A convergence Control Method in the Interactive Multiple Objective Linear Programming(MOLP) Procedure

-상호작용식 다목적 선형계획과정에서의 수렴 통제 방법-

Park, Young Woong*
박 영 웅*

Abstract

Since a convergence strategy in interactive procedure is the origin of a lot of the restrictions and inconvenience of many methods, a new flexible convergence strategy is necessary for developing that is applicable for interactive MOLP as well as interactive multiple objective integer programming(MOIP). A method for computing the volume of the interval defined subsets of weighting vector space is developed using Sobol's LP_τ Generator.

Since the interactive procedure requires a number of iterations, with a fixed number of solutions presented per iteration, a convergence strategy is needed to control the rate at which the interactive procedure converges to a final solution. Thus, in this paper, a new convergency strategy is developed.

1. INTRODUCTION

No universal procedure appears to be preferable for solving all types of multiple objective linear programming(MOLP) problems. Basically, three procedures exist for solving MOLP problems: (1) vector maximization procedures [Steuer, 1986; Zeleny, 1974; Evans and Steuer, 1973a, 1973b; Philip, 1972];(2) goal programming procedures [Ignizio, 1982, 1978, 1976; Lee, 1972; Ijiri, 1965; Charnes, Cooper and Ferguson, 1955]; and (3) interactive procedures [Steuer and Gardiner, 1990; Korhonen and Wallenius, 1988; Steuer, 1986; Korhonen and Laakso, 1986; Nakayama and Sawaragi, 1984; Steuer and Choo, 1983; Zionts and Wallenius, 1983, 1976; Geoffrion, Dyer and Feinberg, 1972; Benayoun, de Montgolfier, Tergny and Larichev, 1971].

* 현대정보기술(주)

A multiple objective linear programming(MOLP) problem can be formulated as follows:

$$\begin{aligned} &\max \{c^1 x = z_1\} \\ &\qquad \vdots \\ &\max \{c^k x = z_k\} \\ &\text{s.t.} \quad x \in S \end{aligned}$$

in which $S = \{ x \in R^n \mid Ax = b,\ x \geq 0 ,\ b \in R^m \}$ in decision space. Sometimes the objective functions are formulated as "max" $\{ C_{x=z}\ x \in S \}$. Let $Z \subset R^k$ be the feasible region in criterion space where $z \in Z$ if and only if there exists an $x \in S$ such that $z = (c^1 \cdots, c^k x)$.

An additional concept used in the study is "the set of all weighting vectors":

$$\Lambda = \left\{ \lambda \in R^k \mid \lambda_i > 0,\ \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

Each objective $c'x$ is multiplied by a strictly positive scalar weight λ_i. Then, the k weighted objectives are summed to form a weighted-sums objective function. Without loss of generality, each weighting vector $\lambda \in R^k$ is normalized so that its elements sum to one. Then, Λ, denote the set of all weighting vectors. And $\overline{\Lambda} = \{\lambda \in \Lambda \mid \lambda_i \in (l_i,\ u_i)\}$ denote the subsets of weighting vector space, where l_i and u_i are specified.

Definition : Set discretization refers to the process of characterizing a continuous set by selecting a finite number of points from subset of weighting vector space.

Since the interactive procedure requires a number of iterations, with a fixed number of solutions presented per iteration, a convergence strategy is needed to control the rate at which the interactive procedure converges to a final solution.

But, the convergence strategy is the origin of a lot of the restrictions and inconveniences of many methods, as for instance, the irrevocability of the decisions [Vanderpooten, 1989]. Therefore, we need to develop a new convergence strategy to control the rate flexibly at which the interactive procedure converges to a final solution.

To develop a new convergence strategy, first of all, a method for computing the volume of interval defined subsets of weighting vector space needs to be developed. By the way, a set discretization method of weighting vector space is needed to develop a method for computing the volume of interval defined subsets of weighting vector space. Set discretization enables us to manage continuous information in a finite fashion.

Set discretization methods of weighting vector space are introduced in Section 2. In

Section 3, a method for computing the volume of interval defined subsets of weighting vector space is presented. The new convergence strategy in the interactive MOLP procedure is introduced in Section 4. Section 5 summarizes the results of this paper.

2. Set Discretization of Weighting Vector Space

The Tchebycheff procedure needs a group of well dispersed weighting vectors from progressively smaller subsets of the weighting vector space [Steuer, Chapters 11 and 14]. In this section, several set discretization methods are introduced.

2.1 Lambda/Filtering Approaches

Groups of well dispersed weighting vectors can be obtained by using the LAMBDA and FILTER codes from Steuer [1993]. The LAMBDA code randomly generates weighting vectors from weighting vector space and interval defined subsets of it defined by $\overline{\Lambda} = \{\lambda \in \Lambda \mid \lambda_i \in (l_i, u_i)\}$. The FILTER code computes groups of dispersed representatives from the set of weighting vectors inputted to it. The most current statement about the functions performed by LAMBDA and FILTER is found in Steuer [1986, Chapter 11].

2.2 Grid Point Approach

Liou [1984] developed a routine for generating weighting vectors that makes it possible to obtain sets of dispersed weighting vectors in one step, avoiding the use of FILTER to eliminate the most redundant of the weighting vectors. Liou's routine follows the strategy of superimposing a pattern of grid points on the weighting vector space.

