Cutting Plane Methods in Decision Analysis

Xiaosong Ding[1] and **Faiz Al-Khayyal**

xiaosong.ding@gmail.com, Department of Information Technology and Media, Mid-Sweden University, SE-851 70, Sundsvall, Sweden.

faiz.alkhayyal@isye.gatech.edu, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205, USA.

Abstract

Several computational decision analysis approaches have been developed over a number of years for solving decision problems when vague and numerically imprecise information prevails. However, the evaluation phases in the DELTA method and similar methods often give rise to special bilinear programming problems, which are time-consuming to solve in an interactive environment with general nonlinear programming solvers. This paper proposes a linear programming based global optimization algorithm that combines the cutting plane method together with the lower bound information for solving this type of problems. The central theme is to identify the global optimum as early as possible in order to save additional computational efforts.

[1] Corresponding author

1 Introduction

With the rapid development of graphical user interfaces, it is possible to bring the use of sophisticated computational techniques for decision analysis to a broader group of users, and many decision analytic tools have emerged. However, most of them consist of some straightforward set of rules applied to precise numerical estimates of probabilities and values no matter how unsure a decision maker is of his or her estimates. The requirement to provide numerically precise information in such models has often been considered unrealistic in real-life decision situations. Besides, sensitivity analysis is often not easy to carry out in more than a few dimensions at a time because of precise figures. When a decision maker is faced with a decision problem that could not be directly judged by his or her empirical experience, or according to available historical data, a module allowing imprecision is obviously of great value.

A number of techniques allowing imprecise statements have been suggested, but they are concentrated more on representation and less on evaluation. In spite of several years of intense activities, only a few decision analytic tools, for example, ARIADNE, *Decide*IT and Winpre, can evaluate imprecise estimates. Among these tools, the DELTA method for decision analysis, described in [4, 5, 6, 7, 11], is an approach towards analyzing decision problems containing imprecise information, represented by intervals and relations. It has been implemented in the Decision Analysis System (DAS) *Decide*IT [8], and has been used in various real-life contexts; e.g., [12]. Due to the introduction of interval and qualitative estimates, the relaxation of classical decision theory in this respect gives rise to special Bilinear Programming (BLP) problems, whose study is a sub-field of Nonlinear Programming (NLP).

In Fig. 1 below, a decision tree is presented,

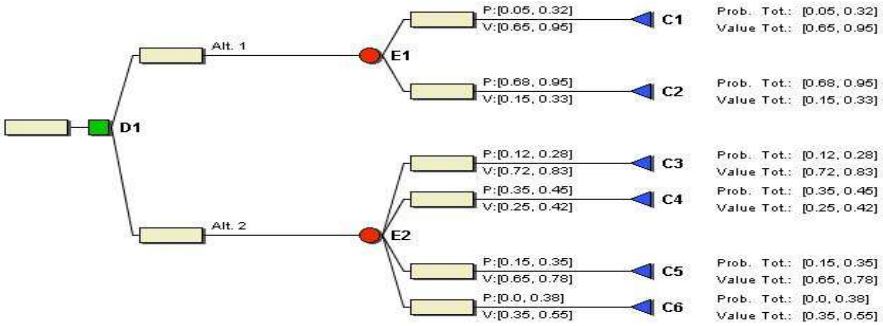

Figure 1: A Decision Tree

where $D1$ is a decision node, $E1$ and $E2$ are probability nodes, representing indeterminism, with associated probability distributions. The leaves are consequence

nodes with convex sets of associated value or utility functions. In DELTA, a *decision frame* represents a decision problem of this type. The idea behind such a frame is to collect all information necessary for the model in one structure. This structure is then filled in with user statements represented as linear inequalities. User statements can be range constraints, core intervals, or comparative statements. In a decision frame, a consequence c_i is denoted by a variable v_i and the user statements can be of the following forms for the numbers a_1, a_2, b_1, b_2, d_1, and d_2.

- Range: v_i is definitely between a_1 and a_2;
- Core interval: v_i is likely to fall between b_1 and b_2; and
- Comparison: v_i is larger than another variable, v_j, by an amount between d_1 and d_2.

