8 Multi-objective Genetic Algorithms for the Method of Inequalities

Tung-Kuan Liu and Tadashi Ishihara

Abstract. Global search capability of genetic algorithms (GAs) provides an attractive method for solving inequalities. For multi-objective optimisation, various multi-objective genetic algorithms (MGAs) have been proposed. However, due to the fundamental difference between the method of inequalities and conventional multi-objective optimisation, it is not immediately apparent how MGAs should be used for solving inequalities. In this chapter, the use of MGAs in the method of inequalities is discussed. For the effective use of MGAs, an auxiliary vector performance index, related to the set of inequalities, is introduced. A simple MGA with Pareto ranking is proposed in conjunction with the auxiliary vector index. The performance of the proposed MGA is tested on a special set of test problems and control design benchmark problems.

8.1 Introduction

Search methods play an important role in the new framework for control systems design presented in this book. One of the global search methods is the genetic algorithms (GAs), which has been proposed by Holland (1975) to apply the principle of biological evolution to artificial systems. For conventional control systems design, GAs have been used to find a controller parameter minimising a single scalar performance index, for which standard non-global hill climbing techniques can fail to find the global minimum, when there are several local minima (Gtefenstete, 1986; Krishnakumar and Goldberg, 1992; Kristinsson and Dumont, 1992; Varsek *et al*, 1993; Porter and Jones, 1992). For vector valued performance index, various multi-objective genetic algorithms (MGAs) (*e.g.*, Goldberg, 1989; Schaffer, 1985; Fonseca and Fleming, 1993; Srinivas and Deb, 1995; Coello *et al*, 2002) have been proposed to find a set of Pareto minimisers.

The multi-objective nature of control systems design is explicitly taken into account in the method of inequalities proposed by Zakian (Zakian and Al-Naib, 1973). Unlike conventional multi-objective methods of design, the method of inequalities seeks an admissible solution, which is not necessarily a Pareto minimiser. Due to this fundamental difference, it is not readily apparent how MGAs should be used to solve inequalities formulated according to the method of inequalities (see the principle of inequalities in Chapter 1).

The authors (Liu *et al*, 1994; Liu, 1997) have proposed a genetic inequalities solver using a simple MGA for an auxiliary vector index related to the set of inequalities that represent the design specifications. The Pareto minimiser of the proposed method reduces to utopian (see below for definition) solutions if the design problem, formulated by the method of inequalities, is solvable. The proposed solver exploits the global search capability of an MGA to solve inequalities.

Fonseca and Fleming (1993) have proposed an MGA and have used it to obtain a Pareto minimiser subject to inequality constraints. Whidborne *et al* (1996) have applied this method to the method of inequalities and have compared it with existing local search methods. Note that computation of a Pareto minimiser is not required in the method of inequalities. Fonseca and Fleming (1998) have extended their method to handle more general preference expressions. The extended method includes the genetic inequalities solver proposed by the authors as a special case. However, the advantage of the fitness assignment method used by Fonseca and Fleming (1993, 1998) is lost when the method is specialised to solve inequalities. In addition, Fonseca and Fleming (1998) have not discussed simple but important properties discussed by the authors (Liu *et al*, 1994; Liu, 1997) for the method of inequalities.

In this chapter, the effective use of MGAs in the method of inequalities is discussed. In particular, the result given by the authors (Liu *et al*, 1994; Liu, 1997) is presented in more detail with new illustrative results.

This chapter is organised as follows: In Section 8.2, the auxiliary vector index related to the design specifications of the method of inequalities is introduced. Some properties of the set of Pareto minimisers for the auxiliary vector index are discussed. A simple genetic inequalities solver proposed by the authors is discussed in Section 8.4. In Section 8.5, the performance of the proposed genetic solver is examined experimentally for the test problems in Chapter 6. The results for the control design benchmark problems in Chapter 7 is given in Section 8.5. Conclusions are given in Section 8.6.

8.2 Auxiliary Vector Index

In this section, the auxiliary vector index is introduced for the use of MGAs to solve the inequalities

$$\phi_i(c) \le \varepsilon_i \quad i \in \tilde{M} = \{1, 2, \ldots, M\} \tag{8.1}$$

where $\phi_i(c)$ is an objective function, c is a design point and ε_i is a tolerance or bound specified by the designer. The computational problem is to determine any value of c that satisfies (8.1) or, if such a value does not exist, to confirm this fact. This is called the admissibility problem.

Define the objective vector

$$\Phi(c) \triangleq \left[\phi_1(c), \phi_2(c), \cdots, \phi_M(c)\right] \tag{8.2}$$

For an effective use of an MGA to solve the inequalities (8.1), the authors (Liu *et al*, 1994) have introduced scalar indices related to the inequality performance specifications (8.1) as

$$\lambda_i(c, \varepsilon_i) \equiv \begin{cases} 0 & \text{if } \phi_i(c) \le \varepsilon_i \\ \phi_i(c) - \varepsilon_i & \text{if } \phi_i(c) > \varepsilon_i \end{cases} \tag{8.3}$$

for $i \in \tilde{M}$. These indices are also related to the ordering relations employed by the local search method called the moving boundaries process (Zakian and Al-Naib, 1973).

Define the auxiliary vector index

$$\Lambda(c, \varepsilon) \triangleq \big[\lambda_1(c, \varepsilon_1),\, \lambda_2(c, \varepsilon_2),\, \cdots,\, \lambda_M(c, \varepsilon_M)\big] \tag{8.4}$$

where

$$\varepsilon \triangleq \big[\varepsilon_1,\, \varepsilon_2,\, \cdots,\, \varepsilon_M\big] \tag{8.5}$$

is a tolerance vector.

The following notations used by Sawaragi *et al* (1985) for vector inequalities are employed. For any two vectors x and y, the vector inequality $x \le y$ means that, for each element, $x_i \le y_i$ holds. In addition, the notation $x \prec y$ is used to mean $x \le y$ and $x \ne y$.

