KYBERNETIKA — VOLUME *59* (2023), NUMBER 1, PAGES 64–87

ROBUST OPTIMALITY ANALYSIS FOR LINEAR PROGRAMMING PROBLEMS WITH UNCERTAIN OBJECTIVE FUNCTION COEFFICIENTS: AN OUTER APPROXIMATION APPROACH

Zhenzhong Gao and Masahiro Inuiguchi

Linear programming (LP) problems with uncertain objective function coefficients (OFCs) are treated in this paper. In such problems, the decision-maker would be interested in an optimal solution that has robustness against uncertainty. A solution optimal for all conceivable OFCs can be considered a robust optimal solution. Then we investigate an efficient method for checking whether a given non-degenerate basic feasible (NBF) solution is optimal for all OFC vectors in a specified range. When the specified range of the OFC vectors is a hyper-box, i. e., the marginal range of each OFC is given by an interval, it has been shown that the tolerance approach can efficiently solve the robust optimality test problem of an NBF solution. However, the hyper-box case is a particular case where the marginal ranges of some OFCs are the same no matter what values the remaining OFCs take. In real life, we come across cases where some OFCs' marginal ranges depend on the remaining OFCs' values. For example, the prices of products rise together in tandem with raw materials, the gross profit of exported products increases while that of imported products decreases because they depend on the currency exchange rates, and so on. Considering those dependencies, we consider a case where the range of the OFC vector is specified by a convex polytope. In this case, the tolerance approach to the robust optimality test problem of an NBF solution becomes in vain. To treat the problem, we propose an algorithm based on the outer approximation approach. By numerical experiments, we demonstrate how the proposed algorithm efficiently solves the robust optimality test problems of NBF solutions compared to a conventional vertex-listing method.

Keywords: linear programming problems, interactive uncertain coefficients, robust optimality analysis, outer approximation approach, convex polytope

Classification: 90C05, 52B12

1. INTRODUCTION

Uncertainty is frequently encountered in real-world applications due to measurement restrictions, noise, insufficient knowledge, etc. The uncertainty has been studied for centuries, dating back to Archimedes' era as a study of measuring perimeters of circles [8]. The uncertainty was expressed by an interval defined by lower and upper

DOI: 10.14736/kyb-2023-1-0064

bounds including the exact value. Current researchers and practitioners still use this expression in dealing with uncertainties because of its conciseness and explicitness.

Sensitivity analysis [1], developed in the last century, is a well-known approach to the influence evaluation of the small fluctuations of coefficients of linear programming (LP) problems. It usually treats a small and local perturbation in a single coefficient and evaluates the sensitivity and stability of an optimal solution against the perturbation. Such a procedure is called the *robust optimality analysis* in this paper. By utilising *shadow price*, it can analyse the robustness against any coefficient of the problem, including any left-hand-side coefficient in constraints. However, it has two drawbacks: one is that the analysis becomes complex when the optimal basic solution is degenerate, and the other is that simultaneous fluctuations of multiple coefficients cannot be treated easily. Then the sensitivity analysis can only provide limited information about the robustness of the optimal solution [20].

To overcome the limitation, Bradley, Hax and Magnanti [1] proposed the *100 Percent Rule*, which can treat multiple uncertain coefficients by obtaining a convex cone from the optimal condition of a basis. When the coefficient values are in the convex cone, the corresponding basic solution is feasible and optimal. The convex cone is called the *optimality assurance cone (OAC)* in this paper. Utilising the OAC, the robustness of an optimal basis can be evaluated by testing whether the given range of uncertain coefficients is included in the OAC. Nevertheless, this test requires enormous computational effort when the range of uncertain coefficients is complex.

Wendell [23, 24] proposed a method using the 100 Percent Rules, called the *tolerance approach*. The tolerance approach enables us to analyse the robust optimality of a given feasible basic solution to an LP problem with little computational effort. In the tolerance approach, a maximal hyper-box included in the OAC is calculated to materialise a range of uncertain coefficients to preserve the optimality. Wondolowski [25] and Filippi [4] improved this approach by enlarging the hyper-box included in the OAC. Enormous research related to the robust optimisation is established in different realms, such as the *fuzzy set approach* [16, 19] and the *probabilistic and stochastic approach* [21, 22].

For the situation where the range of uncertain objective function coefficients (OFCs) are given a priori without knowing a candidate feasible solution, Inuiguchi and Sakawa [17, 14] proposed the concepts of *possible and necessary optimalities*. A possibly optimal solution is a feasible solution optimal for at least one coefficient's values in the given range, while a necessarily optimal solution is a feasible solution optimal for all coefficient's values in the given range. Namely, the necessary optimality corresponds to the robust optimality. Since a possibly optimal solution is usually too weak and loses the optimality by a small fluctuation of coefficient values, a necessarily optimal solution is preferable. However, the necessarily optimal solution does not always exist. Recent studies have shown that the tolerance approach can easily treat the test problem for checking the necessary optimality of an obtained basic feasible solution when the range of uncertain coefficients is expressed as a hyper-box in an oblique coordinate system [15]. This result is extended when the range of uncertain coefficients is a fuzzy set called an *oblique fuzzy vector* [15]. In this case, the test problem becomes a problem for evaluating the degree of the necessary optimality, i. e., to what extent the given basic feasible solution stays optimal against the fluctuation of uncertainty OFCs.

When the range of uncertain coefficients cannot be expressed as a hyper-box in any coordinate system, the necessary optimality test becomes intractable. If the marginal ranges of any uncertain coefficients are the same, no matter what values the remaining uncertain coefficients take, the range of uncertain coefficients is expressed as a hyper-box. In such a case, it is called that these uncertain coefficients are non-interactive. The assumption of the non-interactivity of uncertain coefficients is rather restrictive. In real-world problems, we encounter interactive uncertain coefficients, i. e., the marginal ranges of some uncertain coefficients depend on the values of the remaining coefficients.

In stochastic programming [2, 21], we come across stochastic dependence, a notion similar to the mentioned interactiveness. A part of the stochastic dependency can be expressed by a covariance matrix or a correlation matrix, which shows the correlations between two uncertain variables. Gao and Inuiguchi [5] proposed a numerical approach once the probability distributions of all uncertain coefficients are known. However, obtaining the probability distribution is not an easy task, and the covariance matrix does not cover all classes of stochastic dependence.

In this paper, we treat the case where the range of uncertain OFCs is expressed by a convex polytope. It implies that the range of the uncertain OFCs is defined by a finite number of linear inequalities. When the upper and lower bounds of a sufficient number of linear fractional function values of uncertain coefficients are known, the range of uncertain coefficients reduces to a convex polytope. The range is not always a hyper-box; thus, the uncertain coefficients can be interactive. However, this case is still a special case of the interactive uncertain coefficients. Nevertheless, the problem of testing the necessary optimality of a given non-degenerate basic feasible solution in this special case cannot be solved easily. Then we investigate a solution method for the necessary optimality test problem. We develop an outer approximation algorithm [13] for the test problem utilising the convexity and boundedness of the range of the uncertain OFCs.

Unlike the conventional vertex-listing approach, the proposed approach does not need to list all vertices and check whether they fall into the OAC. This reduction decreases the computational burdens significantly. The proposed approach needs no extra information concerning the uncertainty but only a series of linear constraints. New vertices are obtained by a pivoting process of the simplex method [12]. We demonstrate that the proposed approach outperforms the conventional approach with computational efficiency.

This paper is organised as follows. Section 2 gives some preliminaries about LP and the robust optimality analysis, where we will show some results in the case when the range of uncertain coefficients is expressed as a hyper-box. Then, we carefully explain the outer approximation used in the proposed algorithm in Section 3. The detailed algorithm is concretely shown in Section 4 with a numerical example. In Section 5, we compare our approach with the conventional listing methods through a numerical experiment. The conclusions and an outline of the future work are given in Section 6.

1.1. Notations

In this paper, we use the following notations:

- $\mathbb{R}^n$ denotes an n-dimensional Euclidean space;

- $|\cdot|$ denotes an entry-wise absolute operator;
- $\mathrm{Card}(\cdot)$ denotes the cardinality of a set;
- $\mathbb{I}$ denotes an index set with convenient cardinality;
- I denotes an identity matrix with convenient dimension;
- $A^{-\mathrm{T}}$ denotes the inverse of a transposed non-singular matrix A, i. e., $A^{-\mathrm{T}} = (A^{\mathrm{T}})^{-1} = (A^{-1})^{\mathrm{T}}$.
- $\mathrm{diag}(\boldsymbol{a})$ denotes a diagonal matrix with the diagonal entries being vector $\boldsymbol{a}$.

2. LP PROBLEMS AND ROBUST OPTIMALITY ANALYSIS

In this paper, the following standard form of LP problems are treated:

$$\text{maximize } \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}, \text{ subject to } A\boldsymbol{x} = \boldsymbol{b}, \ \boldsymbol{x} \geq \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ represents the decision variable vector, while $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^n$ are the coefficient matrix and vectors, respectively.

By the simplex method [1], we can separate a *basic feasible* solution $\boldsymbol{x}^*$ into the basic and non-basic sub-vectors $\boldsymbol{x}^*_B \in \mathbb{R}^m$ and $\boldsymbol{x}^*_N \in \mathbb{R}^{n-m}$ respectively by an index set $\mathbb{I}_B(\boldsymbol{x}^*) \in \{1, 2, \ldots, n\}$ satisfying $\mathrm{Card}(\mathbb{I}_B(\boldsymbol{x}^*)) = m$. Orderly by $\mathbb{I}_B(\boldsymbol{x}^*)$, we can also separate the matrix A and vector $\boldsymbol{c}$. Sub-matrices $A_B \in \mathbb{R}^{m\times m}$ and $A_N \in \mathbb{R}^{m\times(n-m)}$, composed of the columns of A indexed by $\mathbb{I}_B(\boldsymbol{x}^*)$ and $\{1, 2, \ldots, n\}\backslash\mathbb{I}_B(\boldsymbol{x}^*)$, denote the basic and non-basic parts of A respectively. Similarly, sub-vectors $\boldsymbol{c}_B \in \mathbb{R}^m$ and $\boldsymbol{c}_N \in \mathbb{R}^{n-m}$ denote the basic and non-basic parts of $\boldsymbol{c}$, respectively. Since A_B should always be non-singular, i. e., A_B^{-1} always exists, the value of $\boldsymbol{x}^*$ can be obtained with $\boldsymbol{x}^*_B = A_B^{-1}\boldsymbol{b}$ and $\boldsymbol{x}^*_N = \boldsymbol{0}$. Therefore, we have the following proposition to confirm the optimality of a basic feasible solution [1, 3].