2.3 Using Points from Sobol's Generator

Instead of using LAMBDA, Sobol's LP_τ generator [Sobol, 1976] can be employed to obtain groups of dispersed weighting vectors. Sobol's LP_τ Generator code is available in FORTRAN by Bratley and Fox [1988]. Since Sobol's LP_τ generator randomly generates vectors in the unit hypercube $[0,1]^k$, Sobol's vectors need to be converted to vectors in weighting vector space using a transformation algorithm. Three different transformations of standard random numbers (i.e., independent random variables uniformly distributed in the unit interval, $0 \leq x \leq 1$) for converting Sobol's vectors in $[0,1]^k$ to vectors in Λ are introduced in Section 3.

3. A Method for Computing the Volume of the Interval Defined Subsets of Weighting Vector Space

In this section, a method for computing the volume of interval defined subsets of weighting vector space is introduced. Although many papers [Steuer and Gardiner, 1990; Korhonen and Wallenius, 1988; Steuer, 1986; Korhonen and Laakso, 1986; Nakayama and Sawaragi, 1984; Steuer and Choo, 1983; Zionts and Wallenius, 1983, 1976; Geoffrion, Dyer and Feinberg, 1972; Benayoun, de Montgolfier, Tergny and Larichev, 1971] have used the interactive method by reducing the weighting vector space. None, except Liou [1984], have shown how to compute both the size of the volume of $\overline{\Lambda}$ when l_i and u_i are specified and the interval l_i and u_i when the size of the volume of $\overline{\Lambda}$ is specified with a given point. Liou [1984] developed the method computing only the size of the volume of $\overline{\Lambda}$ when l_i and u_i are specified.

To develop this method, Sobol's LP_τ generator [Sobol, 1976] was employed. First, by generating Sobol's vectors in $[0,1]^k$ the volume of a unit simplex, $\left\{x \in R^n \mid x \geq 0, \ \sum_{i=1}^{n} x_i \leq 1\right\}$, which is given by $\frac{1}{n!}$ is verified. Second, Sobol's vectors in $[0,1]^k$ are converted to vectors in Λ. Finally, the volume of $\overline{\Lambda}$ is computed. Using this method similarly, the interval l_i and u_i when the size of the volume of $\overline{\Lambda}$ specified with given a point can be computed.

In this section, first of all, we introduce Sobol's LP_τ Generator and verify the volume of a unit simplex by generating Sobol's test points. Then, we introduce a method for computing the volume of $\overline{\Lambda}$, with examples. The ADBASE is used to generate test problems and solve test problems. The GRIDSOBL code is employed to compute the l_i and u_i when the size of the volume of $\overline{\Lambda}$ is specified with a given point.

3.1 Sobol's LP_τ Generator

In the method for computing the volume of $\overline{\Lambda}$, Sobol's Generator is used to produce sequences of quasi random numbers between 0 and 1 (Sobol calls these LP_τ sequences). In this section, a brief summary of Sobol's LP_τ generator from Steuer and Sun (1993, pp. 6-9) is presented. Any sequence of p^h vectors obtained from LP_τ sequences is uniformly distributed over the unit hypercube $[0,1]^k$ and has low discrepancy, where discrepancy is defined as in the next paragraph.

Consider a sequence of N vectors $p^1, p^2, \ldots, p^n \in [0,1]^k$. Let $p = (p^1, p^2, \ldots, p^n)$ be any vector in $[0,1]^k$, subset $G_p = [0, p_1) \times [0, p_2) \times \cdots \times [0, p_n)$, and $S(G_p)$ be a function that yields a count of how many of the p^h in the sequences are in G_p. Then the discrepancy of the $p^1, p^2, \ldots, p^n$ is given by

$$D(p^1, p^2, \ldots p^n) = \underset{p \in [0,1]^k}{SUP} \mid S(G_p) - N p_1 p_2 \ldots p_n \mid$$

where $p_1 p_2 \ldots p_n$ is the volume of G_p according to Lieberman (1991), Sobol and Statnikov (1981) believe that LP_τ sequences possess "the best uniformity characteristic of any uniformly distributed sequence currently known."

Sobol's LP_τ Generator is described as follows. Suppose it is known beforehand that N quasi random numbers are need. Let d be the smallest integer such that $2^d \geq N+1$. Now specify a set of numbers $v_1, v_2, \ldots, v_d$ where

$$v_i = \frac{m_i}{2^i} \text{ (in binary)}$$

in which m_i can be any odd integer as long as $0 < m_i < 2^i$. Then, to generate a sequence of quasi random numbers $y^1, y^2, \ldots$use

$$y^{n+1} = y^n \oplus v_c$$

where $\oplus$ is a bit-by-bit exclusive-or operation, $\ldots b_3 b_2 b_1$ is the binary representation of n, and b_c is the rightmost zero-bit in the binary representation of n (adding a leading zero if necessary).