All value statements are translated and collected together in a *value base* (*V-base*). On the other side, with the usual normalization constraints $\sum_{i\in I} p_i = 1$ and $\sum_{j\in J} p_j = 1$, where I and J are index sets labelling the consequences of two alternatives, all probability statements in a decision problem are translated into a *probability base* (*P-base*). The structure $\prec P, V \succ$ is referred to as a *decision frame*.

Given a *decision frame* $\prec P, V \succ$, the primary evaluation rules in DELTA are based on pair-wise comparisons using a generalization from the principle of maximizing the expected utility. A typical issue in this context is to maximize an expression, such as $\max(\sum_{i\in I} p_i v_i - \sum_{j\in J} p_j v_j)$, which is subject to a constraint set defined by a decision frame. Similar evaluation rules apply in other analysis methods.

More generally, comparative decision rules in computational decision analysis are variations of the following form:

$$\tfrac{1}{2}[\min(\textstyle\sum_{i\in I} p_i v_i - \sum_{j\in J} p_j v_j) + \max(\sum_{i\in I} p_i v_i - \sum_{j\in J} p_j v_j)] \tag{1}$$

In a typical decision situation, imprecise estimates occur in both *P-* and *V-bases*, which results in a special BLP problem. It should be noted that in (1), the corresponding maximization problem is readily solved by minimizing the negation of a minimization problem. Therefore, without losing any generality, throughout the rest of this paper, the focus will be centered on developing a rapid BLP algorithm for solving:

$$\begin{array}{ll} \min & f(p,v) = \sum_{i\in I} p_i v_i - \sum_{j\in J} p_j v_j, \\ \text{s.t.} & \left[\begin{array}{c} L_P \\ L_V \end{array}\right] \le \left[\begin{array}{cc} C_P & 0 \\ 0 & C_V \end{array}\right] \cdot \left[\begin{array}{c} P \\ V \end{array}\right] \le \left[\begin{array}{c} U_P \\ U_V \end{array}\right] \end{array} \tag{2}$$

where L_P, C_P and U_P represent the lower bounds, constraint coefficients and upper bounds in the *P-base*, respectively; $P^t = (p_I^t, p_J^t)$ represents the variables in the *P-base*; and by analogy, these definitions also exist in the *V-base*.

The next section will describe the optimization background employed in our procedure, which is followed by developing a BLP algorithm that combines a cutting plane method in a local optimization phase with a lower bounding method in a global optimization phase. Then a numerical example is solved to illustrate the

elements of the BLP algorithm. Computational results on more than four-hundred randomly generated decision analysis problems indicate that the approach is very effective for solving practical sized decision analysis problems in real time on a laptop architecture computer. Two possible directions for future research on our approach are suggested in the final section.

2 Optimization

Consider the standard disjoint BLP problem:

$$\begin{array}{ll} \min & f(x,y) = c^t x + d^t y + x^t C y, \\ \text{s.t.} & x \in X_0 = \{x \in R^m : A_1 x \leq b_1, x \geq 0\}, \\ & y \in Y_0 = \{y \in R^n : A_2 y \leq b_2, x \geq 0\} \end{array} \tag{3}$$

where $c \in R^m$ and $d \in R^n$ are linear parts for x and y, respectively, $C \in R^{m \times n}$, and X_0 and Y_0 are bounded polyhedral sets. In terms of (2), both c and d are zero vectors, and C is always an indefinite square matrix with only $+1$ or -1 diagonal elements. For example, in Fig. 1:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

The disjoint BLP (3) is one type of general Quadratic Programming (QP) problems with a symmetric indefinite quadratic form matrix. The special cases (2) that arise in computational decision analysis have the added property that the bilinear form matrix C is indefinite. In both cases, the problem is non-convex and global optimization strategies are required to find the absolute minimum objective value, which is called the *global minimum*, and its corresponding solution point, which is called the *global minimizer*. It should be noted that we distinguish between the minimal objective function value and the corresponding point at which it is achieved as *minimum* and *minimizer*, respectively. A general framework for many global optimization strategies is summarized in [18] as:

> "Actually all methods for global optimization consist of two phases: a global phase, aimed at thorough exploration of the feasible region or subsets of the feasible region where it is known the global optimum will be found, and a local phase aimed at locally improving the approximation to some local optima. Often these two phases are blended into the same algorithm, which automatically switches between exploration and refinement."