Proposition 2.1. A basic feasible solution $\boldsymbol{x}^* \in \mathbb{R}^n$ in Problem (1) is optimal if and only if it meets the following conditions:

$$\boldsymbol{c}_N - A_N^{\mathrm{T}}A_B^{-\mathrm{T}}\boldsymbol{c}_B \leq \boldsymbol{0} \text{ and } A_B^{-1}\boldsymbol{b} \geq \boldsymbol{0}, \tag{2}$$

where the optimal solution value is obtained with $\boldsymbol{x}^*_B = A_B^{-1}\boldsymbol{b}$ and $\boldsymbol{x}^*_N = \boldsymbol{0}$, and the optimised value is $\boldsymbol{c}_B^{\mathrm{T}}A_B^{-1}\boldsymbol{b}$.

Proposition 2.1 gives a detailed description of the simplex method's methodology in identifying whether a basic feasible solution is optimal. However, when considering the uncertainty in coefficients, i. e., the coefficients in the LP problem cannot be uniquely determined, there exist several problems.

Firstly, Condition (2) in Proposition 2.1 only guarantees the feasibility and optimality of $\boldsymbol{x}^*$ with the corresponding basic index set $\mathbb{I}_B(\boldsymbol{x}^*)$. However, when the values of the coefficients in A and $\boldsymbol{b}$ change, the value of $\boldsymbol{x}^*$ may also change due to $\boldsymbol{x}^*_B = A_B^{-1}\boldsymbol{b}$, while $\mathbb{I}_B(\boldsymbol{x}^*)$ would keep invariant as long as Condition (2) is satisfied. In this situation, since $\boldsymbol{x}^*_B = A_B^{-1}\boldsymbol{b}$ is always satisfied, if $\boldsymbol{b}$ cannot be uniquely determined, an obtained

optimal solution may not be optimal or even not feasible any more. Secondly, the feasible solution can only be basic, i. e., $\mathrm{Card}(\mathbb{I}_B(\boldsymbol{x}^*)) = m$. Thirdly, which is the most puzzling, is that there should exist no *degeneracy* in $\boldsymbol{x}^*$, i. e., $\boldsymbol{x}_B^* = A_B^{-1}\boldsymbol{b} > \boldsymbol{0}$ should always be satisfied.

Although the problem caused by non-basic or degenerate feasible solution has been studied by Hladík [10] and Gao, Inuiguchi [6], it is trivial to fall into a tangle by taking them into consideration. Therefore, in this paper, we only consider a solution to be *non-degenerated basic feasible* (NBF). Moreover, since we aim to concentrate on the value of the solution, we assume there exists no uncertainty in constraints, i. e., A and $\boldsymbol{b}$ are constant. Hence, the invariance of $\mathbb{I}_B(\boldsymbol{x}^*)$ is equivalent to the invariance of the value of $\boldsymbol{x}^*$.

Consequently, the LP problem with uncertain coefficients is expressed as follows:

$$\text{maximize } \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}, \text{ subject to } A\boldsymbol{x} = \boldsymbol{b}, \ \boldsymbol{x} \geq \boldsymbol{0}, \tag{3}$$

where $\boldsymbol{\gamma} \in \Phi$ denotes the OFC vector with uncertainties and $\Phi \subseteqq \mathbb{R}^n$ is the set including these uncertainties. A and $\boldsymbol{b}$ are constant coefficient matrix and vector, respectively.

Since the LP problem with uncertainties is composed of multiple scenarios, each of which represents a conventional exact one, we have to consider all of them. Since Problem (3) has a fixed feasible set, it is only necessary to consider the scenarios derived by $\boldsymbol{\gamma} \in \Phi$. Therefore, we can utilise the concept of *possible and necessary optimality* by Inuiguchi and Sakawa [17]:

Definition 2.2. (Possible and Necessary Optimality) Let $\Phi \subseteqq \mathbb{R}^n$ be a set including all possible uncertain OFC vectors $\boldsymbol{\gamma}$ in Problem (3). Then, a feasible solution $\boldsymbol{x}^*$ is

- *possibly optimal* for Φ if it is optimal for at least one scenario of $\boldsymbol{\gamma} \in \Phi$,
- *necessarily optimal* for Φ if it is optimal for all scenarios of $\boldsymbol{\gamma} \in \Phi$.

It is noted that the infeasible solution is neither possibly nor necessarily optimal.

Since a possibly optimal solution is too weak to be against the uncertainty, we only focus on a necessarily optimal one. Generally, for an LP problem with a fixed feasible set, the check of whether a feasible solution is necessarily optimal is equivalent to the analysis of the OFC vector. To accomplish such a goal, we use the set defined by Inuiguchi et al. [15]:

$$\mathscr{L}(\boldsymbol{x}^*) := \Big\{\boldsymbol{c} \in \mathbb{R}^n : \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* = \max\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} : A\boldsymbol{x} = b, \boldsymbol{x} \geq \boldsymbol{0}\}\Big\}. \tag{4}$$

Namely, $\mathscr{L}(\boldsymbol{x}^*)$ is the set composed of OFC vectors such that $\boldsymbol{x}^*$ is optimal. Hence, we have the following equivalences:

$$\boldsymbol{x}^* \text{ is possibly optimal} \iff \Phi \cap \mathscr{L}(\boldsymbol{x}^*) \neq \emptyset, \tag{5}$$

$$\boldsymbol{x}^* \text{ is necessarily optimal} \iff \Phi \subseteqq \mathscr{L}(\boldsymbol{x}^*). \tag{6}$$

In (4), $\mathscr{L}(\boldsymbol{x}^*)$ is well-defined, which, nevertheless, is difficult to obtain. Fortunately, when $\boldsymbol{x}^*$ is an NBF solution, we can simply utilise Proposition 2.1 to build the equivalence called *optimality assurance cone* [1, 15]:

Definition 2.3. (Optimality Assurance Cone) Let $\boldsymbol{x}^*$ be an NBF solution for Problem (3). Then the *optimality assurance cone* (OAC) of $\boldsymbol{x}^*$ is defined as:

$$\mathscr{M}(\boldsymbol{x}^*) := \Big\{\boldsymbol{c} \in \mathbb{R}^n : \boldsymbol{c}_N - A_N^{\mathrm{T}} A_B^{-\mathrm{T}} \boldsymbol{c}_B \leq \mathbf{0}\Big\}, \tag{7}$$

where A_B, A_N, $\boldsymbol{c}_B$ and $\boldsymbol{c}_N$ are defined the same as Proposition 2.1.

For convenience, we use a matrix $M(\boldsymbol{x}^*) \in \mathbb{R}^{(n-m)\times n}$ that maintains the order of the OFC vector instead of separating them physically by $\mathbb{I}_B(\boldsymbol{x}^*)$. Hence, we reformulate the OAC as:

$$\mathscr{M}(\boldsymbol{x}^*) = \Big\{\boldsymbol{c} \in \mathbb{R}^n : M(\boldsymbol{x}^*)\boldsymbol{c} \leq \mathbf{0}\Big\}, \tag{8}$$

where we note that if $\boldsymbol{c}^{\mathrm{T}}$ is permuted as $(\boldsymbol{c}_B^{\mathrm{T}}, \boldsymbol{c}_N^{\mathrm{T}})$, then $M(\boldsymbol{x}^*) = (-A_N^{\mathrm{T}} A_B^{-\mathrm{T}}, I)$.

By Proposition 2.1 and Definition 2.3, we understand that $\mathscr{M}(\boldsymbol{x}^*) = \mathscr{L}(\boldsymbol{x}^*)$ if $\boldsymbol{x}^*$ is an NBF solution, which derives the following proposition:

Proposition 2.4. Let $\boldsymbol{x}^*$ denote an NBF solution for Problem (3) with $\boldsymbol{\gamma} \in \Phi$. Then $\boldsymbol{x}^*$ is *necessarily optimal* if and only if the following condition is valid:

$$\Phi \subseteqq \mathscr{M}(\boldsymbol{x}^*), \tag{9}$$

where $\mathscr{M}(\boldsymbol{x}^*)$ denotes the OAC of $\boldsymbol{x}^*$.

Since Φ usually is an infinite set, it is impossible to check (9) for every entry in Φ. Fortunately, if Φ is a convex polytope, we can simplify the procedure with the proposition below:

Proposition 2.5. Let $\boldsymbol{x}^*$ denote an NBF solution for Problem (3) with $\boldsymbol{\gamma} \in \Phi$, where Φ is a convex polytope, and let $\mathbf{V}(\Phi)$ be the vertex set of Φ. Then $\boldsymbol{x}^*$ is *necessarily optimal* if and only if $\mathbf{V}(\Phi) \subseteqq \mathscr{M}(\boldsymbol{x}^*)$. Furthermore, $\boldsymbol{x}^*$ is *possibly optimal* if and only if $\mathbf{V}(\Phi) \cap \mathscr{M}(\boldsymbol{x}^*) \neq \emptyset$ and $\boldsymbol{x}^*$ is not optimal if and only if $\mathbf{V}(\Phi) \cap \mathscr{M}(\boldsymbol{x}^*) = \emptyset$.

P r o o f. We first focus on the necessary optimality. Let $\mathbf{V}(\Phi) = \{\boldsymbol{c}^i \in \mathbb{R}^n : i = 1, 2, \ldots, k\}$, where $k = \mathrm{Card}(\mathbf{V}(\Phi))$. Since Φ is a convex polytope, for any vector $\boldsymbol{c} \in \Phi$, it can be expressed with

$$\boldsymbol{c} = \sum_{i=1}^{k} \lambda_i \boldsymbol{c}^i, \text{ where } \sum_{i=1}^{k} \lambda_i = 1 \text{ and } \lambda_i \geq 0 \text{ for } i = 1, 2, \ldots, k. \tag{10}$$

Since $\mathbf{V}(\Phi) \subseteqq \mathscr{M}(\boldsymbol{x}^*)$ implies $\boldsymbol{c}^i \in \mathscr{M}(\boldsymbol{x}^*)$ for any $i = 1, 2, \ldots, k$, it is not hard to know that $\boldsymbol{c} \in \mathscr{M}(\boldsymbol{x}^*)$ for any $\boldsymbol{c} \in \Phi$, which means $\Phi \subseteqq \mathscr{M}(\boldsymbol{x}^*)$. Hence, if $\mathbf{V}(\Phi) \subseteqq \mathscr{M}(\boldsymbol{x}^*)$, $\boldsymbol{x}^*$ is necessarily optimal. On the other hand, if $\boldsymbol{x}^*$ is necessarily optimal, we have $\Phi \subseteqq \mathscr{M}(\boldsymbol{x}^*)$ by Proposition 2.4. Since $\mathbf{V}(\Phi) \subseteqq \Phi$, it is obviously to have $\mathbf{V}(\Phi) \subseteqq \mathscr{M}(\boldsymbol{x}^*)$.