By way of illustrations, it is shown how ten p^k random vectors from $[0,1]^3$ would be obtained using Sobol's LP_τ Generator. This requires three LP_τ sequences of length $N=10$, one for the first components, one for the second components, the other for the third components. Because $N=10$ in each case, $d=4$ in each case. Pertaining to the first components of the p^k, let $m_1 = 1$, $m_2 = 3$, $m_3 = 5$, and $m_4 = 9$. Then the v_i pertaining to the first components are

$$\begin{aligned} v_1 &= 0.1 \\ v_2 &= 0.11 \qquad \text{(For First Component)} \\ v_3 &= 0.101 \\ v_4 &= 0.1001 \end{aligned}$$

Initializing, let $y^0=0$ and $c=1$. As a result, the $y^1, y^2, \ldots y^{10}$ quasi random numbers for the first components of the p^k are computed and listed in the fourth and fifth columns of Table 1, respectively.

Table 1 : First Components LP_τ Quasirandom Numbers

n+1	n (in binary)	c	First Component	
			$y^n \oplus v_c$ (in binary)	y^{n+1}
1	0	1	$y^0 \oplus v_1$ .0 ⊕ .1 .1	1/2
2	1	2	$y^1 \oplus v_2$ .10 ⊕ .11 .01	1/4
3	10	1	$y^2 \oplus v_1$ .01 ⊕ .10 .11	3/4
4	11	3	$y^3 \oplus v_3$ .110 ⊕ .101 .011	3/8
5	100	1	$y^4 \oplus v_1$ .011 ⊕ .100 .111	7/8
6	101	2	$y^5 \oplus v_2$ .111 ⊕ .110 .001	1/8
7	110	1	$y^6 \oplus v_1$ .001 ⊕ .100 .101	5/8
8	111	4	$y^7 \oplus v_4$ .1010 ⊕ .1001 .0011	3/16
9	1000	1	$y^8 \oplus v_1$ .0011 ⊕ .1000 .1011	11/16
10	1001	2	$y^9 \oplus v_2$ .1011 ⊕ .1100 .0111	7/16

Pertaining to the second components of the p^h, let $m_1 = 1$, $m_2 = 1$, $m_3 = 7$, and $m_4 = 9$. Then, the v_i pertaining to the second components are

$$v_1 = 0.1$$
$$v_2 = 0.01 \qquad \text{(For Second Component)}$$
$$v_3 = 0.111$$
$$v_4 = 0.1001$$

As a result, the $y^1, y^2, \ldots y^{10}$ quasi random numbers for the second components of the p^h are computed and listed in the fourth and fifth columns of Table 2, respectively.

Table 2 : Second Components LP_τ Quasirandom Numbers

n+1 (in binary)	n	c	First Component	
			$y^n \oplus v_c$ (in binary)	y^{n+1}
1	0	1	$y^0 \oplus v_1$ .0 ⊕ .1 .1	1/2
2	1	2	$y^1 \oplus v_2$ .10 ⊕ .01 .11	3/4
3	10	1	$y^2 \oplus v_1$ .11 ⊕ .10 .01	1/4
4	11	3	$y^3 \oplus v_3$ .010 ⊕ .111 .101	5/8
5	100	1	$y^4 \oplus v_1$ .101 ⊕ .100 .001	1/8
6	101	2	$y^5 \oplus v_2$ .001 ⊕ .010 .011	3/8
7	110	1	$y^6 \oplus v_1$ .011 ⊕ .100 .111	7/8
8	111	4	$y^7 \oplus v_4$ .1110 ⊕ .1001 .0111	7/16
9	1000	1	$y^8 \oplus v_1$ .0111 ⊕ .1000 .1111	15/16
10	1001	2	$y^9 \oplus v_2$ .1111 ⊕ .0100 .1011	11/16

Pertaining to the third components of the p^h, let $m_1 = 1$, $m_2 = 1$, $m_3 = 7$ and $m_4 = 9$. Then, the v_i pertaining to the third components are

$v_1 = 0.1$
$v_2 = 0.01$ (For Third Component)
$v_3 = 0.101$
$v_4 = 0.1011$

Table 3: Third Components LP_τ Quasirandom Numbers

n+1 (in binary)	n	c	First Component	
			$y^n \oplus v_c$ (in binary)	y^{n+1}
1	0	1	$y^0 \oplus v_1$ .0 ⊕ .1 .1	1/2
2	1	2	$y^1 \oplus v_2$ .10 ⊕ .01 .11	3/4
3	10	1	$y^2 \oplus v_1$ .11 ⊕ .10 .01	1/4
4	11	3	$y^3 \oplus v_3$ .010 ⊕ .101 .111	7/8
5	100	1	$y^4 \oplus v_1$ .111 ⊕ .100 .011	3/8
6	101	2	$y^5 \oplus v_2$ .011 ⊕ .010 .001	1/8
7	110	1	$y^6 \oplus v_1$ .001 ⊕ .100 .101	5/8
8	111	4	$y^7 \oplus v_4$ .1010 ⊕ .1011 .0001	1/16
9	1000	1	$y^8 \oplus v_1$ .0001 ⊕ .1000 .1001	9/16
10	1001	2	$y^9 \oplus v_2$ .1001 ⊕ .0100 .1101	13/16

As a result, the $y^1, y^2, \ldots y^{10}$ quasi random numbers for the third components of the p^h are computed and listed in the fourth and fifth columns of Table 3, respectively.