The procedure presented herein captures the spirit of this general framework by proposing a refinement to an existing algorithm for the local phase blended with a formulation for the global phase to produce a global optimization algorithm that finds either an exact global minimizer or an epsilon-global minimizer with a specified tolerance.

An important property of (3) to observe is that even though $f(x, y)$ can be shown to be not quasi-concave, an optimal solution (x^*, y^*) exists at an extreme point of $X_0 \times Y_0$, [2]; i.e., x^* is an extreme point of X_0 and y^* is an extreme point of Y_0. However, this property is lost in the jointly constrained case such as:

$$\begin{array}{ll} \min & f(x,y) = c^t x + d^t y + x^t C y, \\ \text{s.t.} & x, y \in \{x \in r^m, y \in r^n : A_1 x + A_2 y \leq b, x \geq 0, y \geq 0\} \end{array} \tag{4}$$

To solve a jointly constrained BLP problem with a non-extremal boundary point optimum poses the greatest computational difficulty. However, for those BLP problems exhibiting extreme point optimal solutions, it is relatively easy to solve. Some computational results are reported in [20, 21].

The other important property of (3) is that any cuts involving variables associated with both X_0 and Y_0 sets will destroy their special structures. Problems do exist where one of the sets has special structure that can be exploited by efficient algorithms which can be used to solve sub-problems in the solution procedure, [23]. Accordingly, we prefer developing linear cuts within only one polyhedron.

3 Local Optimization

The local optimization phase aims at locating a local optimum. Any local optimization technique for finding *Karush-Kuhn-Tucker* (KKT) solutions of quadratic programs, such as Wolfe's simplex method or an interior point method, can accomplish this task. Nevertheless, the structure of the disjoint BLP problem (3) itself suggests a Linear Programming (LP) based vertex following algorithm, which is very convenient and efficient, [16]. The approach consists in starting with an arbitrary fixed $x \in X_0$, and solving the related LP problem with y as the unknown. The solution, y, is then used to solve another LP problem with x as the unknown. This in turn yields a new solution for x. The procedure is repeated until a pair of values $(\overline{x}, \overline{y})$ is found that solves both LP problems. It has been proved that the resulting solution is a KKT point.

DEFINITION 1: Consider $P : \min f(x)$ subject to $x \in S$, where S is a compact polyhedral set and f is non-convex. A *local star minimizer* of P is defined as a point $\overline{x}$ such that $f(\overline{x}) \leq f(x)$ for each $x \in N(\overline{x})$, where $N(\overline{x})$ denotes the adjacent extreme points to $\overline{x}$.

Extending the definition of $N(\overline{x})$ into the disjoint BLP (3), an extreme point is

adjacent to $(\overline{x}, \overline{y})$ if and only if it is of the form $(x^i, \overline{y})$ or $(\overline{x}, y^i)$ where $x^i \in N(\overline{x})$ and $y^i \in N(\overline{y})$.

DEFINITION 2: An extreme point is called a *pseudo-global minimizer* if $f(\overline{x}, \overline{y}) \leq f(x, y)$ for each $x \in B_\delta(\overline{x}) \cap X_0$ and for each $y \in Y_0$, where B_δ is a δ neighborhood around $\overline{x}$.

Intuitively, a pseudo-global minimizer is an extreme point that satisfies the KKT conditions, has no descent directions within its neighborhood, and acts as a local minimizer in x-space and a global minimizer in y-space. In order to obtain a pseudo-global minimizer, we closely follow the algorithm described in [26].

ALGORITHM 1:

1. Find a feasible extreme point x^1 in X_0.
2. [a] Solve: $\min\{f(x^1, y) | y \in Y_0\}$, to yield an optimal y^1.
   [b] Solve: $\min\{f(x, y^1) | x \in X_0\}$, to yield an optimal x^2.
   Repeat until the procedure converges to a local star minimizer $(\overline{x}, \overline{y})$.
3. Let $x^1, \ldots, x^m$ be the adjacent extreme points of $\overline{x}$.
   Solve: $\min\{f(x^i, y) | y \in Y_0\}$, to yield solutions $y^1, \ldots, y^m$.
4. If $f(\overline{x}, \overline{y}) \leq f(x^i, y^i)$ for all i, terminate with $(\overline{x}, \overline{y})$ as a pseudo-global minimizer.
5. Choose one $f(x^r, y^r) \leq f(\overline{x}, \overline{y})$ and go back to 2[b] with $y^1 = y^r$.

