(\sigma_{PES})$ and the performance robustness $var(\sigma_{PES})$. The major objectives of HDD track following control therefore are

$$\begin{aligned} J_1(K) &= var(\sigma_{PES}) \\ J_2(K) &= mean(\sigma_{PES}) \end{aligned} \tag{13}$$

In order to get a precise and efficient prediction of $mean(\sigma_{PES})$ and $var(\sigma_{PES})$ for a given controller, a statistical model is built based on the PESs measured from a large population of drives [32]. This model not only covers the characteristics of a large population of drives but also requires little computational efforts in calculating $mean(\sigma_{PES})$ and $var(\sigma_{PES})$.

Design constraints are $PM_{MIN} = 36^o$, $GM_{MIN} = 5.5dB$, $\omega_{oMIN} = 500Hz$, and $S_{\infty MAX} = 6dB$. By following aforementioned guideline for handling constraints, the corresponding cost functions are defined as

$$\begin{aligned} J_3(K) &= Q[PM_{MIN} - PM(K)] \\ J_4(K) &= Q[GM_{MIN} - GM(K)] \\ J_5(K) &= Q[\omega_{oMIN} - \omega_o(K)] \\ J_6(K) &= Q[S_\infty(K) - S_{\infty MAX}] \end{aligned}$$

and the phase margin $PM(K)$, the gain margin $GM(K)$, the crossover frequency $\omega_o(K)$, and the peak of sensitivity function $S_\infty(K)$=$\|S(K, j\omega)\|_\infty$ are functions of the tunable parameter set

$K \in \Omega_K$. $J_3(K) \sim J_6(K)$ will be zero if the candidate solution K satisfies all constraints. Thus, the original constrained MOP is converted to a unconstrained MOP whose target is to minimize the cost vector $\mathbf{J}(K) = \{J_1(K), ..., J_i(K), ..., J_6(K)\}$ over K. $\mathbf{J}(K)$ is therefore called the *vectorial performance* index.

The parameter for MOGA is listed in Table 2. It took the MOGA 10.5 minutes in a PIII 550 computer to give a Pareto optimal set (Rank 1) with 9 solutions in the $60th$ generation, as shown in Figure 14. One solution, $K = [0.261, 0.864, 0.335]$, which has the minimal $var(\sigma_{PES})$ among the Pareto optimal set, was picked up and loaded into those drives. The performance comparison between this optimized controller and the original one is shown in Table 3. The 4% improvement of $mean(\sigma_{PES})$ and 7% of $var(\sigma_{PES})$ in experiments may be considered moderate because the original controller had been well optimized. It also can be seen that $mean(\sigma_{PES})$ and $var(\sigma_{PES})$ predicted by the model match up reasonably well with the experimental results. The disturbance model was then updated based on new PES measurements, and ready for the next run of optimization. This iterative optimization process can be repeated until satisfactory results are obtained.

5.1 Summary

In this report, a NCMOP of tuning the fixed-structure track following controller for HDDs has been solved by using the MOGA. A disturbance model was used to effectively predict the time-domain performance of candidate solutions for a large population of drives. The controller parameters were then tuned by MOGA, directly towards the minimization of $mean(\sigma_{PES})$ and $var(\sigma_{PES})$ under various frequency-domain constraints. As shown by simulations and experiments, the proposed method was capable of optimizing the controller in a *large range* in which gradient-based methods generally failed.

Our proposed method has the following advantages over existing tuning methods: First, compared to gradient based methods, the GA greatly increases the possibility of finding the global optima for nonconvex problems in large. Second, the GA offers almost linear computational complexity relative to the number of tunable variables. This is a very attractive feature for NCMOPs with a large number of variables, e.g., the track following controllers for dual-stage actuators usually have over twelve tunable variables for which the LMI+approximation+SDP method is practically unsolvable. Third, the MOGA gives designer multiple Pareto solutions without the trial and error for picking weights. Fourth, users can use any optimization setup as long as the cost functions are evaulatable in closed form.

However, like any optimization algorithm, there is “no free lunch” [33]. In the other word, no optimization algorithm is universally better than other algorithms. The major drawback to our approach is that the MOGA suffers from poor numerical trackability. Because the MOGA is stochastic rule based, it converges in some stochastic sense. Our experience shows that the Pareto sets are different from run to run, although they are all close to the global optima in the objective space. Our solution is to use a two-phase optimization method: use the results of MOGA as starting points and locally apply gradient based method.

References

[1] S. Boyd, V. Balakrishnan, E. Feron, and L. E. Ghaoui, "History of linear matrix inequalities in control theory," *Proceedings of the 1994 American Control Conference*, vol. 1, pp. 31–34, July 1994.

[2] M. G. Safonov, K. C. Goh, and J. H. Ly, "Control system synthesis via bilinear matrix inequalities," *Proceedings of the 1994 American Control Conference*, pp. 45–49, June 1994.

[3] K. Zhou, K. Glover, B. Bodenheimer, and J. Doyle, "Mixed h_2 and h_∞ performance objectives," *Proceedings of the 1990 American Control Conference*, pp. 2502–7, May 1990.