Thus

$$p^1, p^2, \ldots, p^9, p^{10} = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}, \begin{bmatrix} 1/4 \\ 3/4 \\ 3/4 \end{bmatrix}, \ldots, \begin{bmatrix} 11/16 \\ 15/16 \\ 9/16 \end{bmatrix}, \begin{bmatrix} 7/16 \\ 11/16 \\ 13/16 \end{bmatrix}$$

For the mathematical reasoning behind the steps of Sobol's LP_τ Generator, see Sobol (1967, 1976 and 1979) and Bratley and Fox (1988, pp. 89-92).

3.2 Verifying the volume of a unit simplex

In this section, by generating Sobol's vectors in $[0, 1]^k$ the volume of a unit simplex, $\left\{x \in R^n \mid x \geq 0, \ \sum_{i=1}^{n} x_i \leq 1\right\}$ which is given by $\frac{1}{n!}$ is verified. Table 4 presents the theoretical volume of a unit simplex and the test volume of a unit simplex when the number of test points is given.

If the number of test points is sufficiently increased, the difference between the theoretical volume of a unit simplex and the test volume of a unit simplex becomes very small.

Table 5 shows how the test volume of a unit simplex approximates the theoretical volume of a unit simplex when the number of test points is increased.

Table 4: Volume of a Unit Simplex

k	Test Size	Points in a unit simplex	Theoretical volume of a unit simplex	Test volume of a unit simplex	Difference
2	1000	507	1/2 (0.5)	0.507	0.007 (1.40%)
3	1000	170	1/6 (0.16667)	0.170	0.00333 (2.00%)
4	4000	163	1/24 (0.04167)	0.04075	0.00092 (2.21%)
5	16000	135	1/120 (0.008333)	0.008438	0.000105(1.26%)
6	64000	86	1/720 (0.001389)	0.001344	0.000045(3.24%)
7	256000	43	1/5040 (0.000198)	0.000168	0.00003(15.15%)
8	1024000	22	1/40320 (0.0000248)	0.0000215	0.0000033(13.31%)
9	4096000	8	1/362880(0.00000276)	0.00000195	0.00000081(29.34%)

Table 5: Volume of a Unit Simplex

k	Test Size	Points in a unit simplex	Theoretical volume of a unit simplex	Test volume of a unit simplex	Difference
2	100000	50023	1/2 (0.5)	0.50023	0.00023 (0.05%)
3	100000	16684	1/6 (0.16667)	0.16684	0.00017 (0.10%)
4	100000	4165	1/24 (0.04167)	0.04165	0.00002 (0.05%)
5	1000000	8356	1/120 (0.008333)	0.008356	0.000023 (0.28%)
6	1000000	1364	1/720 (0.001389)	0.001364	0.000025 (1.80%)
7	2000000	396	1/5040 (0.000198)	0.000198	0.000000 (0.00%)
8	20000000	502	1/40320 (0.0000248)	0.0000251	0.0000003(1.21%)
9	100000000	277	1/362880(0.00000276)	0.00000277	0.00000001(0.36%)

3.3 Converting Sobol's vectors in $[0, 1]^k$ to vectors in the weighting vector space

It is necessary to convert Sobol's vectors in $[0, 1]^k$ to vectors in weighting vector space to use it for computing the volume of the interval defined subsets of weighting vector space. Three different transformations of standard random numbers for converting Sobol's vectors in $[0, 1]^k$ to vectors in weighting vector space are now introduced.[1] The Cartesian coordinates of the point in Λ will be denoted by $\xi_1, \ldots, \xi_n$ while the standard random numbers will be $r_1, \ldots, r_n$.

Algorithm 1:

Algorithm 1 uses n standard random numbers. Then,

$$\xi_j = \frac{\ln r_j}{\ln r_1 + \cdots + \ln r_n}, \quad \text{for } 1 \le j \le n$$

Algorithm 2:

Algorithm 2 uses $n-1$ standard random numbers. Define $C_0=1$. Then,

$$\xi_j = \begin{cases} C_{j-1}\left\{1-(r_j)^{\frac{1}{n-j}}\right\} & \text{, for } 1 \le j \le n \\ \quad \textit{where, } C_j = C_{j-1} - \xi_j & \\ C_{n-1} & \text{, for } j = n \end{cases}$$

1. I wish to thank Dr. I. M. Sobol for furnishing me with the three algorithms that follow.

Algorithm 3:

Algorithm 3 uses $n-1$ standard random numbers that must be reordered: $r_1 \leq r_2 \leq \cdots \leq r_{n-1}$ Define $r_0=0$. Then

$$\xi_j = \begin{cases} r_j - r_{j-1} & \text{for } 1 \leq j \leq n-1 \\ 1 - r_{n-1} & \text{for } j = n \end{cases}$$

Comparing the performance of these three algorithms by plot charts, shows Algorithm 2 to be evenly dispersed. Thus, Algorithm 2 is used to convert Sobol's vector in $[0, 1]^k$ to vectors in Λ .

3.4 Computing the volume of $\overline{\Lambda}$

After converting Sobol's vectors in $[0, 1]^k$ to vectors in Λ, the number of vectors is generated, depending on the dimension k. The vectors that are satisfied in all $\lambda_i \in (l_i, u_i)$ are counted among these total generated vectors. Thus, their percentage of the volume of $\overline{\Lambda}$ can be computed. The volume of $\overline{\Lambda}$ can be computed by multiplying that percentage by the theoretical volume of Λ. All these procedures are coded in FORTRAN by a program called LAMSOBL.