The performance to locate a KKT point in the DELTA method has been reported in [10]. Based on computational observations, a KKT point is found within four iterations, on average. However, checking its adjacent extreme points is relatively time-consuming, especially when we have to return to step 2[b] from step 5.

4 Global Optimization

Given a pseudo-global optimizer, a linear cut needs to be generated within only one polyhedron. The cutting plane techniques for bilinear programs were inspired by similar methods for concave problems, [19, 25]. One of the first such procedures was proposed in [16] to delete local vertex solutions by using Ritter's cut [19]. Another cutting plane approach was developed in [13] by using Tuy's cut [25]. The latter used LP duality theory to reformulate the BLP problem as an equivalent concave minimization problem with an implicitly defined objective function. The polar cuts of [3] were applied in [26] to BLP, where it was proved that the polar cuts uniformly dominate other similar cuts. This approach was further strengthened in [22] by employing negative edge extension polar cuts and disjunctive face cuts, whereupon finite convergence to an exact solution could be guaranteed. In [15], it has been pointed out that [22] might be the most efficient approach for handling

BLP problems. Accordingly, in this paper, we employ polar cuts to cut off local vertex solutions.

Let $\overline{x}$ be an extreme point of X_0 and let $p_j, j \in J$, be the m nonbasic variables at $\overline{x}$, where J is the index set for the nonbasic variables. Denoting by $\overline{e}^j$ the columns of the simplex tableau in extended form, the m-vector x can be written as:

$$x = \overline{x} - \textstyle\sum_{j \in J} \overline{e}^j p_j.$$

Barring the degenerate case, X_0 has precisely m distinct edges incident to $\overline{x}$ and each half line

$$\xi^j = \{x : x = \overline{x} - \overline{e}^j \lambda_j, \lambda_j \geq 0\}, j \in J \tag{5}$$

contains exactly one such edge, [3].

DEFINITION 3: The *generalized reverse polar* of Y_0 for a given scalar α is given by $Y_0(\alpha) = \{x : f(x, y) \geq \alpha\}$ for all $y \in Y_0$.

Following [22, 26], let $(\overline{x}, \overline{y})$ be a pseudo-global minimizer, let the rays ξ^j be defined as in (5), let α be the current best objective value of $f(x, y)$, and let $\overline{\lambda}_j$ be defined by:

$$\overline{\lambda}_j = \begin{cases} \max\{\lambda_j : f(\overline{x} - e^j \lambda_j, y) \geq \alpha \text{ for all } y \in Y_0\} \text{ if } \xi^j \not\subset Y_0(\alpha), \\ -\max\{\lambda_j : f(\overline{x} + e^j \lambda_j, y) \geq \alpha \text{ for some } y \in Y_0\} \text{ if } \xi^j \subset Y_0(\alpha) \end{cases} \tag{6}$$

Then the inequality

$$\textstyle\sum_{j \in J} p_j / \overline{\lambda}_j \geq 1 \tag{7}$$

is a valid cutting plane. The second line in (6) is referred to as the negative extension polar cuts. Inequality (7) is a valid cut in the sense that firstly, it does not contain the current pseudo-global optimum; and secondly, it contains all the candidates $x \in X_0$ for which $\min\{f(x, y) | y \in Y_0\} < \alpha$.

The cutting plane method searches for the global optimum by exhausting all possibilities within one of the two bounded polyhedra. Although ALGORITHM 1 always generates a feasible solution, we have no way of knowing if we have found the global solution until we have cut off all of the pseudo-global optima. Consequently, the key idea is to obtain some lower bound information concerning the global solution. Then at least we can tell how close the current best solution is to the global optimality before an exhaustive search. We employ the convex and concave envelopes of $x_i y_i$ developed in [1, 2] to obtain such a lower bound. Consider the inner product $x^t y$ over the compact hyper-rectangle $\Omega = \{(x, y) : l \leq x \leq L, m \leq y \leq M\}$. Define $\Omega_i = \{(x_i, y_i) : l_i \leq x_i \leq L_i, m_i \leq y_i \leq M_i\}$ so that $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$. The convex and concave envelopes of $x_i y_i$ over Ω_i are defined as:

$$\begin{aligned} Vex_{\Omega_i}[x_i y_i] &= \max\{m_i x_i + l_i y_i - l_i m_i, M_i x_i + L_i y_i - L_i M_i\}, \\ Cav_{\Omega_i}[x_i y_i] &= \min\{M_i x_i + l_i y_i - l_i M_i, m_i x_i + L_i y_i - L_i m_i\} \end{aligned} \tag{8}$$