For the possible optimality part, we know that if $\mathbf{V}(\Phi) \cap \mathscr{M}(\boldsymbol{x}^*) \neq \emptyset$, $\Phi \cap \mathscr{M}(\boldsymbol{x}^*) \neq \emptyset$ due to $\mathbf{V}(\Phi) \subseteqq \Phi$. Hence, $\boldsymbol{x}^*$ is possibly optimal. On the other hand, if $\boldsymbol{x}^*$ is possibly

optimal, $\Phi \cap \mathscr{M}(\boldsymbol{x}^*) \neq \emptyset$ indicates that there exists at least one vector $\hat{\boldsymbol{c}} \in \Phi$ such that $\hat{\boldsymbol{c}} \in \mathscr{M}(\boldsymbol{x}^*)$. Let $\hat{\boldsymbol{c}}$ be expressed by $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_k$ by (10). Assume that in this condition, there exists no vertex $\boldsymbol{c}^i$ in $\mathscr{M}(\boldsymbol{x}^*)$ for any $i = 1, 2, \ldots, k$, i. e., $\nexists i \in \{1, 2, \ldots, k\}$ such that $\boldsymbol{c}^i \in \mathscr{M}(\boldsymbol{x}^*)$, then $\hat{\boldsymbol{c}}$ cannot be in $\mathscr{M}(\boldsymbol{x}^*)$ due to (10), which contradicts the assumption that $\hat{\boldsymbol{c}} \in \mathscr{M}(\boldsymbol{x}^*)$. Therefore, there exist at least one $\boldsymbol{c}^i \in \mathscr{M}(\boldsymbol{x}^*)$, indicating that $\mathbf{V}(\Phi) \cap \mathscr{M}(\boldsymbol{x}^*) \neq \emptyset$. The none optimality part is obvious so we do not give the proof. □

Proposition 2.5 indicates that when Φ is a convex polytope, we can check the necessary optimality of an NBF solution $\boldsymbol{x}^*$ by listing all vertices of Φ and check whether they are in $\mathscr{M}(\boldsymbol{x}^*)$. If they are, then $\boldsymbol{x}^*$ is necessarily optimal. If there only exist some vertices in $\mathscr{M}(\boldsymbol{x}^*)$, then $\boldsymbol{x}^*$ is possibly optimal. Moreover, $\boldsymbol{x}^*$ is not optimal if there exists no vertex in $\mathscr{M}(\boldsymbol{x}^*)$. In this paper, we call this process the *robust optimality analysis* to a specific LP problem.

Although Proposition 2.5 sets the foundation for the robust optimality analysis that checks the necessary optimality of an NBF solution, there still exists a barrier to the calculation. For example, if Φ is an n-dimensional hyper-box, then the cardinality of $\mathbf{V}(\Phi)$ is 2^n, which means we need to check 2^n times to determine whether the solution is necessarily optimal. However, executing such amount of computations is excessively demanding when n is large enough. Hence, we still need some other techniques to address the problem.

2.1. Robust optimality analysis in interval LP problems

As we have mentioned that it is essential to use some techniques to deal with the difficulty in computational complexity, we firstly introduce the one to treat non-interactive uncertain coefficients, i. e., each entry c_i in $\boldsymbol{\gamma} \subseteqq \Phi$ is included by an interval $[c_i^{\mathrm{L}}, c_i^{\mathrm{R}}]$, $i = 1, 2, \ldots, n$. For convenience, we denote the lower and upper bounds of Φ as $\boldsymbol{c}^{\mathrm{L}} := (c_1^{\mathrm{L}}, c_2^{\mathrm{L}}, \ldots, c_n^{\mathrm{L}})^{\mathrm{T}}$ and $\boldsymbol{c}^{\mathrm{R}} := (c_1^{\mathrm{R}}, c_2^{\mathrm{R}}, \ldots, c_n^{\mathrm{R}})^{\mathrm{T}}$, respectively. Furthermore, the centre and width of Φ are denoted as $\boldsymbol{c}^{\mathrm{C}} := (\boldsymbol{c}^{\mathrm{R}} + \boldsymbol{c}^{\mathrm{L}})/2$ and $\boldsymbol{c}^{\mathrm{S}} := (\boldsymbol{c}^{\mathrm{R}} - \boldsymbol{c}^{\mathrm{L}})/2$, respectively.

Since Φ is a compact hyper-box in $\mathbb{R}^n$, the problem is also called an *interval linear programming* problem [7, 11, 18], where literature has shown that it can be solved by the tolerance approach straightforwardly. Hence, we only give a brief review [15, 23], where the main theorem is given by

Theorem 2.6. Let an interval LP problem be defined in (3), where $\boldsymbol{\gamma} \in \Phi$ and Φ be an interval hyper-box in $\mathbb{R}^n$ with $\boldsymbol{c}^{\mathrm{C}}$ and $\boldsymbol{c}^{\mathrm{S}}$ denoting the centre and width of Φ, respectively. Let $\boldsymbol{x}^*$ be an NBF solution and $M(\boldsymbol{x}^*)$ be the matrix for $\mathscr{M}(\boldsymbol{x}^*)$ by (8). For $k = 1, 2, \ldots, n - m$, we define

$$\tau_k = \begin{cases} \dfrac{\sum_{j=1}^{n} M_{kj}(\boldsymbol{x}^*) c_j^{\mathrm{C}}}{\sum_{j=1}^{n} |M_{kj}(\boldsymbol{x}^*)||c_j^{\mathrm{S}}|}, & \text{if } \sum_{j=1}^{n} |M_{kj}(\boldsymbol{x}^*)||c_j^{\mathrm{S}}| > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

$$\tau^{\min} = \min_{\substack{k=1,2,\ldots,n-m \\ \sum_{j=1}^{n} |M_{kj}(\boldsymbol{x}^*)||c_j^{\mathrm{S}}| > 0}} \tau_k. \tag{12}$$

Then $\boldsymbol{x}^*$ is necessarily optimal if and only if $\tau^{\min} \geq 1$. Otherwise, $\boldsymbol{x}^*$ is only possibly optimal.

To illustrate Theorem 2.6, we use the numerical example proposed in [15]:

Example 2.7. Let us consider a production problem. A factory manufactures two items, denoted by A and B. Both items need two resources, denoted by P and Q. It is known that the production of A costs 3 pounds of P and 4 pounds of Q per unit, while the production of B costs 3 pounds of P and 1 pound of Q per unit. However, there are only 42 pounds of P and 24 pounds of Q, and B is scheduled to be produced at most 9 units. Since the profits of A and B are also influenced by the market, the factory does not know them exactly. However, from past experience, the profit of A is always between 15 and 31 thousand JPY and the profit of B is always between 9 and 19 thousand JPY. Since the factory always wants to make the biggest profit, it needs a production schedule for A and B. However, the profits of A and B cannot be uniquely determined. Therefore, the most interesting problem is that, does there exist a schedule that is not affected by the profits of A and B and can always make the biggest profit?

To solve the problem, let us convert it to an LP problem as follows:

$$\begin{aligned} \text{maximize} \quad & c_1 x_1 + c_2 x_2, \\ \text{subject to} \quad & 3x_1 + 4x_2 \leq 42, \\ & 3x_1 + x_2 \leq 24, \\ & x_2 \leq 9, \ x_1, x_2 \geq 0, \end{aligned}$$

where x_1 and x_2 represent the amounts of production for A and B, respectively. The ranges of OFCs c_1 and c_2, i. e., prices of A and B, are given by $[15, 31]$ and $[9, 19]$, respectively. We assume that no matter what value c_2 takes, the range of c_1 would not change, and vice versa. Namely, c_1 and c_2 are non-interactive.

First, an NBF solution should be obtained. We obtain it by solving an LP problem with OFCs defined by the centre values, i. e., c_1^{C} and c_2^{C}, of the conceivable ranges of c_1 and c_2. We add slack variables $x_3, x_4, x_5 \geq 0$ and set $c_3, c_4, c_5 = 0$ with $c_1^{\mathrm{C}} = 23$ and $c_2^{\mathrm{C}} = 14$, we solve the LP problem and obtain $\check{\boldsymbol{x}}^* = (6, 6, 0, 0, 3)^{\mathrm{T}}$. Since the basic index set $\mathbb{I}_B(\check{\boldsymbol{x}}^*)$ is $\{1, 2, 5\}$, we obtain a temporary OAC $\check{\mathscr{M}}(\check{\boldsymbol{x}}^*) \subsetneqq \mathbb{R}^5$ as

$$\check{\mathscr{M}}(\check{\boldsymbol{x}}^*) = \left\{ (c_1, c_2, c_3, c_4, c_5)^{\mathrm{T}} \;\middle|\; \begin{array}{c} \frac{1}{9}c_1 - \frac{1}{3}c_2 + c_3 + \frac{1}{3}c_5 \leq 0 \\ -\frac{4}{9}c_1 + \frac{1}{3}c_2 + c_4 - \frac{1}{3}c_5 \leq 0 \end{array} \right\}.$$

By removing the slack variables x_3, x_4, x_5 with c_3, c_4, c_5, we have the solution $\boldsymbol{x}^* = (6, 6)^{\mathrm{T}}$ with $\mathscr{M}(\boldsymbol{x}^*)$ expressed as:

$$\mathscr{M}(\boldsymbol{x}^*) = \left\{ (c_1, c_2)^{\mathrm{T}} \;\middle|\; \begin{array}{c} \frac{1}{9}c_1 - \frac{1}{3}c_2 \leq 0 \\ -\frac{4}{9}c_1 + \frac{1}{3}c_2 \leq 0 \end{array} \right\}.$$

Since $\boldsymbol{c}^{\mathrm{C}} = (23, 14)^{\mathrm{T}}$ and $\boldsymbol{c}^{\mathrm{S}} = (8, 5)^{\mathrm{T}}$, we apply Theorem 2.6 and obtain $\tau_1 = 19/23$ and $\tau_2 = 50/47$, which indicates that $\tau^{\min} = 19/23 < 1$. Therefore, $\boldsymbol{x}^*$ is not necessarily optimal. The result can also be confirmed in Figure 1, where the vertex $(31, 9)^{\mathrm{T}}$ is out of $\mathscr{M}(\boldsymbol{x}^*)$. Therefore, there does not exist a plan that is not affected by the prices of A and B and can always make the biggest profit.

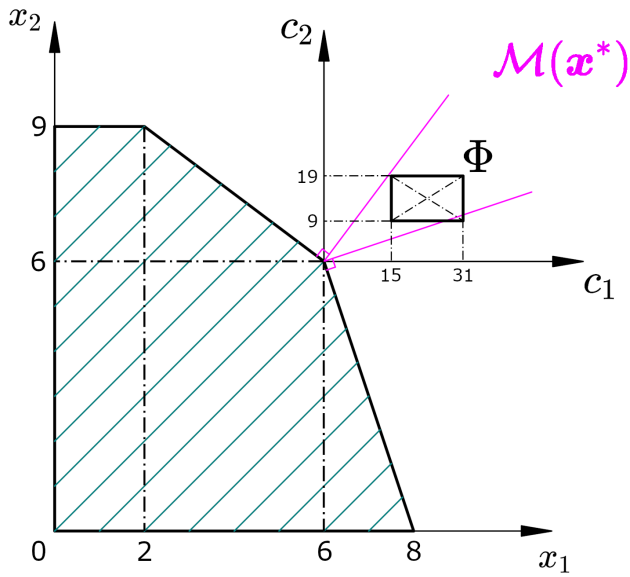


Fig. 1: Illustration of Example 2.7.