For the objective vector (8.2), the following definition is introduced using the above notations.

Definition 8.1. Consider an objective vector $\Phi(c)$ with a design c. The design c_0 is said to be a utopian for the objective vector $\Phi(c)$ if and only if $\Phi(c_0) \le \Phi(c)$ for all c. A vector c is said to be a Pareto minimiser if and only if there exists no vector $\bar{c}\,(\ne c)$ satisfying $\Phi(\bar{c}) \prec \Phi(c)$. A utopian is a Pareto minimiser. A Pareto minimiser that is not utopian is said to be a strict Pareto minimiser.

In Chapter 1, the relation between the ordering relations induced by the principle of inequalities and the Pareto ordering has been discussed. The following result is a restatement of the result in terms of the auxiliary vector index.

Property 8.1. A design c is superior to $\bar{c}$ in the sense of the principle of the inequalities with respect to the objective vector $\Phi(c)$ if and only if c is superior to $\bar{c}$ in the Pareto sense with respect to the auxiliary vector index $\Lambda(c, \varepsilon)$.

Proof. Assume that $\phi_j(\bar{c}) \le \varepsilon_j$ for $j \in \tilde{N} \subset \tilde{M}$. Define $\hat{\Phi}(\bar{c})$ as the vector obtained by deleting the elements $\phi_j(\bar{c})$ $(j \in \tilde{N})$ from the objective vector $\Phi(\bar{c})$. Then the superiority of c in the sense of the principle of inequalities holds if and only if $\hat{\Phi}(c) \prec \hat{\Phi}(\bar{c})$ and $\phi_j(\bar{c}) \le \varepsilon_j \Rightarrow \phi_j(c) \le \varepsilon_j\,(j \in \tilde{N})$.

The condition $\hat{\Phi}(c) \prec \hat{\Phi}(\bar{c})$ is equivalent to $\lambda_i(c, \varepsilon_i) \leq \lambda_i(\bar{c}, \varepsilon_i)$ for all $i \in \tilde{M} \backslash \tilde{N}$ with $\lambda_i(c, \varepsilon_i) < \lambda_i(\bar{c}, \varepsilon_i)$ for some $i \in \tilde{M} \backslash \tilde{N}$. In addition, the condition $\phi_j(\bar{c}) \leq \varepsilon_j \Rightarrow \phi_j(c) \leq \varepsilon_j$ is equivalent to the inequality $\lambda_j(c, \varepsilon_j) \leq \lambda_j(\bar{c}, \varepsilon_j)$ with $\lambda_j(\bar{c}, \varepsilon_j) = 0$. Therefore, the two conditions $\tilde{\Phi}(c) \prec \tilde{\Phi}(\bar{c})$ and $\phi_j(\bar{c}) \leq \varepsilon_j \Rightarrow \phi_j(c) \leq \varepsilon_j$ $(j \in \tilde{N})$ are equivalent to $\Lambda(c, \varepsilon) \prec \Lambda(\bar{c}, \varepsilon)$. In the case that $\phi_i(\bar{c}) > \varepsilon_i$ for all $i \in \tilde{M}$, c is superior to $\bar{c}$ in the sense of the principle of the inequalities if and only if $\Phi(c) \prec \Phi(\bar{c})$, which is obviously equivalent to $\Lambda(c, \varepsilon) \prec \Lambda(\bar{c}, \varepsilon)$ with $\lambda_i(c, \varepsilon_i) > 0$ and $\lambda_i(\bar{c}, \varepsilon_i) > 0$ for all $i \in \tilde{M}$. Consequently, the equivalence of the superiority conditions is proved.

For the effective use of MGAs in the method of inequalities, it is proposed to apply an MGA to the auxiliary vector index. It readily follows from the definitions (8.3), (8.4) and (8.5) that the solvability of the inequalities (8.1) can be stated in terms of the auxiliary vector index as follows.

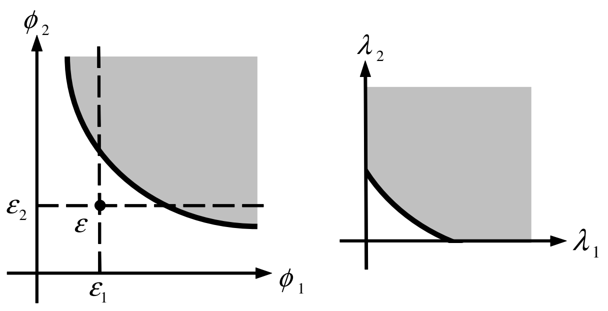
Property 8.2. A design c_0 satisfying all the inequalities (8.1) exists if and only if c_0 is a utopian solution of the auxiliary vector index $\Lambda(c, \varepsilon)$, *i.e.*, $\Lambda(c_0, \varepsilon) = 0$.

The use of the auxiliary vector index has the following advantages:

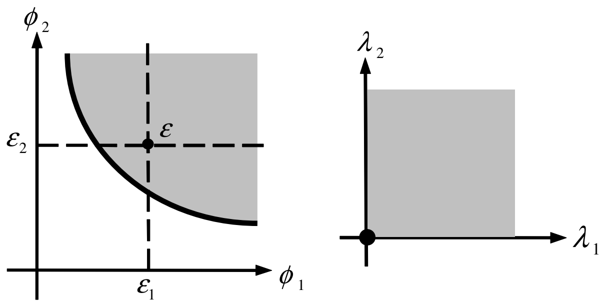
(a) The global search capability of MGAs can be exploited to find a solution of the inequalities (8.1).
(b) An MGA applied with the auxiliary vector index always generates designs belonging to a strict Pareto set even if no design satisfies the given design specifications (8.1). The Pareto set provides the designer useful information for modifying the design specifications.