Accordingly, in (2), we can calculate the convex envelopes for $C_{ii} = 1$ and concave envelopes for $C_{ii} = -1$. Then we can say:

$$\begin{array}{ll} \min & f(p,v) = \sum_{i\in\{i:C_{ii}=1\}} Vex_{\Omega_i}[p_i v_i] - \sum_{i\in\{i:C_{ii}=-1\}} Cav_{\Omega_i}[p_i v_i], \\ \text{s.t.} & \left[\begin{array}{c} L_P \\ L_V \end{array}\right] \leq \left[\begin{array}{cc} C_P & 0 \\ 0 & C_V \end{array}\right] \cdot \left[\begin{array}{c} P \\ V \end{array}\right] \leq \left[\begin{array}{c} U_P \\ U_V \end{array}\right] \end{array} \tag{9}$$

is an underestimating problem for (2), whose solution yields a lower bound on the global minimum of our bilinear decision problem. In (9), the rectangles Ω_i define the bounds on p_i and v_i which are both readily available.

Since solving (9) yields a lower bound on the optimal objective value of (2), if the algorithm cuts off a pseudo-global optimizer with a polar cut and proceeds to search for another one in the smaller set, then we can solve the underestimating problem with the convex and concave envelopes computed over the smaller region to obtain a tighter bound on all global solutions in the reduced feasible set. If the algorithm cuts off the global solution and the objective of the underestimating problem is higher than the current best objective value, then we can stop and use the current best solution as the global solution. If there are many global optimal solution points, the objective of the underestimating problem will be smaller than the global value until all global solution points have been cut off.

If the feasible region has not been exhausted and the underestimating problem is still giving optimal values lower than the current best solution, then it is always possible to stop the search procedure early with a known feasible point and a lower bound on the global optimum. In that case, an error bound will be available to show how far we are away from global optimality in the worst scenario.

Denote by X_0 the original feasible region or its subset obtained after the introduction of generated polar cuts. The global optimization algorithm for (2) can be summarized as follows:

ALGORITHM 2:

1. Let the best objective value, *obj*, be a large positive number, and let an epsilon tolerance, ϵ, be a prescribed small number.
2. Calculate the lower bound, *bound*, for X_0^i by using (9).
3. If $bound > obj$ or $|bound - obj| \leq \epsilon$, or the unexplored feasible region X_0^i at stage i is empty, terminate with *obj* as the global minimum.
4. Find a pseudo-global minimizer by using ALGORITHM 1, and update *obj* if necessary.
5. If $|bound - obj| \leq \epsilon$, terminate with *obj* as the global minimum.
6. Solve m LP problems by using (6) to obtain $\overline{\lambda}_j$, and generate the polar cut by using (7), and introduce it into X_0^i.
7. increase i to $i + 1$, go back to 2.

We do not employ the extreme face finding routine and disjunctive face cuts as in [22] because they are relatively expensive to calculate. Instead, we take advantage of ALGORITHM 1 to locate a pseudo-global optimizer. As pointed out in the last section, ALGORITHM 1 is also time-consuming if it proves necessary to frequently locate a new local star minimizer. Therefore, it is difficult to determine which procedure is more efficient from a computational viewpoint. ALGORITHM 2 simply adds the lower bound computation in order to more quickly identify when a global optimizer has been found.

5 Numerical Example

In this section, a numerical example will be used to illustrate ALGORITHM 2. The data for this experiment is randomly generated by Matlab to simulate a real-life decision situation, [9].