The following examples illustrate LAMSOBL. In the first example, which has three objectives, we wish to find the volume of the interval defined subset of weighting vector space given intervals such as

$$0.2 \leq \lambda_1 \leq 0.8$$
$$0.2 \leq \lambda_2 \leq 0.8$$
$$0.2 \leq \lambda_3 \leq 0.8$$

Figure 1 shows this volume. Theoretically, the volume of the interval defined subset of weighting vector space is 16% ($\frac{0.08}{0.5}$ * 100) of the total volume. Using LAMSOBL, the volume of the interval defined subset of weighting vector space is 16.01% ($\frac{0.080055}{0.5}$ * 100) of the total volume. The difference compares to the theoretical area is 0.000055, which is small enough to be ignored. We can even further reduce this difference by increasing the number of test points.

For a more complicated example, let us find the volume of the reduced weighting vector space given intervals such as

$$0.1 \leq \lambda_1 \leq 0.5$$
$$0.4 \leq \lambda_2 \leq 0.7$$
$$0.1 \leq \lambda_3 \leq 0.5$$

Figure 2 shows this volume. Theoretically, the volume of the interval defined subset of weighting vector space is 19% ($\frac{0.095}{0.5}$ * 100)of the total area. Using LAMSOBL, the volume of the interval defined subset of weighting vector space is 19.00% ($\frac{0.095027}{0.5}$ * 100) of the total volume. The difference, compared to the theoretical volume, is 0.000027. We can also reduce this difference by increasing the number of test points.

With four objectives, it is more difficult to compute the volume of the interval defined subset of weighting vector space. With more than four objectives, it is almost impossible to be calculated by hand.

Consider another example with five objectives. We wish to find the volume of the interval defined subset of weighting vector space given intervals such as

$$0.000000 \leq \lambda_1 \leq 0.501850$$
$$0.000000 \leq \lambda_2 \leq 0.501850$$
$$0.000000 \leq \lambda_3 \leq 0.501850$$
$$0.000000 \leq \lambda_4 \leq 0.650925$$
$$0.000000 \leq \lambda_5 \leq 0.501850$$

Using LAMSOBL, the volume of the interval defined subset of weighting vector space is 45.00 % ($\frac{0.01876500}{0.04166667}$ * 100%) of the total volume.

LAMSOBL and GRIDSOBL are coded in FORTRAN by the author. The LAMSOBL code computes the volume of interval defined subsets of the weighting vector space,

given intervals of each weighting vector. The GRIDSOBL code computes the $l_i$and u_i when the size of the volume of $\overline{\Lambda}$ specified with given a point in the weighting vector space.

4. Convergence Control

The Interactive Procedure requires a number of iterations, with a fixed number of solutions presented per iteration. The Interactive Procedure samples the set of nondominated criterion vectors by solving batches of Tchebycheff programming problems whose weighting vectors come from

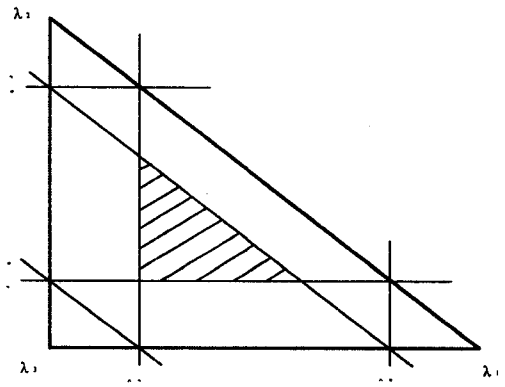

Figure 1 Volume of the Interval Defined Subset of Weighting Vector Space

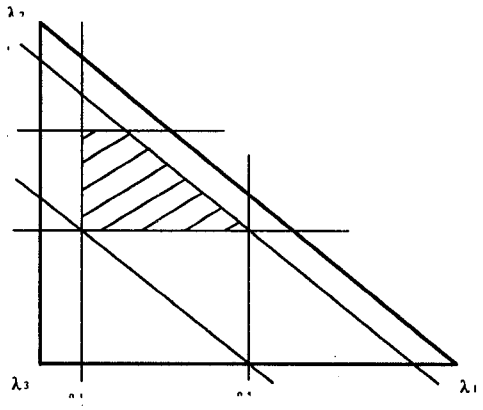

Figure 2 Volume of the Interval Defined Subset of Weighting Vector Space

$$\Lambda^{(h)} = \left\{\lambda \in R^k \mid \lambda_i \in (\, l_i^{(h)},\ u_i^{(h)}),\ \sum_{i=1}^{k} \lambda_i = 1\right\}$$

where the $(\, l_i^{(h)},\ u_i^{(h)})$bounds are iteratively tightened as the iteration count h increases. Thus, a convergence strategy to control the rate at which the interactive procedure converges to a final solution is needed to develop.