It is apparent that the first advantage is the main reason for using an MGA for solving the inequalities (8.1). If the design problem formulated by the method of inequalities has a solution, the Pareto optimal set generated by the MGA applied to the auxiliary vector index eventually reduces to utopians. The second advantage works only for ill-posed problems but it provides information that cannot be obtained by a failure of a local search method. When a local search method fails to find an admissible solution, it is not easy to decide whether the algorithm is trapped in a local region or no solution satisfying design specifications exists.

Although the strict Pareto set for the auxiliary vector index (8.4) appears only for ill-posed problems, it is interesting to compare it with the Pareto set of the objective vector (8.2) related to the design specifications (8.1). For the two dimensional performance index ($M = 2$), the relation between the two Pareto sets in $\phi_1 - \phi_2$ and $\lambda_1 - \lambda_2$ planes can be illustrated as in Figure 8.1, where the Pareto set is indicated by bold lines and the shaded areas show the achievable regions. The case that the tolerance vector ε is outside of the achievable region in $\phi_1 - \phi_2$ is shown in Figure 8.1(a). In this case, no utopian exists and the Pareto set of the auxiliary vector index $\Lambda(c, \varepsilon)$ in the $\lambda_1 - \lambda_2$



(a) Strict Pareto set



(b) Utopians

Fig. 8.1. The two Pareto sets

plane is a clipped version of that of the vector performance index (8.2). In the case that the tolerance ε is inside of the achievable region, the result is shown in Figure 8.1(b). In the $\lambda_1 - \lambda_2$ plane, the Pareto set is the origin which corresponds to utopian solutions of the auxiliary vector index $\Lambda(c, \varepsilon)$.

In the following, simple properties conjectured from the two dimensional case are given for the general case. The Pareto sets for the objective vector $\Phi(c)$ and the auxiliary vector index $\Lambda(c, \varepsilon)$ are denoted by P and $Q(\varepsilon)$, respectively.

It is known that the principle of inequalities provides a Pareto minimiser for the objective vector (8.2) if the tolerance vectors are properly chosen (Zakian, 1996). For the two dimensional case, this result corresponds to the case that the tolerance vector ε is on the Pareto set as shown in Figure 8.2(a). The following is a restatement of this fact.

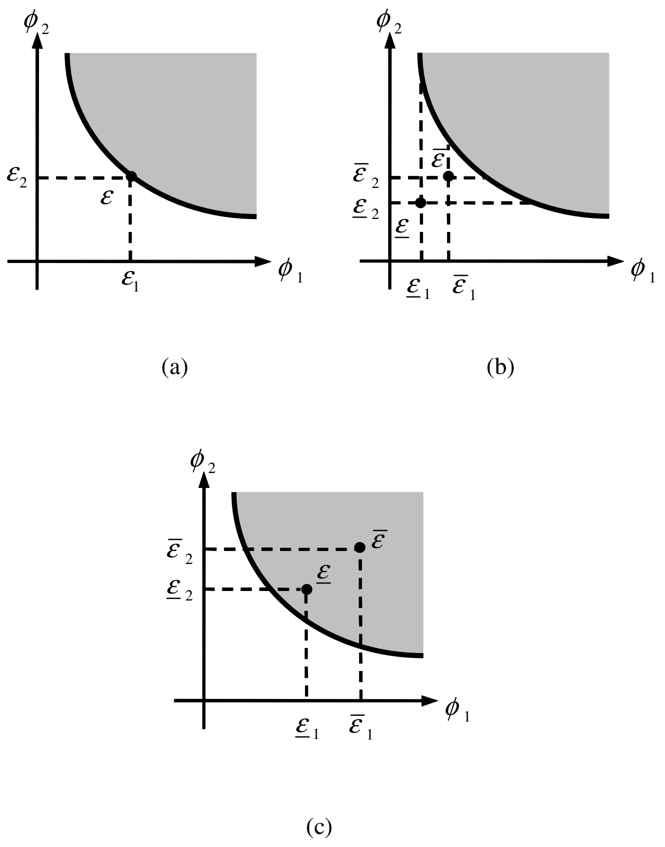

ϕ_2
ε_2
ε
ε_1
ϕ_1
(a)
ϕ_2
$\overline{\varepsilon}$
$\overline{\varepsilon}_2$
$\underline{\varepsilon}_2$
$\underline{\varepsilon}$
$\underline{\varepsilon}_1$
$\overline{\varepsilon}_1$
ϕ_1
(b)
ϕ_2
$\overline{\varepsilon}_2$
$\overline{\varepsilon}$
$\underline{\varepsilon}$
$\underline{\varepsilon}_2$
$\underline{\varepsilon}_1$
$\overline{\varepsilon}_1$
ϕ_1
(c)

Fig. 8.2. The tolerance vector and the Pareto set

Property 8.3. If the auxiliary vector index $\Lambda(c, \varepsilon)$ has a unique utopian c^* satisfying $\Lambda(c^*, \varepsilon) = 0$, then the utopian c^* is a Pareto minimiser of the objective vector $\Phi(c)$.

Proof. Assume that the unique utopian c^* is not a Pareto minimiser for the objective vector $\Phi(c)$. Then, by the definition of the Pareto minimiser, there exists vector $c^{**} \neq c^*$ satisfying $\Phi(c^{**}) \prec \Phi(c^*)$. Note that $\Lambda(c^*, \varepsilon) = 0$ is equivalent to $\Phi(c^*) \leq \varepsilon$. It follows from $\Lambda(c^*, \varepsilon) = 0$ and $\Phi(c^{**}) \prec \Phi(c^*)$ that $\Phi(c^{**}) \leq \varepsilon$, *i.e.*, $\Lambda(c^{**}, \varepsilon) = 0$. Consequently, c^{**} is also a utopian. This contradicts the uniqueness assumption.

Two properties corresponding to the case shown in Figure 8.2(b) for the two dimensional case are given. The first property is concerned with the inside of the Pareto set $Q(\varepsilon)$ for $\Lambda(c,\varepsilon)$.