[4] H. S. Lee, "Controller optimization for minimum position error signals of hard disk drives," *Proceedings of the 2000 American Control Conference*, pp. 3081–3085, June 2000.

[5] R. Ehrlich and D. Curran, "Major HDD TMR sources and projected scaling with TPI," *IEEE Transactions on Magnetics*, vol. 35, pp. 885–91, March 1999.

[6] D. Abramovitvh, T. Hurst, and D. Henze, "Decomposition of baseline noise sources in hard disk position error signals in disk drives," *Proceedings of the American Control Conference*, vol. 5, pp. 2901–2905, 1997.

[7] S. Skogestad, *Multivariable Feedback Control.* Chichester, West Sussex, England: John Wiley and Sons Ltd., 1996.

[8] C. Scherer, P. Gahinet, and M. Chilali, "Multiobjective output-feedback control via LMI optimization," *IEEE Transactions on Automatic Control*, vol. 42, pp. 896–911, 1997.

[9] L. E. Ghaoui, F. Oustry, and M. Aitrami, "A cone complementarity linearization algorithm for static output-feedback and related problem," *IEEE Transactions on Automatic Control*, vol. 42, pp. 1171–1176, 1997.

[10] H. A. Hindi, B. Hassibi, and S. P. Boyd, "Multiobjective h_2/h_∞-optimal control via finite dimensional q-parametrization and linear matrix inequalities," *Proceedings of the 1998 American Contol Conference*, pp. 3244–49, June 1998.

[11] T. Kawabe and T. Tagami, "A real coded genetic algorithm for matrix inequality design approach of robust PID controller with two degree of freedom," *Proceedings of the 12th IEEE International Symposium on Intelligent Control*, pp. 119–124, July 1997.

[12] D. E. Goldberg, *Genetic Algorithms in Search Optimization, and Machine Learning.* Reading, MA: Addison Wesley.

[13] R. Bellman, *Adaptive Control Process: A Guided Tour.* Princeton, NJ: Princeton University Press, 1961.

[14] J. H. Holland, "Outline for a logical theory of adaptive systems," *Journal of the Association for Computing Machinery*, vol. 3, pp. 297–314, 1962.

[15] Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs, Second Extended Edition.* New York: Springer-Verlag Berlin Heidelberg, 1992.

[16] H. B. Kamepalli, "The optimal basics for GAs," *IEEE Potentials*, pp. 25–27, April, 2001.

[17] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handing with evolutionary algorithms - part i: A unified formulation," *IEEE Transactions on System, Man, and Cybernetics- Part A: Systems And Humans*, vol. 28, January 1998.

[18] Z. Michalewicz and C. Z. Janikow, "Handling constraints in genetic algorithms," in *Proceeding of 4th International Conference in Genetic Algorithms* (R. K. Belew and L. B. Booker, eds.), (San Mateo, CA), pp. 151–157, Morgan Kaufmann, 1991.

[19] K. D. Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975.

[20] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," *Proceeding of 1st International Conference on Genetic Algorithms*, pp. 93–100, 1985.

[21] H. Tamaki, H. Kita, and S. Kobayashi, "Multi-objective optimization by genetic algorithm: A review," *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pp. 517–522, May 1996.

[22] B. Chen, Y. Cheng, and C. H. Lee, "A genetic approach to mixed h_2/h_∞ optimal PID control," *IEEE Control Systems*, pp. 51–60, Octobor 1995.

[23] A. J. Chipperfield, N. V. Dakev, P. J. Fleming, and J. F. Whidborne, "Multiobjective robust control using evolutionary algorithms," *IEEE Proc. of the International Conference on Industrial Tech.*, pp. 269–273, 1996.

[24] *Optimization Toolbox (Version 2) User's Guide.* The MathWorks Inc., 2000.

[25] A. Buchanan, *Ethics, Efficiency, and the Market.* Totowa, NJ: Rowman and Allanheld Texts in Philosophy, 1985.

[26] C. M. Fonseca and P. J. Fleming, "Multiobjective genetic algorithm made easy: Selection, sharing and mating restriction," *Genetic Algorithm in Engineering Systems: Innovations and Applications*, pp. 45–52, September 1995. Conference Publication No 414.

[27] T. K. Liu, T. Ishihara, and H. Inooka, "Multiobjective control systems design by genetic algorithms," *Proceedings of the 34th SICE Annual Conference*, pp. 1521–26, July 1995.

[28] D. E. Goldberg and P. Segrest, "Finite markov chain analysis of genetic algorithm," *Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 1–8, 1987.

[29] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," *Proceeding of the First ICEC*, pp. 82–87, 1994.

[30] N. Srinivas and K. Deb, "Multiobjective optimization using non-dominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2(3), pp. 221–248, 1994.

[31] G. F. Franklin, J. Powell, and M. L. Workman, *Digital Control of Dynamic Systems.* Addison-Wesley, 2nd ed., 1990.

[32] B. Zhu, L. Guo, and M. Tomizuka, "A statistical PES model for direct controller optimization towards minimum PES in hard disk drives.." Maxtor Techanical Report, July 2000.

[33] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.