3. ROBUST OPTIMALITY ANALYSIS WITH A CONVEX POLYTOPE

Unlike the robust optimality analysis for LP problems with non-interactive uncertain coefficients, the interactive ones are more complicated.

Proposition 2.5 indicates that if all vertices of a convex polytope representing the ranges of uncertain coefficients are included by the OAC of an NBF solution, then the solution is necessarily optimal. However, vertex listing requires enormous computational effort. For example, let a convex polytope be defined as

$$\Phi = \Big\{ \boldsymbol{c} \in \mathbb{R}^n : G\boldsymbol{c} \leq \boldsymbol{g} \Big\}, \tag{13}$$

where $G \in \mathbb{R}^{p \times n}$ and $\boldsymbol{g} \in \mathbb{R}^p$ are constant matrix and vector, respectively. Since Φ is bounded, p must be excessively greater than n.

Intuitively, the process of listing all vertices of the convex polytope defined in (13) is equivalent to the combination of choosing n constraints from the total p constraints, of which the complexity order is extremely large with $O(p\mathrm{C}n)$. Although it is known that the maximum vertex number of the convex polytope defined in (13) is $O(p^{n/2})$ [9], much less than $O(p\mathrm{C}n)$, it still requires enormous computational effort.

Fortunately, if a convex polytope including Φ already makes a feasible solution necessarily optimal, the solution must be necessarily optimal for Φ. Namely,

Lemma 3.1. If a feasible solution $\boldsymbol{x}^* \in \mathbb{R}^n$ is necessarily optimal for a convex polytope Λ that includes Φ, i. e., $\Phi \subseteqq \Lambda$, then it is also necessarily optimal for Φ.

P r o o f . The proof is trivial, so we leave it to readers. □

The advantage of Lemma 3.1 is that it is not necessary to find all vertices at the beginning. Instead, we build a convex polytope in a much simpler structure and cut it iteratively to approximate the exact polytope. If the convex polytope at a certain step already makes a feasible solution necessarily optimal, we can terminate the procedure and state the solution to be necessarily optimal. Moreover, if a vertex of the current polytope belongs to the exact polytope but is out of the OAC, we can also terminate the procedure and state that there does not exist a necessarily optimal solution.

Hence, if we want to use Lemma 3.1 to reduce the calculation, we need to develop an algorithm satisfying the following conditions:

(i) Let Φ be the exact convex polytope, and Λ^k be the polytope including Φ at k_{th} step. Since we need to utilise Proposition 2.5 to check whether all vertices of Λ^k are in the OAC, we prefer a convex polytope Λ^0 in a simple structure for vertex-listing at the initial step.

(ii) Since the process is iterative, we need to update the convex polytope such that $\Phi \subseteqq \Lambda^{k+1} \subsetneqq \Lambda^k$. Moreover, we need the vertex set of Λ^k to obtain the vertex set of Λ^{k+1}.

Therefore, we found that the *outer approximation approach* can treat the problem, where we concentrated on the following two aspects:

(i) the construction of the initial convex polytope Λ^0,

(ii) the cutting approach to obtain Λ^{k+1} from Λ^k.

3.1. Construction of the initial convex polytope

Horst, Thoai and Tuy [13] showed that there exist several methods to construct the initial convex polytope. However, since Φ and Λ^k for any k are convex, we prefer the one with the least computation in obtaining all its vertices.

Moreover, since the convex polytope Φ is described by (13), which already ensures it is convex and closed, we need the initial process to check whether Φ is bounded. To accomplish both purposes, we use the following proposition [13] to build the initial convex polytope Λ^0:

Proposition 3.2. Given a convex polytope defined as $\Phi = \{\boldsymbol{c} \in \mathbb{R}^n : G\boldsymbol{c} \leq \boldsymbol{g}\}$, it is bounded if the following results are finite:

$$\alpha_0 = \max\left\{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{c} :\ G\boldsymbol{c} \leq \boldsymbol{g}\right\}, \tag{14}$$

$$\alpha_i = \min\left\{c_i :\ G\boldsymbol{c} \leq \boldsymbol{g}\right\},\ i = 1, 2, ..., n, \tag{15}$$

where $\boldsymbol{e} := (1, 1, \ldots, 1)^{\mathrm{T}}$.

Moreover, if Φ is bounded, then the initial triangular polytope Λ^0 is built as:

$$\Phi \subseteqq \Lambda^0 := \Big\{ \boldsymbol{c} \in \mathbb{R}^n : \ \boldsymbol{e}^{\mathrm{T}} \boldsymbol{c} \leq \alpha_0, \ c_i \geq \alpha_i, \ i = 1, 2, ..., n \Big\}, \tag{16}$$

where the $(n+1)$ vertices are obtained as $(\alpha_1, \alpha_2, \ldots, \alpha_n)^{\mathrm{T}}$ and

$$\Big(\alpha_1, \alpha_2, \ldots, \alpha_0 - \sum_{\substack{j=1 \\ j \neq k}}^{n} \alpha_j, \ldots, \alpha_n \Big)^{\mathrm{T}}, \ k = 1, 2, \ldots, n.$$

Although there exist other methods to build the initial convex polytope, such as the hyper-box, we confirm that the triangular polytope has its advantages:

- Since all initialisation methods can check the boundedness of Φ, the fewer polyhedral boundaries Λ^0 has, the less computational effort is required. As the triangular polytope has the fewest polyhedral boundaries with $(n+1)$, the number of conventional LP problems to be solved is $(n+1)$, fewer than other methods.
- Due to our specific cutting approach, we show that the number of cutting, which is always less than or equal to the number of constraints in Φ, is not much affected by the shape of the initial convex polytope. Hence, it is trivial to concentrate too much on the shape of Λ^0.
- The vertex set of Λ^0 can be simultaneously obtained when solving the series of LP problems. Moreover, since the triangular polytope has the least number of vertices, it requires the least computational effort.

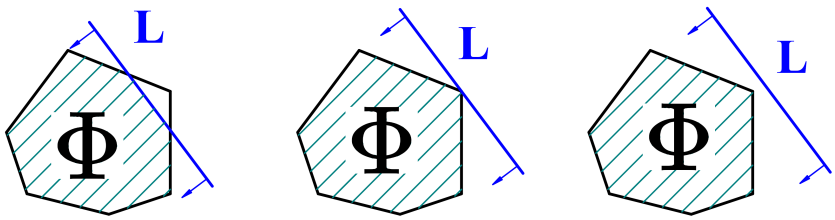
After obtaining the initial convex polytope Λ^0 with the vertex set $\mathbf{V}(\Lambda^0)$, we go to the iteration procedure.

3.2. Cutting and updating strategy on polytopes

Assume that we have obtained the polytope Λ^k with its vertex set $\mathbf{V}(\Lambda^k)$ at a specific step k, and we want to cut and update Λ^k to obtain Λ^{k+1} and $\mathbf{V}(\Lambda^{k+1})$. Since Φ is a convex polytope, we can use one of its constraints to cut Λ^k at each iteration. Moreover, since the maximum number of cutting is less than or equal to p, i. e., the number of constraints in Φ, we have $\Lambda^p = \Phi$, which guarantees the termination of the algorithm. However, to accomplish the task, we still need to solve several problems:

(i) The OAC of an NBF solution is necessary when checking its necessary optimality. However, a convex polytope, which is a set, does not enable us to do that. Hence, we need to choose a point in the polytope as the OFC vector to obtain an NBF solution.

(ii) When cutting a convex polytope, some boundaries may become useless since the polytope becomes smaller. The useless boundaries called *redundant constraints*, increase the calculation in obtaining new vertices in every iteration. However, searching for redundant constraints requires extra computational effort. Hence, it is essential to remove them efficiently.



(a) Valid. (b) Redundant but not strictly redundant. (c) redundant and strictly redundant.

Fig. 2: A constraint **L** to a convex polytope Φ.

(iii) After cutting a convex polytope and removing redundant constraints, some vertices disappear and some emerge. Hence, we need to add the new vertices and efficiently remove the useless ones.

For the first problem, we cannot randomly choose the vertex of Λ^0 because they are not in Φ. However, let $\boldsymbol{c}^0$ and $\boldsymbol{c}^i$, $i = 1, 2, \ldots, n$ denote the optimal solutions to (14) and (15), respectively. Then we define

$$\bar{\boldsymbol{c}} = \frac{1}{1+n} \sum_{i=0}^{n} \boldsymbol{c}^i. \tag{17}$$

We have $\bar{\boldsymbol{c}} \in \Phi$ because $\boldsymbol{c}^i \in \Phi$, $i = 0, 1, 2, \ldots, n$ and Φ is convex. Then we can use $\bar{\boldsymbol{c}}$ to obtain an optimal solution to the LP problem:

$$\text{maximize } \bar{\boldsymbol{c}}^{\mathrm{T}} \boldsymbol{x}, \text{ subject to } A\boldsymbol{x} = \boldsymbol{b}, \ \boldsymbol{x} \geq \boldsymbol{0} \tag{18}$$

where the necessary optimality is tested if it is an NBF solution.

3.2.1. Removal of redundant constraints

Nevertheless, it is not easy to remove redundant constraints efficiently. To develop an algorithm that can efficiently identify them, let us review some essential knowledge first:

Definition 3.3. (Redundant Constraint) A constraint **L** in defining a convex polytope Φ is *redundant* if Φ remains the same after the removal of **L**. Moreover, if all vertices of Φ strictly belong to the *interior* of **L**, then **L** is *strictly redundant*.

An illustration is in Figure 2. Before we go ahead to analyse the redundant constraint, it is vital to differentiate it from the *degeneracy*, of which the definition is given as:

Definition 3.4. (Degeneracy) A vertex $\boldsymbol{v} \in \Phi$ is *degenerate* if the number of hyperplanes of Φ passing $\boldsymbol{v}$ is strictly more than n.