Property 8.4. Let $Q_+(\varepsilon)$ denote the Pareto sets consisting of the designs satisfying $\Lambda(c,\varepsilon) > 0$. Then, $Q_+(\varepsilon) \subset P$, where P is the Pareto set for $\Phi(c)$, holds for $\varepsilon > 0$. In addition, for any two tolerance vectors satisfying $\varepsilon \prec \bar{\varepsilon}$, $Q_+(\varepsilon)$ and $Q_+(\bar{\varepsilon})$ satisfy $Q_+(\varepsilon) \supset Q_+(\bar{\varepsilon})$.

Proof. Note that $\Lambda(c,\varepsilon) = \Phi(c) - \varepsilon > 0$ by the assumption. Consider a design $c \in Q_+(\varepsilon)$, for which there exists no design superior to c. Assume that $c \notin P$, which implies that there exist a design c^* such that $\Phi(c^*) \prec \Phi(c)$. Then, for the design c^*, $\Phi(c^*) - \varepsilon \prec \Phi(c) - \varepsilon = \Lambda(c,\varepsilon)$ holds, which is a contradiction to $c \in Q_+(\varepsilon)$. Consequently, $c \in P$. Write the set of designs satisfying $\Phi(c) > \varepsilon$ as $D(\varepsilon)$. The set $Q_+(\varepsilon)$ can be written as $Q_+(\varepsilon) = P \cap D(\varepsilon)$. Since $D(\varepsilon) \supset D(\bar{\varepsilon})$ for $\varepsilon \prec \bar{\varepsilon}$, it is obvious that $Q_+(\varepsilon) \supset Q_+(\bar{\varepsilon})$ holds.

The second property is concerned with the edge of the Pareto set $Q(\varepsilon)$.

Property 8.5. Assume the following two conditions:

(i) There exists $c_0 \in P$ such that $\phi_j(c_0) < \varepsilon_j$ for $j \in \tilde{N} \subset \tilde{M}$.
(ii) For any $c_0 \in P$ satisfying i, there exists $c_1 \in P$ such that $\phi_j(c_0) < \phi_j(c_1) < \varepsilon_j$ for $j \in \tilde{N}$ and $\hat{\Phi}(c_1) \prec \hat{\Phi}(c_0)$ where $\hat{\Phi}(c)$ denotes the vector obtained by deleting the elements $j \in \tilde{N}$ from $\Phi(c)$.

Then, there exists $c_* \in P$ satisfying $\phi_j(c_*) = \varepsilon_j$ for $j \in \tilde{N} \subset \tilde{M}$ and $c_* \in Q(\varepsilon)$,

Proof. Note the existence of $c_1 \in P$ satisfying $\hat{\Phi}(c_1) \prec \hat{\Phi}(c_0)$ for $c_0 \in P$ is possible although there exists no $c \in P$ such that $\Phi(c) \prec \Phi(c_0)$. Define $\hat{\Lambda}(c_0,\hat{\varepsilon}) \triangleq \hat{\Phi}(c_0) - \hat{\varepsilon}$ where $\hat{\varepsilon}$ denotes the vector obtained by deleting the elements $j \in \tilde{N}$ from ε. The assumptions guarantee that, for any $c_0 \in P$ satisfying the first condition, it is possible to find $c_1 \in P$ such that $\hat{\Lambda}(c_1,\hat{\varepsilon}) \prec \hat{\Lambda}(c_0,\hat{\varepsilon})$. Note that $\hat{\Lambda}(c_1,\hat{\varepsilon}) \prec \hat{\Lambda}(c_0,\hat{\varepsilon})$ is equivalent to $\Lambda(c_1,\varepsilon) \prec \Lambda(c_0,\varepsilon)$. By continuing this procedure, it is possible to construct a sequence of the decisions $\{c_k, k = 0,1,2,\ldots\}$ which converges to $c_* \in P$ satisfying $\phi_j(c_*) = \varepsilon_j$. Since no decision $c \neq c_*$ exists such that $\hat{\Lambda}(c,\hat{\varepsilon}) \prec \hat{\Lambda}(c_*,\hat{\varepsilon})$, *i.e.*, $\Lambda(c,\varepsilon) \prec \Lambda(c_*,\varepsilon)$, it is concluded that $c_* \in Q(\varepsilon)$.

It can easily be checked that the two assumptions are satisfied at least for the two dimensional case where $N = 1$ and the objective functions $\phi_i(c)$ $(i = 1,2)$ are smooth functions of the design c. Let $Q_*(\varepsilon)$ denote the set of designs satisfying the above property. Then it follows from Property 8.4 and Property 8.5 that the Pareto set $Q(\varepsilon)$ can be written as $Q(\varepsilon) = Q_*(\varepsilon) \cup Q_+(\varepsilon)$.

For non-unique utopians, the following property is obtained. The result corresponds to the two dimensional case shown in Fig. 2 (c).

Property 8.6. Let $Q(\varepsilon)$ denote the Pareto optimal set for the auxiliary vector index $\Lambda(c, \varepsilon)$. Assume that, for the tolerance vectors ε and $\bar{\varepsilon}$ satisfying $\varepsilon \prec \bar{\varepsilon}, Q(\varepsilon)$ and $Q(\bar{\varepsilon})$ consist of utopian solutions. Then, $Q(\varepsilon) \subset Q(\bar{\varepsilon})$.

Proof. The result readily follows from the fact that $c \in Q(\varepsilon)$, where the set $Q(\varepsilon)$ consists of utopian solutions, if and only if $\Phi(c) \leq \varepsilon$.