4.1 Conventional Convergence Strategy

The following conventional convergence control method is from Steuer [1986, Chapter 13]. Notations are defined as following:

P= Iteration sample size

w= Final iteration $(l_i,\ u_i)$ interval width

r= Λ-reduction factor

t= Total number of the iteration

k= The number of objectives

Parameters t and P are negotiated with the decision maker. However, based on experience, the analyst selects a value for the final interval width w between $\frac{1}{2k}$ and $\frac{3}{2k}$ which should normally suffice. Then, the analyst fits a value for the Λ-reduction factor r. It is suggested that all this be done in rough accordance with the following guideline relationships

$$\begin{array}{ccccc} & & P \gtrapprox k & & \\ & & t \approx k & & \\ 1/(2k) & \lessapprox & w & \lessapprox & 3/(2k) \\ \sqrt[k]{1/p} & \lessapprox & r & \lessapprox & \sqrt[t-1]{w} \end{array}$$

and then form the intervals of the h^{th} iteration using the Λ-reduction factor r

$$(\, l_i^{(h+1)},\ u_i^{(h+1)}) = \begin{cases} (0,\ r^h), & \text{if } \lambda_i^{(h)} - \frac{r^h}{2} \le 0 \\ (1-r^h,\ 1), & \text{if } \lambda_i^{(h)} + \frac{r^h}{2} \ge 0 \\ \left(\lambda_i^{(h)} - \frac{r^h}{2},\ \lambda_i^{(h)} + \frac{r^h}{2}\right), & \textit{otherwise} \end{cases}$$

in which r^h is r raised to the h^{th} power.

For example, if w=0.2 is selected with k=6, P=8, and t=6, the Λ-reduction factor r can be fit a value of 0.710 because

$$\sqrt[k]{1/p} = 0.707 \lessapprox r \lessapprox \sqrt[t-1]{w}$$

and then form the intervals in the h^{th} iteration using the Λ-reduction factor r=0.710:

When h=0, $(l_i^{(1)}, u_i^{(1)}) = (0,1)$ for all i.

When h=1,

$$(l_i^{(1)}, u_i^{(1)}) = \begin{cases} (0, 0.71), & \text{if } \lambda_i - 0.355 \le 0 \\ (1 - 0.71, 1), & \text{if } \lambda_i + 0.355 \le 0 \\ (\lambda_i - 0.355, \lambda_i + 0.355), & \textit{otherwise} \end{cases}$$

From the guideline relationships, it has found that: (1) the larger P, the fewer iterations are required; (2) the smaller r, the faster we reduce weighting vector space; (3) the narrower w, the more pressure there is for P and t to be large.

4.2 New Convergency Strategy

The most important purpose in controlling convergence is to provide the potential optimal solution to the Decision Maker(DM) flexibly. If the number of iterations is too large, it take more time to end up the procedure. If the number of the iteration is too small, the potential optimal solution might be excluded since the (l_i, u_i) bounds of the reduced weighting vector space around the selected point is tightened to end up the procedure fast.

With the new convergence strategy, the convergence rate can be controlled during interacting with DM. (There is no more restriction for P, t, and no more calculation for the Λ-reduction factor r)

GRIDSOBL enables to develop new convergence strategy. For example, when the DM wants to use t=4, and asks to reduce the weighting vector space proportionally such as 50%, 25%, 12.5%, and 6.25% around points which the DM has chosen in each iteration, GRIDSOBL can find the (l_i, u_i) bounds of the subset of weighting vector space around any given point. Furthermore, when the DM asks to reduce the size of the nondominated set, rather than the volume of the subset of weighting vector space, proportionally such as 50%, 25%, 12.5%, and 6.25% around points which the DM has chosen in each iteration, the resulting curves in Section 4 used to determine the best strategy for convergence control. Of course, the number of iteration can be changed while interacting with DM is occurred. In any iteration, if DM feels the selected solution is very close to the DM's desired solution, the (l_i, u_i) bounds of the reduced

weighting vector space around the selected point can be more tightened to conclude the procedure earlier. Reversely, if DM feels the selected solution is very far away from the DM's desired solution, the (l_i, u_i) bounds of the reduced weighting vector space around the selected point can be less tightened to conclude the procedure later.

To demonstrate how the new convergency strategy works, an example situation with 3 objectives is presented. Consider that the DM asks to reduce the size of the nondominated set proportionally such as 70%, 50%, 30%, and 10% around points which have chosen in each iteration. First of all, find the percentages of the volume of the reduced weighting vector space corresponding to the percentages of the number of the nondominated set. Using the resulting curve, the percentages of the volume of the reduced weighting vector space corresponding to are 50%, 30, 15%, and 5%, respectively. Then, using GRIDSOBL, find the (l_i, u_i) bounds of the reduced weighting vector space around a given point.

Assuming, at the first iteration, the weighting vector of the solution which the DM has selected is (0.201, 0.623, 0.176), the (l_i, u_i) bounds of the 70% interval defined subset of weighting vector space (from now, we use 'the 70% interval defined subset' instead of 'the 70% interval defined subset of weighting vector space' for convenience) around given point are as following:

$$0.000 \leq \lambda_1 \leq 0.701$$
$$0.123 \leq \lambda_2 \leq 1.000$$
$$0.000 \leq \lambda_3 \leq 0.676$$

The figure 3 shows the 70% interval defined subset around a point (0.201, 0.623, 0.176) with the bounds.