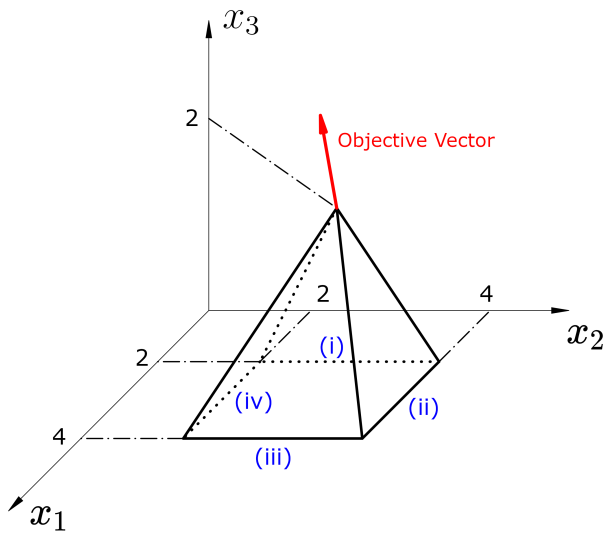


Fig. 3: Degenerate vertex with no redundant constraint.

Namely, since a vertex in $\mathbb{R}^n$ can be exactly determined by n independent hyperplanes, if there exist extra hyperplanes passing the vertex, then it is possible to remove some of them without losing the vertex.

However, it is noted that *the existence of a degenerate vertex is not equivalent to the existence of redundant constraints*. The main reason is that a constraint that can be removed for one vertex may not be removable for another vertex. A typical example is shown in Figure 3, where vertex $(3,3,2)^{\mathrm{T}}$ constrained by (i), (ii), (iii) and (iv) is degenerate but we cannot remove any of them.

Since the degeneracy rarely happens but can cause great trouble, we assume that there exists no degenerate vertex in the convex polytope. As it is mentioned that the redundant constraint should be removed for acceleration, we need an algorithm to identify it efficiently. Moreover, if this process is too complicated to execute, the whole time would be even longer than no redundant constraint removal.

Horst and Tuy [13] proposed a proposition to identify whether a constraint is redundant:

Proposition 3.5. A constraint $\mathbf{L}_j = \{\boldsymbol{c} \in \mathbb{R}^n : l_j(\boldsymbol{c}) \leq 0\}$, $j \in \mathbb{I}_K$ in a convex polytope $\mathbf{P} = \{\boldsymbol{c} \in \mathbb{R}^n : l_i(\boldsymbol{c}) \leq 0, i \in \mathbb{I}_K\}$ is redundant if there exists an $i_0 \in \mathbb{I}_K \backslash \{j\}$ such that

$$\begin{aligned} &\Big\{\boldsymbol{c} \in \mathbb{R}^n : l_j(\boldsymbol{c}) = 0, \big(\forall i \in \mathbb{I}_K \backslash \{i_0\}, l_i(\boldsymbol{c}) \leq 0\big)\Big\} \\ \subseteqq &\Big\{\boldsymbol{c} \in \mathbb{R}^n : l_{i_0}(\boldsymbol{c}) \geq 0\Big\}. \end{aligned} \tag{19}$$

Therefore, we say that the constraint $\mathbf{L}_j$ is redundant for $\mathbf{P}$ relative to $\mathbf{L}_{i_0}$.

Proposition 3.5 gives a method to identify a redundant constraint. However, since it utilises the inclusion relation of subsets, the realisation is extremely difficult. Moreover,

since the vertex set of a convex polytope has a deep relationship with the change of constraints, we prefer updating the vertex set simultaneously. Therefore, it is necessary to develop an algorithm for identifying redundant constraints based on the vertex set of the polytope. Hence, we have the following theorem [13]:

Theorem 3.6. Given a polytope $\mathbf{P} = \{\boldsymbol{c} \in \mathbb{R}^n : l_i(\boldsymbol{c}) \leq 0, i \in \mathbb{I}_K\}$ and a valid cut $\mathbf{L}_k = \{\boldsymbol{c} \in \mathbb{R}^n : l_k(\boldsymbol{c}) \leq 0\}$, let $\mathbf{V}_k^-(\mathbf{P}) := \{\boldsymbol{c} \in \mathbf{V}(\mathbf{P}) : l_k(\boldsymbol{c}) < 0\} \neq \emptyset$ denote the vertex set of $\mathbf{P}$ in the interior of $\mathbf{L}_k$. Then a constraint $\mathbf{L}_j = \{\boldsymbol{c} \in \mathbb{R}^n : l_j(\boldsymbol{c}) \leq 0\}$, $j \in \boldsymbol{K}$ of $\mathbf{P}$ is redundant for $\mathbf{P} \cap \mathbf{L}_k$ relative to $\mathbf{L}_k$ if and only if the following condition is valid:

$$\forall \boldsymbol{c} \in \mathbf{V}_k^-(\mathbf{P}),\ l_j(\boldsymbol{c}) < 0. \tag{20}$$

Furthermore, since the polytope is assumed to be convex, we can use a more simplified one:

Theorem 3.7. Let $\mathbf{P} = \{\boldsymbol{c} \in \mathbb{R}^n : G\boldsymbol{c} \leq \boldsymbol{g}\}$, $G \in \mathbb{R}^{m \times n}$ and $\boldsymbol{g} \in \mathbb{R}^m$, denote a convex polytope, and let $\mathbf{V}(\mathbf{P})$ be its vertex set. Then for the i-th constraint in $\mathbf{P}$ with $\mathbf{L}_i = \{\boldsymbol{c} \in \mathbb{R}^n : G_{i\cdot}\boldsymbol{c} \leq g_i\}$, let $\mathbf{V}_i^=(\mathbf{P}) := \{\boldsymbol{v} \in \mathbf{V}(\mathbf{P}) : G_{i\cdot}\boldsymbol{v} = g_i\}$. If $\mathrm{Card}(\mathbf{V}_i^=(\mathbf{P})) < n$, it is *redundant* for Φ. Moreover, if $\mathrm{Card}(V_i^=(\mathbf{P})) = 0$, i. e., $\forall \boldsymbol{v} \in \mathbf{V},\ G_{i\cdot}\boldsymbol{v} < g_i$, it is *strictly redundant.*

P r o o f . Here we only prove the *redundant* case. Since Definition 3.3 indicates that a constraint is *redundant* if the vertex set remains the same after removing it, we only need to concentrate on the vertex set.

As is known that a vertex in $\mathbb{R}^n$ can only be determined by at least n hyperplanes. Therefore, by Definition 3.4, if a vertex in $\mathbb{R}^n$ is determined exactly by n hyperplane, there exists no degeneracy. Otherwise, if there exist strictly more than n hyperplanes that pass the vertex, there exists degeneracy and the constraints over n can be removed without losing the vertex.

Assume that there exists a hyperplane with less than n constraints. For instance, the i-th constraint such that $\mathbf{L}_i = \{\boldsymbol{c} \in \mathbb{R}^n : G_{i\cdot}\boldsymbol{c} \leq g_i\}$ with the corresponding hyperplane $\{\boldsymbol{c} \in \mathbb{R}^n : G_{i\cdot}\boldsymbol{c} = g_i\}$ and the vertex set $\mathbf{V}_i^=(\mathbf{P}) := \{\boldsymbol{c} \in \mathbf{V} : G_{i\cdot}\boldsymbol{c} = g_i\}$ on it.

Let $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$ denote all vertices in $\mathbf{V}_i^=(\mathbf{P})$. If $k < n$, we can always pick $(n-k)$ vertices from $\mathbf{V}(\mathbf{P}) \backslash \mathbf{V}_i^=(\mathbf{P})$ to form new hyperplanes such that:

(i) The vertex set $\mathbf{V}^*$ such that $\mathbf{V}_i^=(\mathbf{P}) \subsetneqq \mathbf{V}^*$ and $\mathrm{Card}(\mathbf{V}^*) \geq n$ is on the hyperplane.

(ii) The hyperplane is one of the boundaries of the feasible set formed by $\mathbf{P}$ because $\mathbf{P}$ is bounded.

Therefore, it is obvious that $\mathbf{L}_i$ is redundant and can be removed. □

By Theorem 3.6, it is not hard to derive the following two lemmas:

Lemma 3.8. Let $\mathbf{P}$ and $\mathbf{V}(\mathbf{P})$ be given the same as Theorem 3.6. If there exists no constraint such that $\forall \boldsymbol{c} \in \mathbf{V}(\mathbf{P}),\ G_{i\cdot}\boldsymbol{c} < g_i$, or no hyperplane with only degenerate vertices, then there exists no redundant constraint in $\mathbf{P}$.

Proof. If $\forall \boldsymbol{c} \in \mathbf{V}(\mathbf{P})$, $G_i.\boldsymbol{c} < g_i$, there exists no vertex on the hyperplane, hence can be removed. If there exists a hyperplane with only degenerate vertices on it, then by Definition 3.4, for any vertex on it, the hyperplane is removable. Therefore, we can remove it. □

Lemma 3.9. Λ^0 defined by Proposition 3.2 has no redundant constraint or degenerate vertex.

Proof. The polytope constructed by Proposition 3.2 has at most $(n+1)$ constraints with $(n+1)$ vertices. Moreover, the polytope should always be closed and bounded. Since it is obvious that in $\mathbb{R}^n$, a closed and bounded polytope has at least $(n+1)$ hyperplanes, there should at least exist $(n+1)$ constraints. Hence, no redundant constraint exists in Λ^0. On the other hand, the definition of Λ^0 already prevents generating degenerate vertex because it guarantees that there are strictly n hyperplanes passing each vertex of Λ^0. Therefore, there exists neither redundant constraint nor degenerate vertex in Λ^0. □

Lemma 3.9 is obvious and its proof is trivial. However, we still present it since it asserts that at the initial step, we do not need to identify whether there exist redundant constraints or degenerate vertices.

3.2.2. Update of the vertex set

After developing the algorithm for identifying redundant constraints by the vertex set, we come to the process of updating the vertex set after applying a valid cut on a polytope. Since we have assumed that there exists no degenerate vertex in the polytope, let us consider the valid cut expressed as $\mathbf{L} = \{\boldsymbol{c} \in \mathbb{R}^n : \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{c} \leq \boldsymbol{\beta}\}$, where $\boldsymbol{\alpha} \in \mathbb{R}^n$, $\boldsymbol{\beta} \in \mathbb{R}$ on the polytope Λ^k.

Similar to the procedure in removing redundant constraints, we separate $\mathbf{V}(\Lambda^k)$ by $\mathbf{L}$ to $\mathbf{V}^-(\Lambda^k)$ and $\mathbf{V}^+(\Lambda^k)$, where $\mathbf{V}^-(\Lambda^k) := \{\boldsymbol{v} \in \mathbf{V}(\Lambda^k) : \boldsymbol{v} \in \mathbf{L}\}$ and $\mathbf{V}^+(\Lambda^k) := \{\boldsymbol{v} \in \mathbf{V}(\Lambda^k) : \boldsymbol{v} \notin \mathbf{L}\}$. Since the generated vertices on the hyper-plane $\{\boldsymbol{c} \in \mathbb{R}^n : \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{c} = \boldsymbol{\beta}\}$ are the same for both $\mathbf{V}^-(\Lambda^k)$ and $\mathbf{V}^+(\Lambda^k)$, it is convenient to choose the one with fewer vertices, denoted as $\widehat{\mathbf{V}}$.