Properties 8.1—8.6 suggest that the use of the auxiliary vector index restricts the Pareto set in the region of interest and reduces the necessary computation burden. In addition, the properties provide some qualitative understanding of the Pareto set of $\Lambda(c, \varepsilon)$. According to the range of the tolerance vectorε, the Pareto set $Q(\varepsilon)$ can be classified into three types:

A. For sufficient small ε, $\Lambda(c, \varepsilon) > 0$ for all designs. Then the Pareto set $Q(\varepsilon)$ coincides with the Pareto set P for $\Phi(c)$.
B. For medium range of ε, designs satisfying $\Lambda(c, \varepsilon) \geq 0$ appear. The Pareto set $Q(\varepsilon)$ includes the Pareto sets $Q_+(\varepsilon)$ and $Q_*(\varepsilon)$ included in the Pareto set P for $\Phi(c)$. As ε increases, the size of $Q(\varepsilon)$ decreases until a utopian appears.
C. For sufficiently large ε, $Q(\varepsilon)$ consists of utopians but includes solutions not contained in P. As ε increases, the size of $Q(\varepsilon)$ increases.

A local parameter search algorithm is not effective for the type A and B since no utopian exists. For this type, an MGA can generate an approximate Pareto set. Noting the above qualitative property, the designer can use the information of the approximate Pareto set to soften the design specifications so that a utopian can be found. On the other hand, for the type C, an MGA may find many utopian solutions. In such a case, the above qualitative properties can be used to tighten the design specifications to reduce the number of the utopian solutions.

8.3 Genetic Inequalities Solver

The method proposed in the previous section can be used with any MGA to construct an inequalities solver. It should be noted that the objective of the method of inequalities is to find a solution satisfying the set of the inequalities while that of an MGA is to characterise the Pareto optimal set. An MGA constructed by taking account of the difference is more appropriate to use for an inqualities solver. In this section, such an MGA is proposed.

8.3.1 Bias Problem of MGAs

It is well known that a simple GA has a bias problem called *genetic drift* for a scalar performance index with multiple minima (*e.g.*, Goldberg, 1989). The GA search has a tendency to concentrate to on one of the minima. The source of the bias is recognised as the stochastic error of the evolution. The effect of the genetic drift can be reduced by preserving population diversity. Various methods including *sharing* and *crowding* have been proposed to reduce the bias (*e.g.*, Goldberg, 1989).

For MGAs, the bias problem is more complex. The vector evaluated genetic algorithm (VEGA) proposed by Schaffer (1985) is well known as a simple MGA. In VEGA, the population is divided into equally sized subpopulations corresponding to the scalar components in the vector performance index. The fitness is assigned by the independent championship in each component of a vector performance index but the mating and the crossover are performed for the whole population. The VEGA has a tendency to ignore middling individuals. Such a bias is apparently undesirable to characterise the whole Pareto set. The genetic drift alone is not a source of the bias of the VEGA.

8.3.2 Ranking Methods for MGAs

MGAs differ from conventional GAs in the fitness assignment. Notice that no Pareto concept is used in the evolution of the VEGA. A part of the bias in the VEGA can be considered due to the bias in the fitness assignment. To reduce the bias, Goldberg (1989) has suggested a fitness assignment using the Pareto (non-dominated) ranking. The procedure is described as follows: First, the Pareto set of the current population is identified. Individuals in the Pareto set are assigned a rank of 1. The Pareto set is removed from the current population. The Pareto set of the remaining population is identified and individuals in the second Pareto set are assigned a rank of 2. This ranking procedure is continued until all individuals in the current population are assigned ranks. The fitness of an individual is assigned based on the rank.

Fonseca and Fleming (1993) have proposed an MGA called the multiple objective genetic algorithm (MOGA) using a ranking method different from the Pareto ranking: The rank of an individual is defined by one plus the number of the other individuals dominating the individual. By this definition, an individual in the non-dominated set is assigned a rank of 1 as in the Pareto ranking. However, the rank of an individual outside of the Pareto set can be different from that obtained by the Pareto sorting. The comparison of the ranking methods is shown in Fig. 3 for the two dimensional case. The ranks of the two points indicated by the arrows in the method of Fonseca and Fleming are different from the ranks obtained by the Pareto ranking.

8.3.3 A Simple MGA with the Pareto Ranking

Fonseca and Fleming (1993) have pointed out that their ranking method is useful to take account of the preference of the decision maker. In addition, they claim that their ranking method makes it easy to analyse some properties mathematically. Using their sorting method, they have proposed an optimiser

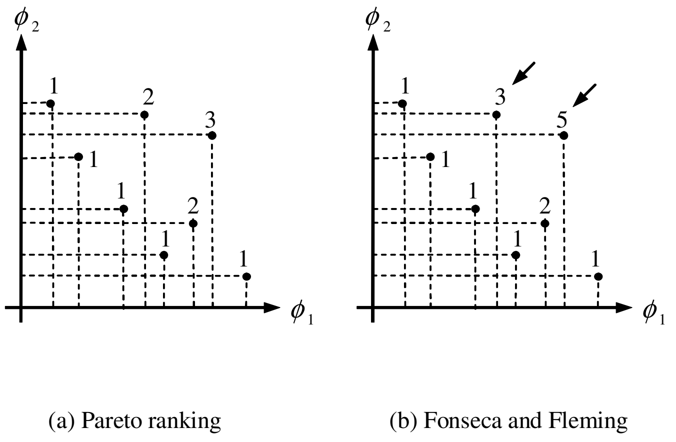


Fig. 8.3. Comparison of the ranking methods

(Fonseca and Fleming, 1998), which can deal with the method of inequalities as a special case.

All the inequality design specifications in the method of inequalities are considered to have equal importance. Preference ordering of the inequalities in the method of inequalities may introduce unnecessary confusions. It should be noted that the advantage of the fitness assignment method used by Fonseca and Fleming (1993, 1998) is lost when the optimiser (Fonseca and Fleming, 1998) is specialised to the method of inequalities. The fitness assignment based on the Pareto ranking is essential for the use of a MGA in the method of inequalities since it explicitly represents that the method of inequalities does not introduce any preference order among performance indexes.