Suppose now we have a decision situation consisting of two alternatives with six consequences in each alternative. Correspondingly, $P = (p_{11}, \ldots, p_{16}, p_{21}, \ldots, p_{26})^t$ and $V = (v_{11}, \ldots, v_{16}, v_{21}, \ldots, v_{26})^t$. The matrices C_P and C_V are given below:

$$C_P = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$C_V = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

In C_P, each entry represents the coefficient of each variable. For example, the first and second lines are normalization requirements, $\sum_{i \in I} p_i$ and $\sum_{j \in J} p_j$, with respect to each alternative, whereas the third line means $p_{21} - p_{25}$, and etc. The contents in C_V are explained analogously. In practice, the value base does not contain compound value statements since they lack semantic content.

The bounds L_P, L_V, U_P and U_V are listed in Table 1. In addition, each variable in the *P-base* is restricted within the interval $[0, 1]$, and each variable in the *V-base* is in:

$$\begin{bmatrix} 0.151 \\ 0.210 \\ 0.080 \\ 0.277 \\ 0.740 \\ 0.340 \end{bmatrix} \leq \begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \\ v_{14} \\ v_{15} \\ v_{16} \end{bmatrix} \leq \begin{bmatrix} 0.866 \\ 0.592 \\ 0.174 \\ 0.541 \\ 0.791 \\ 0.593 \end{bmatrix}, \quad \begin{bmatrix} 0.083 \\ 0.018 \\ 0.780 \\ 0.156 \\ 0.020 \\ 0.057 \end{bmatrix} \leq \begin{bmatrix} v_{21} \\ v_{22} \\ v_{23} \\ v_{24} \\ v_{25} \\ v_{26} \end{bmatrix} \leq \begin{bmatrix} 0.480 \\ 0.303 \\ 0.848 \\ 0.354 \\ 0.152 \\ 0.637 \end{bmatrix}.$$

C_P :	Rows	L_P	U_P	C_V :	Rows	L_V	U_V
	1	1.000	1.000		1	-1.692	1.692
	2	1.000	1.000		2	-0.576	0.061
	3	-0.437	0.182		3	-0.141	0.510
	4	-0.296	0.394		4	-1.182	-0.302
	5	-0.131	0.313		5	-0.576	0.297
	6	-0.543	0.376		6	-0.523	0.456
	7	0.773	1.692				

Table 1: Data

If we simply use the cutting plane method, the feasible region of the *P-base* will be exhausted in four cuts. Nevertheless, the second lower bound information in Table 2 is enough to guarantee the global optimum; i.e., the stopping rule, $bound > obj$, is satisfied in ALGORITHM 2, step 3. Therefore, ALGORITHM 2 terminates in one iteration; thus, saving the additional computational effort of performing three more iterations.

Iteration	Objective Value	Lower Bound
1	-0.60742032702968	-0.61148735388339
2	-0.54069572061830	-0.54476274747200
3	-0.51501142176621	-0.51907844861991
4	-0.46983913868983	-0.47390616554353

Table 2: An Numerical Example

6 Computational Experience

We launched a number of simulated instances to test ALGORITHM 2. The experiment is performed on a personal computer with Windows 2000, Matlab 6.5 & Tomlab [24], Pentium-III 1000 MHz CPU and 512MB memory. The commercial LP

solver is SQOPT [14] from Systems Optimization Laboratory, Stanford University. In total, we tested 400 instances consisting of 10 groups, each of them including 40 data sets with the consequences from 11 to 50. If an instance has N consequences, the corresponding BLP problem contains 4N variables and around 2N constraints. Consequently, this experiment is trying to solve disjoint BLP problems sized up to 200 variables. Detailed numerical results are shown in Table 3.

C	P_V	V_V	P_C	V_C	$Time$	C	P_V	V_V	P_C	V_C	$Time$
11	22	22	11	10	2.0901	31	62	62	31	30	7.7427
12	24	24	14	12	2.5595	32	64	64	34	32	13.3900
13	26	26	14	12	3.8995	33	66	66	34	32	8.8322
14	28	28	15	14	2.0418	34	68	68	35	34	11.1826
15	30	30	15	14	2.3370	35	70	70	35	34	12.1919
16	32	32	18	16	5.4044	36	72	72	38	36	12.6346
17	34	34	18	16	3.8734	37	74	74	38	36	15.7232
18	36	36	19	18	4.6272	38	76	76	39	38	12.4650
19	38	38	19	18	4.0508	39	78	78	39	38	18.9598
20	40	40	22	20	7.9871	40	80	80	42	40	20.8625
21	42	42	22	20	5.0642	41	82	82	42	40	16.7570
22	44	44	23	22	5.9443	42	84	84	43	42	12.0811
23	46	46	23	22	7.1294	43	86	86	43	42	19.5244
24	48	48	26	24	5.5255	44	88	88	46	44	17.6758
25	50	50	26	24	5.9522	45	90	90	46	44	25.3698
26	52	52	27	26	6.8825	46	92	92	47	46	22.2200
27	54	54	27	26	8.1232	47	94	94	47	46	30.8439
28	56	56	30	28	10.9514	48	96	96	50	48	19.9436
29	58	58	30	28	7.5786	49	98	98	50	48	20.9942
30	60	60	31	30	9.8275	50	100	100	51	50	19.3494