Since every vertex in $\widehat{\mathbf{V}}$ is strictly constrained by n constraints, a vertex $\boldsymbol{v} \in \widehat{\mathbf{V}}$ is picked out and let $\mathbb{I}_{\boldsymbol{v}}$ with $\mathrm{Card}(\mathbb{I}_{\boldsymbol{v}}) = n$ denote the index of active constraints. Therefore, a polyhedral convex cone on $\boldsymbol{v}$ formed with Λ^k is given by:

$$\left\{ \boldsymbol{c} \in \mathbb{R}^n : \sum_{j=1}^{n} G_{ij}\boldsymbol{c} \leq g_i,\ i \in \mathbb{I}_{\boldsymbol{v}} \right\}. \tag{21}$$

By adding the constraint $\mathbf{L}$, we can form the following tableau:

c_1	c_2	$\dots$	c_n	y_1	y_2	$\dots$	y_n	y_{n+1}	RHS
$G_{i_1 1}$	$G_{i_1 2}$	$\dots$	$G_{i_1 n}$	1	0	$\dots$	0	0	g_{i_1}
$G_{i_2 1}$	$G_{i_2 2}$	$\dots$	$G_{i_2 n}$	0	1	$\dots$	0	0	g_{i_2}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
$G_{i_n 1}$	$G_{i_n 2}$	$\dots$	$G_{i_n n}$	0	0	$\dots$	1	0	g_{i_n}
α_1	α_2	$\dots$	α_n	0	0	$\dots$	0	1	β

(22)

According to the previous setting and the tableau, we have the following conditions:

1. $c_1,\ c_2, \ldots,\ c_n,\ y_{n+1}$ must be the basic variables of $\boldsymbol{v}$.
2. y_{n+1} must not be a basic variable of the generated vertex.
3. $y_1,\ y_2, \ldots,\ y_n,\ y_{n+1} \geq 0$.
4. The generated vertices must be the neighbours of $\boldsymbol{v}$, which means the basis of any generated vertex from $\boldsymbol{v}$ should only be derived by one-step pivoting.

Hence the updated tableau can be written as the one below:

c_1	c_2	$\ldots$	c_n	y_1	y_2	$\ldots$	y_n	y_{n+1}	RHS
1	0	$\ldots$	0	$\check{G}_{i_1 1}$	$\check{G}_{i_1 2}$	$\ldots$	$\check{G}_{i_1 n}$	0	$\check{g}_{i_1}$
0	1	$\ldots$	0	$\check{G}_{i_2 1}$	$\check{G}_{i_2 2}$	$\ldots$	$\check{G}_{i_2 n}$	0	$\check{g}_{i_2}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
0	0	$\ldots$	1	$\check{G}_{i_n 1}$	$\check{G}_{i_n 2}$	$\ldots$	$\check{G}_{i_n n}$	0	$\check{g}_{i_n}$
0	0	$\ldots$	0	$\check{G}_{i_{n+1} 1}$	$\check{G}_{i_{n+1} 2}$	$\ldots$	$\check{G}_{i_{n+1} n}$	1	$\hat{\beta}$

(23)

As a result, a generated vertex should satisfy the following conditions.

1. $c_1,\ c_2, \ldots,\ c_n$ must be in the basis and exactly one state in $y_1,\ y_2, \ldots,\ y_n$ is in the basis.
2. The solution of the basic state derived from $y_1,\ y_2, \ldots,\ y_n$ should be strictly positive.

Therefore, we can generate all new vertices on the extra constraint by repeating the procedure for all $\boldsymbol{v} \in \widehat{\mathbf{V}}$ iteratively. However, we note that not all of these generated vertices are the vertices of Λ^{k+1}, because only a part of constraints of Λ^k are considered. Therefore, the confirmation of the satisfaction of all constraints of Λ^k is essential. Only the generated vertices satisfying all constraints of Λ^k are added to the vertex set of Λ^{k+1}.

4. ALGORITHM FOR ROBUST OPTIMALITY ANALYSIS

After proposing the theorem and method concerned with the initialisation and iterations, we propose the following algorithm:

Algorithm 4.1. Robust Optimality Analysis

- **Initialisation**:
  1. Check whether $\Phi = \{\boldsymbol{c} \in \mathbb{R}^n : G\boldsymbol{c} \leq \boldsymbol{g}\}$ is bounded by Proposition 3.2. If it is not, terminate the procedure with unboundedness. Otherwise, build the initial polytope with $\Lambda^0 := \{\boldsymbol{c} \in \mathbb{R}^n : \boldsymbol{e}^{\mathrm{T}}\boldsymbol{c} \leq \alpha_0, c_i \geq \alpha_i, i = 1, 2, ..., n\} = \{\boldsymbol{c} \in \mathbb{R}^n : G^0\boldsymbol{c} \leq \boldsymbol{g}^0\}$ with its vertex set $\mathbf{V}(\Lambda^0)$ containing $(n+1)$ entries.
  2. Choose the median point of all vertices in $\mathbf{V}(\Lambda^0)$ as the OFC vector and obtain an NBF solution $\boldsymbol{x}^*$, which gives the OAC as $\mathcal{M}(\boldsymbol{x}^*) := \{\boldsymbol{c} \in \mathbb{R}^n : \boldsymbol{c}_N - A_N^{\mathrm{T}} A_B^{-\mathrm{T}} \boldsymbol{c}_B \leq \mathbf{0}\} = \{\boldsymbol{c} \in \mathbb{R}^n :\ M\boldsymbol{c} \leq \mathbf{0}\}$.

- **Iteration** $k = 0, 1, 2, 3, ...$

  1. Check whether $\mathbf{V}(\Lambda^k) \subseteqq \mathcal{M}(\boldsymbol{x}^*)$. If it is, terminate with $\boldsymbol{x}^*$ being necessarily optimal. Otherwise, go next.
  2. Pick a vertex $\boldsymbol{v} \in \mathbf{V}(\Lambda^k)$ such that $\boldsymbol{v} \notin \mathcal{M}(\boldsymbol{x}^*)$. If $\boldsymbol{v} \in \Phi$, terminate with no necessarily optimal solution exists. Otherwise, go next.
  3. (*Adding a New Constraint*) Choose a constraint in Φ as $\mathbf{L}_i = \{\boldsymbol{c} \in \mathbb{R}^n : G^k_{i\cdot}\boldsymbol{c} \leq \boldsymbol{g}^k_i\}$ such that $\boldsymbol{v} \notin \mathbf{L}_i$. (we can update Φ by removing $\mathbf{L}_i$ from it.) Add $\mathbf{L}_i$ to Λ^k to obtain Λ^{k+1}:
  $$\Lambda^{k+1} = \Lambda^k \cap \mathbf{L}_i = \left\{\boldsymbol{c} \in \mathbb{R}^n : G^{k+1}\boldsymbol{c} \leq \boldsymbol{g}^{k+1}\right\}. \tag{24}$$
  4. (*Separate the Vertex Set*) Separate $\mathbf{V}(\Lambda^k)$ by $\mathbf{L}_i$ and obtain:
  $$\mathbf{V}^+(\Lambda^k) = \left\{\boldsymbol{c} \in \mathbf{V}(\Lambda^k) : G^{k+1}_{i\cdot}\boldsymbol{c} > \boldsymbol{g}^{k+1}_i\right\}, \tag{25}$$
  $$\mathbf{V}^-(\Lambda^k) = \left\{\boldsymbol{c} \in \mathbf{V}(\Lambda^k) : G^{k+1}_{i\cdot}\boldsymbol{c} \leq \boldsymbol{g}^{k+1}_i\right\}. \tag{26}$$
  5. (*Update the Vertex Set*) Choose the vertex set of $\mathbf{V}^-(\Lambda^k)$ and $\mathbf{V}^+(\Lambda^k)$ that has fewer entries and denote it as $\widehat{\mathbf{V}}_k$. For every $\boldsymbol{v}^j \in \widehat{\mathbf{V}}_k$, generate new vertices on the hyperplane of $\mathbf{L}_i$ by pivoting Tableau (22) and denote the vertex set as $\mathbf{V}(\boldsymbol{v}^j)$. Then obtain the new vertex set $\mathbf{V}(\Lambda^{k+1})$ with:
  $$\mathbf{V}(\Lambda^{k+1}) = \mathbf{V}^-(\Lambda^k) \bigcup \left(\cup_j \mathbf{V}(\boldsymbol{v}^j) \right) \tag{27}$$
  6. (*Remove the Redundant Constraint*) Apply Theorem 3.7 to Λ^{k+1} with vertex set $\mathbf{V}^-(\Lambda^k)$ and update Λ^{k+1}. For each constraint $\mathbf{L}_j$ in Λ^{k+1}, let $\mathbf{V}^=_j(\Lambda^{k+1}) = \{\boldsymbol{v} \in \mathbf{V}(\Lambda^{k+1}) : G^{k+1}_{j\cdot}\boldsymbol{c} = \boldsymbol{g}_j\}$. If $\mathrm{Card}(\mathbf{V}^=_j(\Lambda^{k+1})) < n$, remove $\mathbf{L}_j$ from Λ^{k+1}.
  7. Repeat the iteration with $k \leftarrow k + 1$.

To illustrate Algorithm 4.1, let us reuse Example 2.7 but change some situations.

Example 4.2. Let us reconsider the production problem in Example 2.7. This time we assume that A and B are manufactured in Japan, but A is exported to Australia and B is exported to the USA. The manufacturing condition, i. e., the constraints, is the same as Example 2.7. The price of A is specified in AUD, and the price of B is specified in USD, while we consider the gross profit in JPY. Since A and B are exported to different foreign countries, their profits are also affected by the exchange rate. From past experience, the profit of A, i. e., c_1, is never beyond 5 thousand JPY and the profit of B, i. e., c_2, is never beyond 3.5 thousand JPY. Moreover, in the past five years, it has been observed that c_1 and c_2 satisfied a linear constraint with $3c_1 + 4c_2 \geq 23$, where 23 denotes 23 thousand JPY. Then, we would like to know the amounts of products A and B that maximise the gross profit.