Goldberg (1989) has pointed out that the Pareto ranking is not sufficient to reduce the bias. He suggested the use of some sharing scheme with the Pareto ranking. Following the suggestion, Srinivas and Deb (1995) have proposed the non-dominated sorting genetic algorithm (NSGA) and have shown its superiority over VEGA by extensive simulations. Fonseca and Fleming (1993) have proposed to use a sharing method with their ranking method.

Independent of the above MGAs, the authors have proposed to use the auxiliary vector performance index with a MGA using the Pareto ranking. The MGA used by the authors does not include any method to reduce the bias other than the Pareto ranking.

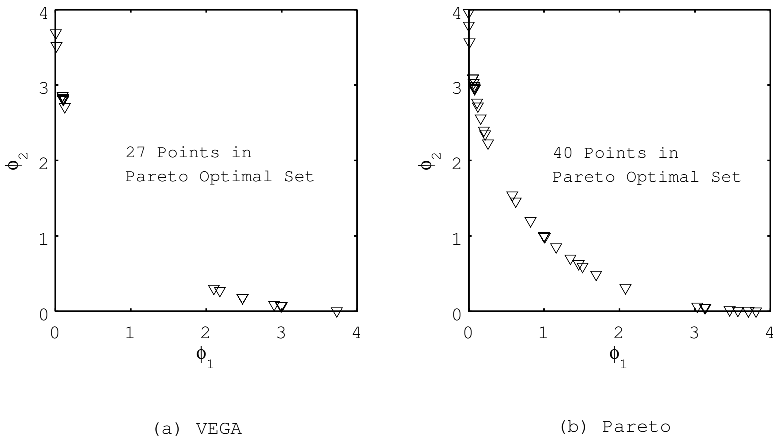


Fig. 8.4. Comparison of Pareto fronts

To show the performance of such a simple MGA, the Pareto set for simple two dimensional functions

$$\phi_1(t) = t^2, \quad \phi_2(t) = (t-2)^2, \quad -6 \leq t \leq 6 \tag{8.6}$$

is compared with that obtained by VEGA in Figure 8.4. In the comparison, typical GA parameters are used: the tunable parameter t is represented by 16-bit binary code, the population is 100, the crossover rate is 0.8, the mutation rate is 0.001 and the generation number is 3. It is apparent that the use of the Pareto ranking effectively reduces the bias against middling points of the Pareto set.

The primary reason to use an MGA for the method of inequalities is to exploit the global search capability. The appearance of the strict Pareto set means that the method of inequalities problem has no solution. For the application to the method of inequalities, the search capability of an MGA is more important than the characterisation of the Pareto set when the problem has no solution. A sharing method preserves population diversity by reducing fitness of a group of individuals. Intuitively, the search capability is sacrificed by introducing a sharing method. This observation together with the result of Figure 8.4 suggests that a simple MGA with the Pareto ranking is an appropriate choice for the method of inequalities.

8.3.4 An Efficient Computation of the Pareto Ranking

To make the MGA more efficient, it is possible to use an efficient computation method for the Pareto ranking. In MGAs, the brute force sorting is often used. For a set of N points in M dimensional space, the computational

Table 8.1. Comparison of the ranking methods

	Computation Time (Sec.)		
Number of Points	Proposed Method	Brute Force Method	Improvement Index (%)
30	0.0084	0.0067	−20.24
50	0.0211	0.0181	−14.22
80	0.0453	0.0466	3.56
100	0.0552	0.0734	32.97
500	1.3141	1.798	36.82

Calculation condition: MATLAB 6.1 for Windows XP on PC (Intel P4, 2.53GHz, 1GB RAM)

complexity for the brute force sorting is $O(MN^2)$. It is known that the algorithm found by Kung *et al* (1975) provides the computational complexity $O(N(\log N)^{M-2})+O(N\log N)$, which is smaller than the computational complexity $O(MN^2)$ required for brute force sorting. Kung's algorithm is of great theoretical importance. However, this computational complexity is a worst case estimate that is valid only for sufficiently large N. It has been reported by Gtefenstette (1986) that the population to be generated by GA is usually at most several hundred, sometimes less than hundred. It is doubtful that the Kung's estimate is valid for practical GA computation.

A heuristic computation procedure for non-dominated sorting is proposed. The proposed sorting scheme is based on a simple observation that, if two vectors x and $y$with non-negative elements satisfy the relation $x \prec y$, then the inequality

$$\sum_{i=1}^{M} x_i \le \sum_{i=1}^{M} y_i \tag{8.7}$$

holds. Consequently, the inequality (8.7) is a necessary condition for $x \prec y$.

Using the above observation, we propose the following improved ranked-based fitness assignment method which includes the sorting using (8.7). Suppose that a set $\Pi \equiv \{\Theta_1, \Theta_2, \ldots, \Theta_N\}$ with $\Theta_i \equiv \{\theta_{i1}, \theta_{i2}, \ldots, \theta_{iM}\}$ is given. Consider the following procedure.

Step 1: Sort Π from the least to the largest according to $\sum_{j=1}^{M}\theta_{ij}$. Let $K \equiv 1$.
Step 2: Let $Q \equiv \emptyset$, $G \equiv \emptyset$.
Step 3: Let the first point Θ_1 of Π be the criterion.
Step 4: Remove the dominated points Θ_i by checking whether or not the following conditions are satisfied.

$$\max_j g_{ij} \le 0 \wedge (\exists_j\ )(g_{ij} < 0) \tag{8.8}$$

where g_{ij} is the j-th component of the vector $g_i \equiv \Theta_1 - \Theta_i$, $i = 2, 3, \cdots, N$ and $j = 1, 2, 3 \cdots, m$.

Step 5: Remove the criterion Θ_1 and the dominated points Θ_i from Π. Save Θ_1 to a temporary non-dominated set Q, Θ_i to a temporary dominated set G.