At the second iteration, assuming the weighting vector of the solution which the DM has selected is (0.425, 0.304, 0.271), the (l_i, u_i) bounds of the 50% interval defined subset around given point are as following:

$$0.133 \leq \lambda_1 \leq 0.716$$
$$0.012 \leq \lambda_2 \leq 0.595$$
$$0.000 \leq \lambda_3 \leq 0.562$$

The figure 4 shows the 50% interval defined subset around point (0.425, 0.304, 0.271)

with the bounds.

At the third iteration, assuming the weighting vector of the solution which the decision maker has selected is (0.347, 0.316, 0.337), the (l_i, u_i) bounds of the 30% interval defined subset around given point are as following:

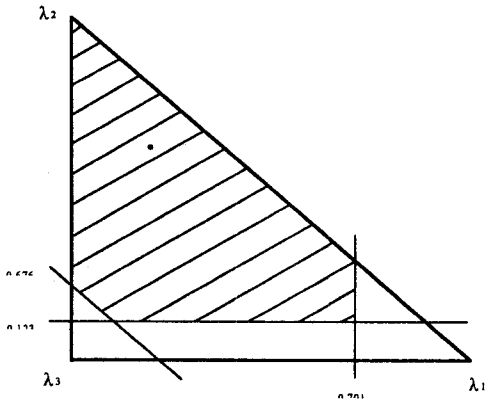

Figure 3 The (l_i, u_i) Bounds of the 70% Interval Defined Subset

$$0.124 \leq \lambda_1 \leq 0.570$$
$$0.093 \leq \lambda_2 \leq 0.539$$
$$0.114 \leq \lambda_3 \leq 0.560$$

The figure 5 shows the 30% interval defined subset around point (0.347, 0.316, 0.337) with the bounds.

At the final iteration, assuming the weighting vector of the solution which the decision maker has selected is (0.385, 0.314, 0.301), the (l_i, u_i) bounds of the 10% interval defined subset around given point are as following:

$$0.257 \leq \lambda_1 \leq 0.513$$
$$0.186 \leq \lambda_2 \leq 0.442$$
$$0.173 \leq \lambda_3 \leq 0.429$$

The figure 6 shows the 10% interval defined subset around point (0.385, 0.314, 0.301) with the bounds.

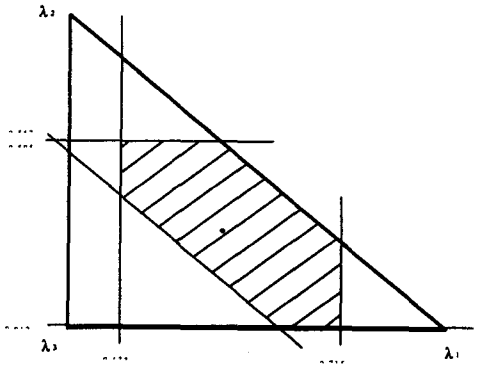

Figure 4 The (l_i, u_i) Bounds of the 50% Interval Defined Subset

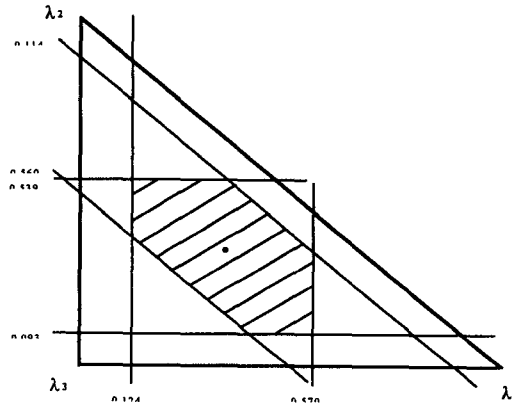

Figure 5 The (l_i, u_i) Bounds of the 30% Interval Defined Subs

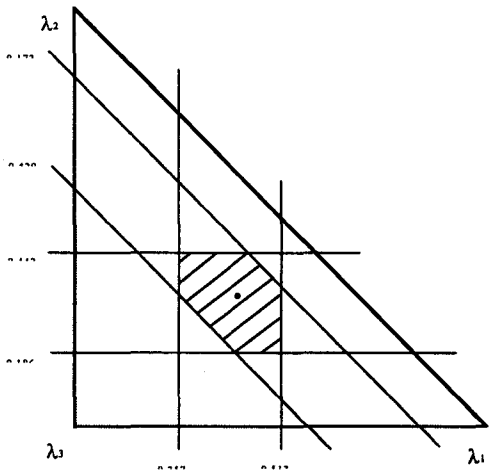

Figure 6 The (l_i, u_i) Bounds of the 10% Interval Defined Subset

5. Conclusion

Since a convergence strategy in interactive procedure is the origin of a lot of the restrictions and inconvenience of many methods, a new flexible convergence strategy was developed that is applicable for interactive MOLP. Thus, a method for computing the volume of the interval defined subsets of weighting vector space is developed using Sobol's LP_τ Generator[Sobol,1976]. With the new convergence strategy, the convergence rate can be controlled during interacting with decision maker flexibly. That means that this is no more restriction for P, t, and no more calculation for the Λ-reduction factor r.