At first, we convert the problem to an LP problem as follows:

$$\begin{array}{rl} \text{maximize} & c_1x_1 + c_2x_2, \\ \text{subject to} & 3x_1 + 4x_2 \leq 42, \\ & 3x_1 + x_2 \leq 24, \\ & x_2 \leq 9,\ x_1, x_2 \geq 0, \end{array} \tag{28}$$

with c_1, c_2 satisfying

$$\left\{ \left(c_1, c_2\right)^{\mathrm{T}} : 3c_1 + 4c_2 \geq 23, c_1 \leq 5, c_2 \leq 3.5 \right\} \tag{29}$$

Similar to Example 2.7, we solve the problem in Figure 4 by following the step in Algorithm 4.1 (It is noted that we ignore the process of adding and removing slack variables):

Initialisation: Since the convex polytope $\Phi = \{\boldsymbol{c} \in \mathbb{R}^2 : G\boldsymbol{c} \leq \boldsymbol{g}\}$, where

$$G = \begin{pmatrix} -3 & -4 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \boldsymbol{g} = \begin{pmatrix} -23 \\ 5 \\ 3.5 \end{pmatrix},$$

we can verify its boundedness, which is a polytope formed by Constraint (1), (2) and (3). As the result, we obtain the initial polytope $\Lambda^0 = \{\boldsymbol{c} \in \mathbb{R}^2 : G^0\boldsymbol{c} \leq \boldsymbol{g}^0\}$, where

$$G^0 = \begin{pmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \text{ and } \boldsymbol{g}^0 = \begin{pmatrix} 8.5 \\ -3 \\ -2 \end{pmatrix},$$

formed by Constraint (4), (5) and (6). The vertex set can also be obtained as $\mathbf{V}(\Lambda^0) = \{(3,2)^{\mathrm{T}}, (6.5,2)^{\mathrm{T}}, (3,5.5)^{\mathrm{T}}\}$. Therefore, the median point of $\mathbf{V}(\Lambda^0)$ is $(\frac{25}{6}, \frac{19}{6})^{\mathrm{T}}$, which derives the NBF solution as $\boldsymbol{x}^* = (6,6)^{\mathrm{T}}$ with the OAC:

$$\mathscr{M}(\boldsymbol{x}^*) = \left\{ (c_1, c_2)^{\mathrm{T}} \;\middle|\; \begin{array}{c} \frac{1}{9}c_1 - \frac{1}{3}c_2 \leq 0 \\ -\frac{4}{9}c_1 + \frac{1}{3}c_2 \leq 0 \end{array} \right\}.$$

Iteration $k = 0$:

1. Since $(6.5,2)^{\mathrm{T}} \notin \mathscr{M}(\boldsymbol{x}^*)$, $\mathbf{V}(\Lambda^0) \nsubseteq \mathscr{M}(\boldsymbol{x}^*)$, hence go next;
2. Since $(6.5,2)^{\mathrm{T}} \notin \Phi$, hence go next;
3. Since $(6.5,2)^{\mathrm{T}}$ is out of Constraint (2), add it to Λ^0 and update to Λ^1 with Constraint (4), (5), (6) and (2);
4. Separate $\mathbf{V}(\Lambda^0)$ and obtain
$$\mathbf{V}^+(\Lambda^0) = \{(6.5,2)^{\mathrm{T}}\} \text{ and } \mathbf{V}^-(\Lambda^0) = \{(3,2)^{\mathrm{T}}, (3,5.5)^{\mathrm{T}}\};$$
5. By pivoting, we can generate new vertices $(5,2)^{\mathrm{T}}$ and $(5,3.5)^{\mathrm{T}}$. Hence update vertex set $\mathbf{V}(\Lambda^1) = \{(3,2)^{\mathrm{T}}, (3,5.5)^{\mathrm{T}}, (5,2)^{\mathrm{T}}, (5,3.5)^{\mathrm{T}}\}$;
6. Since every constraint in Λ^1 strictly contains 2 vertices, there exists no redundant constraint. Hence go to the next iteration.

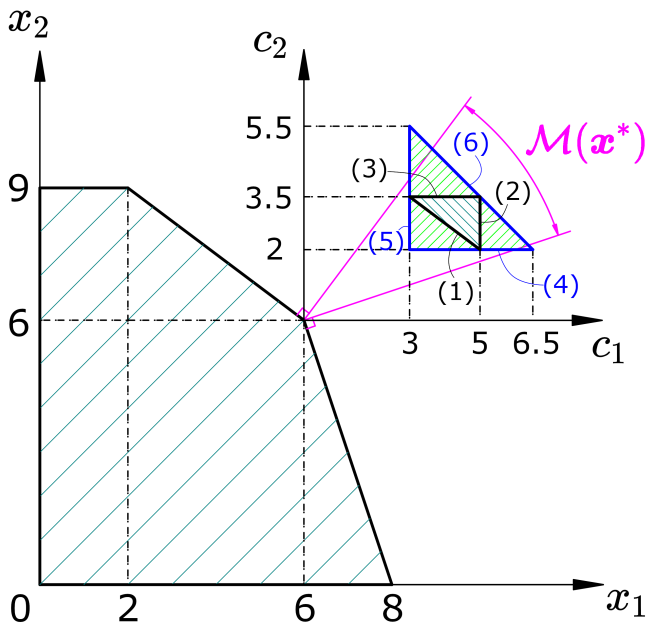


Fig. 4: Example 4.2.

Iteration $k = 1$:

1. Since $(3, 5.5)^{\mathrm{T}} \notin \mathscr{M}(\boldsymbol{x}^*)$, $\mathbf{V}(\Lambda^1) \nsubseteqq \mathscr{M}(\boldsymbol{x}^*)$, hence go next;
2. Since $(3, 5.5)^{\mathrm{T}} \notin \Phi$, hence go next;
3. Since $(3, 5.5)^{\mathrm{T}}$ is out of Constraint (3), add it to Λ^1 and update to Λ^2 with Constraint (4), (5), (6), (2) and (3);
4. Separate $\mathbf{V}(\Lambda^1)$ and obtain
$$\mathbf{V}^+(\Lambda^1) = \{(3, 5.5)^{\mathrm{T}}\} \text{ and } \mathbf{V}^-(\Lambda^1) = \{(3, 2)^{\mathrm{T}}, (5, 2)^{\mathrm{T}}, (5, 3.5)^{\mathrm{T}}\};$$
5. By pivoting, we can generate new vertices $(3, 3.5)^{\mathrm{T}}$ and $(5, 3.5)^{\mathrm{T}}$. Hence $\mathbf{V}(\Lambda^2) = \{(3, 2)^{\mathrm{T}}, (5, 2)^{\mathrm{T}}, (5, 3.5)^{\mathrm{T}}, (3, 3.5)^{\mathrm{T}}\}$;
6. Since Constraint (6) contains only 1 vertex $(5, 3.5)^{\mathrm{T}}$, remove it and update Λ^2 with Constraint (4), (5), (2) and (3). Then go to the next iteration.

Iteration $k = 2$:

1. Since $\mathbf{V}(\Lambda^2) \subseteqq \mathscr{M}(\boldsymbol{x}^*)$, terminate the procedure and state that $\boldsymbol{x}^* = (6, 6)^{\mathrm{T}}$ to be necessarily optimal.

Therefore, we verify that the solution $(6, 6)^{\mathrm{T}}$ is the schedule that is not affected by the profits of A and B and can always make the biggest profit.

5. COMPUTATIONAL ANALYSIS AND SIMULATIONS

In this section, we analyse the computational complexity of our approach compared with the conventional vertex-listing method. Before going ahead, we refer the LP problem with interactive uncertain OFCs in the form of Problem (3) to the *LP problem with uncertain OFCs within a* $(p \times n)$*-scale convex polytope*, where $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{\gamma} \in \Phi$. $\Phi \subsetneqq \mathbb{R}^n$ is the convex polytope defined by (13), i. e., $\Phi = \{\boldsymbol{c} \in \mathbb{R}^n : G\boldsymbol{c} \leq \boldsymbol{g}\}$, where $G \in \mathbb{R}^{p\times n}$ and $\boldsymbol{g} \in \mathbb{R}^p$.

At first, we consider the conventional vertex-listing method, of which the algorithm follows as:

(i) List all vertices of Φ and denote the set as $\mathbf{V}(\Phi)$.

(ii) Obtain an NBF solution $\boldsymbol{x}^*$ with $\mathscr{M}(\boldsymbol{x}^*)$ by the median point of $\mathbf{V}(\Phi)$.

(iii) Check whether all entries in $\mathbf{V}(\Phi)$ are in $\mathscr{M}(\boldsymbol{x}^*)$.

We can see that the algorithm is not difficult in Step (ii) and (iii). However, Step (i) listing all vertices of Φ requires enormous computational effort.

As we have mentioned that the maximum vertex number of a convex polytope defined in (13) is $O(p^{n/2})$, we use it as the worst complexity order for listing all vertices. Moreover, in each iteration of listing a new vertex, the matrix inversion calculation is involved, of which the worst complexity order is $O(n^3)$. It is noted that the complexity order of the matrix inversion becomes lower and lower in recent years, which, however, lacks a milestone to determine the final result. To avoid the relevant trouble, we assume the matrix inversion is of $O(n^3)$ complexity order. Therefore, the complexity order of the vertex-listing method is

$$O(p^{n/2}) \cdot O(n^3) = O(p^{n/2} \cdot n^3). \tag{30}$$

Then, let us analyse the proposed approach. In the initialisation, it is necessary to solve the LP problem for $(n + 1)$ times to find the initial polytope with the vertex set, after which an NBF solution with its OAC is obtained.

In the iteration process, we need multiple assumptions to accomplish the analysis. For example, although it is known that the number of iterations is up to p, the algorithm usually terminates at a much earlier step. Moreover, unlike the vertex-listing method, the number of vertices and removable redundant constraints at each iteration are unknown. Therefore, we can only analyse its worst case approximately, where:

(i) The number of iterations equals its maximum, i. e., p.

(ii) Only one vertex is removed at each iteration. However, when separating the vertex set, we assume all vertices are used, including the removed one.

(iii) No redundant constraint is removed during all iterations.

(iv) All vertices generated by a cut, of which the number should be n, are assumed to be valid.

m	n	p	$T_1/(s)$	$T_2/(s)$	Nec.	m	n	p	$T_1/(s)$	$T_2/(s)$	Nec.
3	5	10	0.0698	0.0539	True	4	7	18	2.3431	0.1839	True
3	5	12	0.1118	0.0526	True	4	7	18	2.1856	0.1408	True
3	5	12	0.1157	0.0638	True	4	7	20	5.4029	0.1502	True
3	5	12	0.1825	0.0992	True	4	7	20	5.3161	0.1621	True
3	5	12	0.1828	0.0738	True	4	7	20	5.2396	0.1856	True
3	6	15	0.5978	0.1632	True	4	8	21	11.5032	0.2000	True
3	6	15	0.491	0.1519	True	4	8	24	40.4563	0.2802	True
3	6	15	0.4212	0.0968	True	4	8	25	82.0311	0.2356	True
3	6	15	0.4165	0.0919	True	4	8	25	71.5362	0.2114	True
3	6	16	0.6579	0.136	True	4	8	25	59.9396	0.2183	True

Tab. 1: Some simulation results.

Since the time spent on checking whether a vertex belongs to a set is too short compared with the update process of vertex sets, we ignore it.