Step 6: If $\Pi \neq \emptyset$ then go to **Step 3**.

Step 7: Rank the points of Q as K. Let $\Pi \equiv G$ and $K \equiv K + 1$.

Step 8: If $\Pi \neq \emptyset$ then go to **Step 2**.

Step 9: The fitness to points is given by the following linear function

$$V_N(i) = \frac{2(f_{\max} - 1)}{N - 1} [N - \text{rank}(i)] \tag{8.9}$$

where rank(i), which is less than or equal to N, are the ranks of points and $f_{\max}$ considered as a GA diversity maintaining parameter (Kristinsson and Dumont, 1992) can be specified by a designer. In this chapter, it is taken as $1 < f_{\max} \leq 2$.

In the worst case, the computational complexity of the proposed sorting procedure is $O(MN^2) + O(N \log N)$ which is not better than $O(MN^2)$ required for the brute force sorting. However, in the most optimistic case, the computational complexity required by the proposed method is $O(MN) + O(N \log N)$. Note that the brute force sorting always requires the computational complexity $O(MN^2)$.

To show the effectiveness of the proposed sorting method, a simple numerical experiment is performed. The trial vector sets are random matrix, made by MATLAB built-in functions, with ten elements uniformly distributed in the interval $[0, 10]$. Each sorting method is implemented 100 times and the mean calculation time is calculated. The result is summarised in Table 8.1 where the improvement index is defined as $R_p = (t_b - t_p)/t_p$, where t_b is the calculation time of the brute force sorting and t_p is that of the proposed sorting. The index R_p increases as the number of points n increases. The result shows that the proposed method provides faster sorting than the brute force method.

8.4 Numerical Test Problems

The ability of the proposed genetic inequalities solver is evaluated by the experiment for the test problems given in Chapter 6. There are 14 problems in

Table 8.2. GA parameters used in the experiment

Population Size	Reproduction Rate	Crossover Rate	Mutation Rate
100	0.8	1.0	0.01

Table 8.3. Summary of the search results

Problem Number	Average Number of Function Evaluations	Maximum Number of Function Evaluations	Minimum Number of Function Evaluations	Number of Failed Searches
1	33036	66340	11140	0
2	21324	72500	1380	0
3	74404	194420	17060	0
4	16860	36820	2500	0
5.1	164284	240020	30660	2
5.2	90884	221060	5380	0
5.3	123012	240020	1780	1
5.4	51668	125700	2580	0
5.5	84892	178100	15860	0
6.1	68156	215940	4180	0
6.2	65068	164740	2740	0
6.3	83092	210820	9940	0
6.4	74724	174180	2980	0
6.5	78796	162020	7460	0

total. For each problem, it is required to find an admissible point satisfying the inequality defined for the scalar-valued function with the two variables. Since the problems are not multi-objective, this experiment evaluates the search capability of the proposed MGA as a simple GA. Many possibilities exist for the choice of the GA parameters used in the experiment. The standard choice of the GA parameters shown in Table 8.2 are used for all the problems. To satisfy the precision requirement of the problems, a string of 60 bits (30 bits for each variable) is used to represent an individual in the GA. For each test problem, ten trials are performed using randomly generated initial populations. The search is terminated when an admissible solution is found or 3000 generations are obtained. The results are summarised in Table 8.3 where the problem numbers are defined in Chapter 6. The MGA can successfully find an admissible solution in all the trials except for the problems 5.1 and 5.3 in which a small number of trials failed among the ten trials. Note that 230020 times function evaluations correspond to 3000 generations. The results of all the trials in the problems 5.1 and 5.3 are shown in Table 8.5. It is seen that the final points of the searches for the failed cases (the third and sixth trials in the problem 5.1 and the sixth trial in the problem 5.3) are very close to the admissible regions, which can be confirmed by the value of the test functions. It is fair to say that the searches are almost successful. The source of the difficulty is not easy to identify. However, it is confirmed experimentally that the improved search results can be obtained by increasing the number of the population to 200.

Table 8.4. Search results for 5.1 and 5.3

Problem Number	Test Times	x_1	x_2	f	Number of Function Evaluations
5.1	1	−7.0030	0.9940	−100.000000	167220
	2	−7.0007	0.9988	−100.000000	147380
	3*	−7.0312	0.9384	−99.999000	240020
	4	−7.0024	0.9951	−100.000000	152260
	5	−6.9970	1.0059	−100.000000	169060
	6*	−7.0157	0.9687	−99.999800	240020
	7	−6.9971	1.0060	−100.000000	123780
	8	−6.9971	1.0057	−100.000000	230020
	9	−7.0027	0.9945	−100.000000	142420
	10	−7.0020	0.9960	−100.000000	30660
5.3	1	0.997460	0.994932	−99.999994	211140
	2	0.998140	0.996484	−99.999993	176820
	3	0.999580	0.999384	−99.999995	73060
	4	1.001464	0.999384	−99.999997	115220
	5	0.999880	1.000004	−99.999994	200980
	6*	1.007760	1.015624	−99.999940	240020
	7	0.999260	0.998592	−99.999999	1780
	8	0.998280	0.996800	−99.999991	4020
	9	1.000420	1.000652	−99.999996	2980
	10	1.002920	1.005968	−99.999990	204100

*failed search

These results suggest that all the test problems are not particularly difficult for the GA in spite of the fact that the results are obtained by the standard choice of GA parameters without special tuning. This property of the GA is very attractive as a global search method. A problem of the GA is that it usually requires a large number of function evaluations. But the progress in computer technology is a great help for the GA to solve practical problems.

8.5 Control Design Benchmark Problems

In this section, the genetic inequalities solver proposed in Section 8.3 is applied to the two control benchmark problems presented in Chapter 7 where the simulated annealing inequalities solver (SAIS) has been compared with the moving boundaries process (MBP(R)).