- C represents the number of consequences;
- $P_V = V_V$ represent the number of variables in *P-base* and *V-base*;
- $P_C = V_C$ represent the number of constraints in *P-base* and *V-base*;
- $Time$ is the average CPU time in seconds.

Table 3: Detailed Results

As for the worst case, the global optimum might not be found until we cut off all pseudo-global optimizers, and the lower bound information becomes useless. This will make ALGORITHM 2 no different from a pure cutting plane approach. The indefinite QP is a type of very difficult problem because it was demonstrated that the indefinite QP is NP-hard even with only one negative eigenvalue, [17]. However,

according to the computational results obtained, we observed that they overall are quite encouraging. Most of the problems are solved within the first three iterations and the lower bound information obtained from (9) actually takes effects.

7 Further Research

For ALGORITHM 2 presented in this paper, some additional research directions are suggested. In calculating a tight lower bound, other approaches exist. For example, LINGO can generate a very tight lower bound (often it is the global optimum) even though its best objective value always remains far from the lower bound for a very long period. The Reformulation Linearization Technique (RLT) in [20, 21] is also a promising method in the sense that RLT can generate feasible points as well as lower bounds for BLP problems. Moreover, it has been proved that its lower bounds are at least as good as those generated by [1, 2]. However, the main drawback is that the amount of work required to construct the necessary matrix increases very rapidly as the number of variables and constraints grow due to the combinatorial number of cross products that must be considered. Therefore, RLT is not particularly attractive for our BLP problem which contains so many lower and upper bounds, although it would be interesting to test its relative merits for different problem sizes. A worthwhile direction would be to search for a tighter lower bound, than the one proposed herein, that is relatively inexpensive to compute.

The BLP problem in DELTA is very special; i.e., it only includes $x_i y_i$, which makes the matrix C in (3) simply possess diagonal entries with $+1$ and -1. However, as shown in [2], it is also possible to handle the case where the diagonal entries are arbitrary real numbers, thus creating weighted utility objective functions.

Suppose $b \in R^n$ and let $B = diag(b)$. Then the convex envelope of B is:

$$
\begin{aligned}
&Vex_{\Omega_i}[b_i x_i y_i] = \max\{b_i \varphi_i^1(x_i y_i), b_i \varphi_i^2(x_i y_i)\}, \\
&\text{where} \\
&\varphi_i^1(x_i y_i) = \begin{cases} m_i x_i + l_i y_i - l_i m_i & \text{if} \quad b_i > 0 \\ M_i x_i + l_i y_i - l_i M_i & \text{if} \quad b_i \le 0 \end{cases}, \\
&\varphi_i^2(x_i y_i) = \begin{cases} M_i x_i + L_i y_i - L_i M_i & \text{if} \quad b_i > 0 \\ m_i x_i + L_i y_i - L_i m_i & \text{if} \quad b_i \le 0 \end{cases}
\end{aligned} \tag{10}
$$

Moreover, the convex envelope of $x_i y_i$ can also be extended to $x_i y_j$ where $i \neq j$, and thereby underestimate arbitrary bilinear objective functions.

References

[1] Al-Khayyal, F.A. and Falk J.E. "Jointly Constrained Biconvex Programming", Mathematics of Operations Research, 8:273-286, 1983.

[2] Al-Khayyal, F.A. "Jointly Constrained Bilinear Programs and Related Problems: An Overview", Computers & Mathematics with Application, Vol.19, No.11:53-62, 1990.

[3] Balas, E. "Intersection Cuts - a New Type of Cutting Planes for Integer Programming", Operations Research 19:19-39, 1971.