By Tableau (22) and (23), it is not hard to understand that the process of pivoting an $(l \times k)$ tableau has the same computational complexity as solving an $(l \times k)$-scale LP problem. Moreover, by algebraic analysis, it is known that the complexity order of solving an $(l \times k)$-scale LP problem equals the one of a non-singular $(l \times l)$ matrix inversion when $l \leq k$. Therefore, the complexity order of the proposed algorithm is

$$O\left(\sum_{j=1}^{p}\left[(n+2)+(j-1)\cdot(n-1)\right]\right)\cdot O(n^3) = O(p^2\cdot n^4). \tag{31}$$

Since in a large-scale problem p is much greater than n and n is much greater than 4, the results of (30) and (31) show that, even in the worst case, our approach can still treat the robust optimality analysis better than the vertex-listing method.

Furthermore, to validate our approach by comparisons generally, we ran a series of computer simulations, which are carried out by `Python3.8.9` with `numpy1.22.1` and `scipy1.7.3` in `MacOS` with `Intel(R) Core(TM) i7-4770 HQ CPU@2.20GHz` and `16GB 1600MHz DDR3 RAM`.

Since the existence of a necessarily optimal solution usually requires more computational effort than a non-existence case, we only show the former case. The result is shown in Table 1, where m and n denote the scale of LP problems defined in (3), p denotes the number of the constraints of the convex polytope in (13).

The result shows that our approach has a better performance even when the scale of the LP problem is not large. Moreover, when the scale becomes too large to be solved by the vertex-listing method, our approach can still deal with it in a reasonable time.

On the other hand, if we focus on the complexity of the proposed approach, we have Figure 5 by another series of simulations. We can find that its expected computational amount is almost linear to the scale of the LP problem. However, the simulation also indicates that our approach is not always better than the conventional one. When there exists degeneracy in the convex polytope, our approach can only terminate without giving any result. For example, in Figure 5 at about 1500-scale, the computational time becomes very small because the program encounters a degenerate case and has to give up the computation.

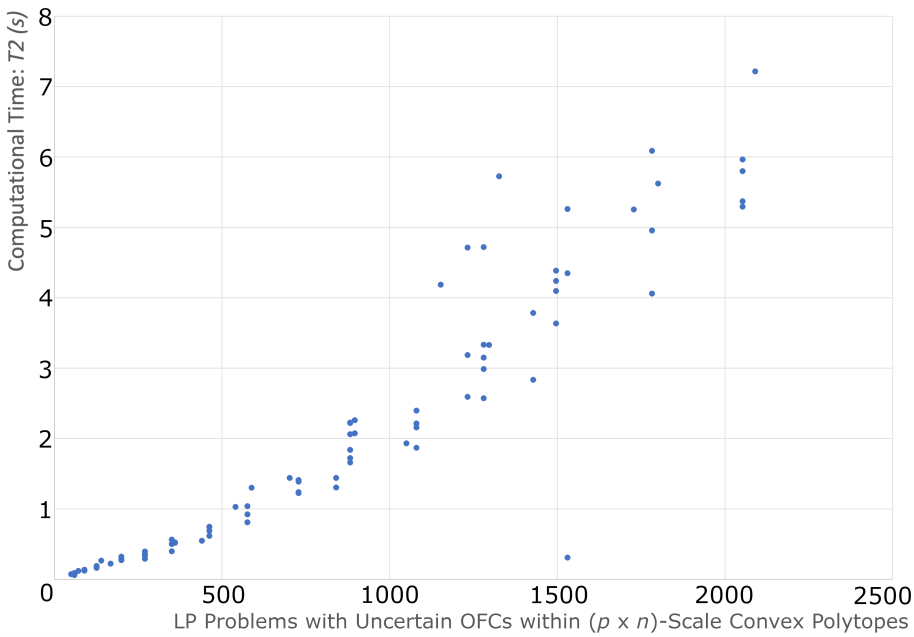


Fig. 5: The simulation result of the outer approximation approach.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a robust optimality analysis in LP problems with uncertain OFCs. We investigated a method for testing whether an NBF solution is optimal for all conceivable OFCs when a convex polytope represents the set of conceivable uncertain OFC vectors. Such a test problem is called a robust optimality test problem.

We reviewed the tolerance approach to LP problems and described that the robust optimality test problem can be solved easily by utilising the tolerance approach when the set of conceivable uncertain OFC vectors is represented by a hyper-box, i. e., when the marginal ranges of some OFCs stay unchanged no matter what values the remaining OFCs take. However, in other cases, we showed that the computational complexity of the robust optimality test problem would grow enormously as it may require a vertex-listing process. Therefore, we proposed an algorithm based on the outer approximation to the robust optimality test problem when a convex polytope represents the set of conceivable uncertain OFC vectors.

In the proposed approach, we built a simplex (cover) including the range of uncertain OFCs and initialised the vertex set with the vertices of the simplex. Then we approximate the cover to the given polytope by cutting it with a linear constraint composing the polytope, which is not satisfied with a member of the vertex set. Using the added linear constraint, we update the vertex set and continue this procedure until all members of the vertex set are in the OAC or we find a point of a polytope not in the OAC. We demonstrated the computational advantage of the proposed approach over a conventional vertex-listing method through a numerical experiment. The result showed that

the proposed approach is more advantageous in large-scale problems.

In the proposed approach, we assumed the non-degeneracy of the basic feasible solution. It is possible to extend the proposed approach to the robust optimality test problem at a degenerate basic feasible solution. Moreover, we also assumed that the degeneracy never occurs at any vertex of the convex polytope showing the range of uncertain OFCs. To overcome these limitations would be our future work.

ACKNOWLEDGEMENT

This work was supported by JSPS KAKENHI Grant Number JP18H01658.

(Received May 2, 2022)

REFERENCES

[1] S. P. Bradley, A. C. Hax, and T. L. Magnanti: Applied Mathematical Programming. Addison-Wesley Publishing Company, 1977.

[2] S. Curry, I. Lee, S. Ma, and N. Serban: Global sensitivity analysis via a statistical tolerance approach. Europ. J. Oper. Res. *296* (2022), 1, 44–59. DOI:10.1016/j.ejor.2021.04.004

[3] G. Dantzig: Linear programming and extensions. In: Linear programming and extensions. Princeton Zniversity Press, 2016.

[4] C. Filippi: A fresh view on the tolerance approach to sensitivity analysis in linear programming. Europ. J. Oper. Res. *16* (2005), 1, 1–19. DOI:10.1016/j.ejor.2004.01.050

[5] Z. Gao and M. Inuiguchi: Estimating the optimal probability of a candidate basic solution in stochastic linear programming. In: 60th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), IEEE 2021, pp. 640–643.

[6] Z. Gao and M. Inuiguchi: An analysis to treat the degeneracy of a basic feasible solution in interval linear programming. In: The Ninth International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making (IUKM 2022). Publ. in Lecture Notes in Computer Science pp. 130–142, 2022. DOI:10.1007/978-3-030-98018-4_11

[7] E. Garajová and M. Hladík: On the optimal solution set in interval linear programming. Comput. Optim. Appl. *72* (2019), 1, 269–292. DOI:10.1007/s10589-018-0029-8

[8] T. L. Heath et al.: The works of Archimedes. Courier Corporation, 2002.

[9] M. Henk, J. Richter-Gebert, and G. M. Ziegler. Basic properties of convex polytopes. In J. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Edition*, pages 243–270, Boca Raton, FL, 2004. CRC Press.

[10] M. Hladík: Multiparametric linear programming: support set and optimal partition invariancy. Europ. J. Oper. Res. *202* (2010), 1, 25–31, 2010. DOI:10.1016/j.ejor.2009.04.019

[11] M. Hladík: Complexity of necessary efficiency in interval linear programming and multiobjective linear programming. Optim. Lett. *6* (2012), 5, 893–899. DOI:10.1007/s11590-011-0315-1

[12] R. Horst, J. De Vries, and N. V. Thoai: On finding new vertices and redundant constraints in cutting plane algorithms for global optimization. Oper. Res. Lett. *7* (1988), 2, 85–90. DOI:10.1016/0167-6377(88)90071-5

[13] R. Horst and H. Tuy: Global optimization: Deterministic approaches. Springer Science and Business Media, 2013.

[14] M. Inuiguchi: Necessary efficiency is partitioned into possible and necessary optimalities. In: 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), IEEE 2013, pp. 209–214. DOI:10.1109/IFSA-NAFIPS.2013.6608401

[15] M. Inuiguchi, Z. Gao, and C. O. Henriques: Robust optimality analysis of non-degenerate basic feasible solutions in linear programming problems with fuzzy objective coefficients. Fuzzy Optimization and Decision Making *22* (2023), 51–79. DOI:10.1007/s10700-022-09383-2

[16] M. Inuiguchi and J. Ramík: Possibilistic linear programming: a brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem. Fuzzy Sets Systems *111* (2000), 1, 3–28. DOI:10.1016/S0165-0114(98)00449-7

[17] M. Inuiguchi and M. Sakawa: Possible and necessary efficiency in possibilistic multiobjective linear programming problems and possible efficiency test. Fuzzy Sets Systems *78* (1996), 2, 231–241. DOI:10.1016/0165-0114(95)00169-7

[18] M. Inuiguchi and M. Sakawa: An achievement rate approach to linear programming problems with an interval objective function. J. Oper. Res. Soc. *48* (1997), 1, 25–33. DOI:10.1038/sj.jors.2600322

[19] M. Inuiguchi and M. Sakawa: Robust optimization under softness in a fuzzy linear programming problem. Int. J. Approx. Reas. *18* (1998), 1–2, 21–34. DOI:10.1016/s0888-613x(97)10002-0

[20] B. Jansen, J. De Jong, C. Roos, and T. Terlaky: Sensitivity analysis in linear programming: just be careful! Europ. J. Oper. Res. *101* (1997), 1, 15–28. DOI:10.1016/s0377-2217(96)00172-5

[21] P. Kall and J. Mayer: Stochastic Linear Programming: Models, Theory, and Computation. Second Edition. Springer, Boston 2011.

[22] M. J. Todd: Probabilistic models for linear programming. Math. Oper. Res. *16* (1991), 4, 671–693. DOI:10.1287/moor.16.4.671

[23] R. E. Wendell: The tolerance approach to sensitivity analysis in linear programming. Management Sci. *31* (1985), 5, 564–578. DOI:10.1287/mnsc.31.5.564

[24] R. E. Wendell: Tolerance sensitivity and optimality bounds in linear programming. Management Sci. *50* (2004), 6, 797–803. DOI:10.1287/mnsc.1030.0221

[25] F. R. Wondolowski Jr.: A generalization of wendell's tolerance approach to sensitivity analysis in linear programming. Decision Sci. *22* (1991), 4, 792–811. DOI:10.1111/j.1540-5915.1991.tb00365.x

Zhenzhong Gao, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531. Japan.
e-mail: zhenzhong@inulab.sys.es.osaka-u.ac.jp

Masahiro Inuiguchi, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531. Japan.
e-mail: inuiguti@sys.es.osaka-u.ac.jp