Reference

[1]. Benayoun, R., J. de Montgolfier, J. Tergny, and O. Larichev (1971). "Linear Programming with Multiple Objective Functions: Step Method (STEM)," Mathematical Programming, Vol. 1, No. 3, pp. 366-375.

[2]. Bratley, P., and B. L. Fox (1988). "Algorithm 659 Implementing Sobol's Quasirandom Sequence Generator," ACM Transactions on Mathematical Software, Vol. 14, No. 1, pp. 88-100.

[3]. Charnes, A., W. W. Cooper, and R. O. Ferguson (1955). "Optimal Estimation of Executive Compensation by Linear Programming," Management Science, Vol. 1, No. 2, pp. 138-151.

[4]. Evans, J. P. and R. E. Steuer (1973a). "A Revised Simplex Method for Multiple Objective Programs," Mathematical Programming, Vol. 5, No. 1, pp. 54-72.

[5]. Evans, J. P. and R. E. Steuer (1973b). "Generating Efficient Extreme Points in Linear Multiple Objective Programming: Two Algorithms and Computing Experience," in J. L. Cochrane and M. Zeleny (eds.), Mutiple Criteria Decision Making, University of South Carolina Press, Columbia, South Carolina, pp. 349-365.

[6]. Geoffrion, A. M., J. S. Dyer, and A. Feinberg (1972). "An Interactive Approach for Multicriteron Optimization, with an Application to the Operation of an Academic Department," Management Science, Vol. 19, No. 4, pp. 357-368.

[7]. Ignizio, J. P. (1976). Goal Programming and Extensions, D. C. Heath, Massachusetts.

[8]. Ignizio, J. P. (1978). "A Review of Goal Programming: A Tool for Multiobjective Analysis," Journal of the Operational Research Society, Vol. 29, No. 11, pp. 1109-1119.

[9]. Ignizio, J. P. (1982). Linear Programming in Single - Multiple- Objective Systems, Prentice-Hall, New Jersey.

[10]. Ijiri, Y. (1965). Management Goals and Accounting for Control, Randy-McNally, Chicago.

[11]. Korhonen, P., and J. Laakso (1986). "A Visual Interactive Method for Solving the Multiple Criteria Program," European Journal of Operational Research, Vol. 24, No. 2, pp. 277-287.

[12]. Korhonen, P., and J. Wallenius (1988). "A Pareto Race," Naval Research Logistics, Vol. 35, No. 6, pp. 615-623.

[13]. Lee, S. M. (1972). Goal Programming for Decision Analysis, Auerbach Publishers, Philadelphia.

[14]. Lieberman, E. R. (1991). Multi-Objective Programming in the USSR, Academic Press, New York.

[15]. Liou, F. H. (1984). "A Routine for Generating Grid Point Defined Weighting Vectors," Masters Thesis, Department of Management Science & Information Technology, University of Georgia, Athens, Georgia.

[16]. Nakayama, H., and Y. Sawaragi (1984). "Satisficing Trade-off Method for Multiobjective Programming," Lecture Notes in Economics and Mathematical Systems, Vol. 229, pp. 113-122.

[17]. Philip, J. (1972). "Algorithms for the Vector Maximization Problem," Mathematical Programming, Vol. 2, No. 2, pp. 207-229.

[18]. Sobol, I. M. (1976). "Uniformly Distributed Sequences with an Additional Uniform Property," USSR Computational Mathematics and Mathematical Physics, Vol. 16, pp. 236-242.

[19]. Sobol, I. M., and R. B. Statnikov (1981). Vybor Optimal 'hykh Parametrov v Zadachakh co Mnogimi Kriteriyami (Choosing Optimal Parameters in Problems with Many Criteria), Nauka, Moscow.

[20]. Steuer, R. E. (1986). Multiple Criteria Optimization: Theory, Computation, and Application, John Wiley & Sons, New York.

[21]. Steuer, R. E. (1993). "The Tchebycheff, Aspiration Criterion Vector, and Combined Procedures of Interactive Multiple Objective Programming," Avtomatika, Vol.4.

[22]. Steuer, R. E., and L. R. Gardiner (1990). "Interactive Multiple Objective Programming: Concepts, Current Status, and Future Directions," in A. Carlos, and B. Costa (eds.), Readings in Multiple Criteria Decision Aid, pp. 413-444.

[23]. Steuer, R. E., and M. Sun (1993). "The Parameter Space Investigation Method of Multiple Objective Nonlinear Programming: A Computational Investigation," Faculty of Management Science, University of Georgia, Athens, Georgia.

[24]. Vanderpooten, D. [1989]. "Description and Analysis of Some Representative Interactive Multicriteria Procedures," Mathematical Computational Modelling, Vol. 12, No. 10/11, pp. 1221-1238.

[25]. Zeleny, M. (1974). "Linear Multiobjective Programming," Lecture Notes in Economics and Mathematical Systems, Vol. 95.

[26]. Zionts, S., and J. Wallenius (1976). "An Interactive Programming Method for Solving the Multiple Criteria Problem," Management Science, Vol. 22, No. 6, pp. 652-663.

[27]. Zionts, S., and J. Wallenius (1983). "An Interactive Multiple Objective Linear Programming Method for a Class of Underlying Nonlinear Utility Functions," Management Science, Vol. 29, No. 5, pp. 519-529.