[4] Danielson, M. *Computational Decision Analysis*, Doctoral Thesis, Department of Computer and Systems Sciences, Stockholm University and Royal Institute of Technology, 1997.

[5] Danielson, M. "Generalized Evaluation in Decision Analysis", in press, to appear in European Journal of Operational Research, 2004.

[6] Danielson, M. and Ekenberg, L. "A Framework for Analysing Decisions under Risk", European Journal of Operational Research, Vol.104(3):474-484, 1998.

[7] Danielson, M. and Ekenberg, L. "Multi-criteria Evaluation of Decision Trees", Proceedings of the 5th International Conference of the Decision Sciences Institute, 1999.

[8] Danielson, M. Ekenberg, L. Johansson, J. and Larsson, A. "The *Decide*IT Decision Tool", Proceedings of ISIPTA-2003, 2003.

[9] Ding, X.S. Danielson, M. and Ekenberg, L. "Non-linear Programming Solvers for Decision Analysis Support Systems", Proceedings of International Conference on Operations Research (OR 2003), pp.475-482, Springer-Verlag, 2004.

[10] Ding, X.S. Ekenberg, L. and Danielson, M. "A Fast Bilinear Optimization Algorithm", submitted, 2003.

[11] Ekenberg, L. Boman, M. and Linneroth-Bayer, J. "General Risk Constraints", Journal of Risk Research, 4(1):31-47, 2001.

[12] Ekenberg, L. Brouwers, L. Danielson, M. Hansson, K. Johansson, J. Riabacke, A. and Vári A. "Simulation and Analysis of Three Flood Management Strategies", IIASA Interim Report, IR-03-003, Laxenburg, Austria, 2003.

[13] Gallo, G. and Ülkücü, A. "Bilinear Programming: an Exact Algorithm", Mathematical Programming 12:173-194, 1977.

[14] Gill, P.E. Murray, W. and Saunders, M.A. "User's Guide for SQOPT 5.3: A Fortran Package for Large-scale Linear and Quadratic Programming", Report NA 97-4, Department of Mathematics, University of California, San Diego, USA, 1997.

[15] Horst, R. and Pardalos, P.M. *Handbook of Global Optimization*, Kluwer, Dordrecht, 1995.

[16] Konno, H. "Bilinear Programming", Parts I and II, Technical Report No. 71-9 and 71-10, Operations Research House, Stanford University, USA, 1971.

[17] Pardalos, P.M. and Vavasis, S.A. "Quadratic Programming with One Negative Eigenvalue Is NP-hard", Journal of Global Optimization, 1:15-23, 1991.

[18] Pardalos, P.M. and Romeijn, H.E. *Handbook of Global Optimization Volume 2*, Kluwer Academic Publishers, the Netherlands, 2002.

[19] Ritter, K. "A Method for Solving Maximum-Problems with a Nonconcave Quadratic Objective Function", Z. Wahrscheinlichkeitstheorie verw. Geb. 4:340-351, 1966.

[20] Sherali, H.D. and Adams, W.P. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, Kluwer Academic Publishers, Dordrecht/Boston/London, 1999.

[21] Sherali, H.D. and Alameddine, A. "A New Reformulation Linearization Algorithm for Bilinear Programming Problems", Journal of Global Optimization, Vol.2:379-410, 1992.

[22] Sherali, H.D. and Shetty, C.M. "A Finitely Convergent Algorithm for Bilinear Programming Problems Using Polar Cuts and Disjunctive Face Cuts", Mathematical Programming, Vol.19:14-31, 1980.

[23] Shetty, C.M. and Sherali, H.D. "Rectilinear Distance Location-Allocation Problem: A Simplex Based Algorithm", Proceedings of the International Symposium on Extremal Methods and Systems Analyses, Springer-Verlag, Vol.174:442-464, 1980.

[24] http://www.tomlab.biz/

[25] Tuy, H. "Concave Programming under Linear Constraints", Dokl. Akad. Naul SSR 159:32-35; English translation in Soviet Math. Dokl. 5:1437-1440, 1964.

[26] Vaish, H. and Shetty, C.M. "A Cutting Plane Algorithm for the Bilinear Programming Problem", Naval Reaearch Logistics Quarterly 24:83-94, 1977.