The first problem is the design of a lead-lag compensator with three design parameters for a third order SISO plant. The aim of the design is to obtain a good step response. Four design specifications are given. The second problem is the design of a decentralised compensator for a loosely coupled

Table 8.5. Results of control design problems

Problem Number	Average Number of Function Evaluations	Minimum Number of Function Evaluations	Maximum Number of Function Evalutaions	Percent Failed Trials (%)
1	5412	1300	11460	0
2	1292	660	3220	0

two-input-two-output plant. The controller includes four design parameters. In addition, one design parameter is included in the plant. The aim of the design is to obtain a good step response with little cross coupling. Nine design specifications are given. For the detailed information on both problems, see Sections 7.5 and 7.A.

The genetic inequalities solver with the standard GA parameters shown in Table 8.2 are used. Each design parameter is represented by a string of 10 bits. Ten trials have been performed for the two problems with different initial populations. Table 5 summarises the results of the experiment. For both problems, the proposed genetic inequalities solver is successful to find an admissible solution for all the trials. But the results suggest that the number of function evaluations required for finding an admissible solution is highly dependent on the initial population. Although the Problem 1 appears simpler than the Problem 2, the proposed genetic inequalities solver requires many more function evaluations for the Problem 1 than are required for the Problem 2. This is a contrast to the SAIS results where almost the same number of maximum number of function evaluations is required for both problems. Since the search mechanisms of the two methods are completely different, it seems difficult to give a simple explanation on the difference.

Note that the results of the SAIS are obtained from 100 initial conditions. On the other hand, the results in Table 8.5 are obtained from ten trials each of which includes 100 initial conditions. The maximum number of function evaluations for the Problem 1 is close to that required for the SAIS reported in the Section 7.5. For the Problem 2, the proposed genetic inequalities solver requires less function evaluations than that required by the SAIS. The proposed genetic inequalities solver does not requie much more function evaluations than the SAIS at least for the two benchmark problems.

8.6 Conclusions

The application of MGAs to control systems design based on the method of inequalities has been discussed. The fundamental difference between the method of inequalities and the conventional multi-objective optimisation has been emphasised. It has been shown that the global search capability of

MGAs can effectively be utilised by introducing the auxiliary vector index related to the design specifications of the method of inequalities. The genetic inequalities solver using a simple MGA with the Pareto ranking has been proposed. The experimental results for the test problems and the design examples have confirmed the global search capability of the proposed solver.

It is an important future work to develop a systematic method for utilising the Pareto set when the problem is ill-posed. Although the proposed MGA utilises only the Pareto ranking to reduce the bias in the Pareto optimal set, the usefulness of the other techniques should be assessed quantitatively.

A large freedom exists to construct other MGAs for the method of inequalities. It seems possible to construct a new MGA by combining the efficient local search capability of a local search method such as the moving boundaries process with the global search capability of GAs.

References

Coello, C.A., D.A. Veldhuizen and G.B. Lamont, (2002) *Evolutionary algorithms for solving multi-objective problems*, Kluwer Academic.

Fonseca, C.M. and P.J. Fleming, (1993) Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. *Proc. of the Fifth International Conference on Genetic Algorithms*, pp. 416–423.

Fonseca, C.M. and P.J. Fleming, (1998) 'Multiobjective optimization and multiple constraint handling with evolutionary algorithms-Part I: Unified formulation.' *IEEE Trans. Systems, Man, and Cybernetics-Part A: Systems and Humans*, 28:26–47.

Goldberg, D.E., (1989) *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley.

Gtefenstette, J.J., (1986) Optimization of control parameters for genetic algorithms. *IEEE Trans. Systems, Man and Cybernetics*, 16:122-128.

Holland, J.H., (1975) *Adaptation in Natural and Artificial Sysmtems*, University of Michigan Press.

Krishnakumar, K. and D.E. Goldberg, (1992) Control system optimization using genetic algorithms. *Journal of Guidance, Control, and Dynamics*, 15:735–740.

Kristinsson, K. and G.A. Dumont, (1992) System identification and control using genetic algorithms. *IEEE Trans. Systems, Man and Cybernetics*, 22:1033–1046.

Kung, H.T., F. Luccio and F.P. Preparata, (1975) On finding the maxima of a set of vectors. *J. of the ACM*, 22:469–476.

Liu, T.K., (1997) *Application of multiobjective genetic algorithms to control systems design*, Ph. D thesis, Tohoku University.

Liu, T.K., T. Ishihara and H. Inooka, (1994) An application of genetic algorithms to control system design. *Proceedings of the first Asian Control Conference*, pp. 701–704.

Porter, B. and A.H. Jones, (1992) Genetic tuning of digital PID Controllers. *Electronics Letters*, 28(9):843–844.

Sawaragi, Y., H. Nakayama and T. Tanino, (1985) *Theory of multiobjective optimization*, Academic Press.

Schaffer, J.D., (1985) Multiple objective optimization with vector evaluated genetic algorithms. *Proc. First Int. Conf. on Genetic Algorithms and Their Applications*, pp. 93–100.

Srinivas, N. and K. Deb, (1995) Multiobjective optimization using nondominated sorting in genetic algorithm. *Evolutionary Computation*, 2:221–248.

Varsek, A., T. Urbancic and B. Filipic, (1993) Genetic algorithms in controller design and tuning. *IEEE Trans. Systems, Man and Cybernetics*, 23:1330–1339.

Whidborne, J.F., D.W. Gu and I. Postlethwaite, (1996) Algorithms for solving the method of inequalities – A comparative study. *Proceedings of American Control Conference, Seattle, WA*, pp. 3393–3397.

Zakian, V., (1996) Perspectives on the principle of matching and the method of inequalities, *Int. J. Contr.*, 65(1):147-175.

Zakian, V. and U. Al-Naib, (1973) Design of dynamical and control systems by the method of inequalities. *Proc. IEE*, 120(11):1421–1427.

Part IV

Case Studies