Discrete Comput Geom 37:351–367 (2007)
DOI: 10.1007/s00454-006-1300-9

Discrete & Computational
Geometry

Dynamic Enumeration of All Mixed Cells

Tomohiko Mizutani, Akiko Takeda, and Masakazu Kojima

Department of Mathematical and Computing Sciences, Tokyo Institute of Technology,
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552, Japan
{mizutan8,takeda,kojima}@is.titech.ac.jp

Abstract. The polyhedral homotopy method, which has been known as a powerful numerical method for computing all isolated zeros of a polynomial system, requires all mixed cells of the support of the system to construct a family of homotopy functions. The mixed cells are reformulated in terms of a linear inequality system with an additional combinatorial condition. An enumeration tree is constructed among a family of linear inequality systems induced from it such that every mixed cell corresponds to a unique feasible leaf node, and the depth-first search is applied to the enumeration tree for finding all the feasible leaf nodes. How to construct such an enumeration tree is crucial in computational efficiency. This paper proposes a dynamic construction of an enumeration tree, which branches each parent node into its child nodes so that the number of feasible child nodes is expected to be small; hence we can prune many subtrees which do not contain any mixed cell. Numerical results exhibit that the proposed dynamic construction of an enumeration tree works very efficiently for large scale polynomial systems; for example, it generated all mixed cells of the cyclic-15 problem for the first time in less than 16 hours.

1. Introduction

The polyhedral homotopy continuation method [14], which is based on Bernshtein's theorem [1], is known to be a powerful numerical method [6], [11], [12], [15], [16], [23] for computing all isolated zeros of a polynomial system $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$. Let $\mathbb{R}$ and $\mathbb{C}$ be the set of real and complex numbers, respectively. Each $f_i(\boldsymbol{x})$ denotes a complex-valued polynomial in a variable vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{C}^n$. Let $\mathbb{Z}_+^n$ denote the set of nonnegative integer vectors in $\mathbb{R}^n$. We represent each component polynomial $f_i(\boldsymbol{x})$ as

$$f_i(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in \mathcal{A}_i} c_i(\boldsymbol{a})\, \boldsymbol{x}^{\boldsymbol{a}},$$

for some nonempty finite subset $\mathcal{A}_i$ of $\mathbb{Z}_+^n$ and some nonzero $c_i(\boldsymbol{a}) \in \mathbb{C},\ \boldsymbol{a} \in \mathcal{A}_i$. Here $\boldsymbol{x}^{\boldsymbol{a}} = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$ for $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{Z}_+^n$. The set $\mathcal{A}_i$ consists of m_i elements, and is called the support of $f_i(\boldsymbol{x})$. Also, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$ is called the support of $\boldsymbol{f}(\boldsymbol{x})$. In this paper we focus on a fully mixed polynomial system where all supports $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ are all distinct. See [9] and [14] for the semi-mixed case where some of them are identical to each other.

Enumeration of all mixed cells of the support $\mathcal{A}$ of a polynomial system $\boldsymbol{f}(\boldsymbol{x})$, which is the subject of this paper, plays an essential role in the polyhedral homotopy method. Using the mixed cells, we construct a family of polyhedral homotopy functions between start systems, which are auxiliary polynomial systems whose zeros can be computed easily, and the target system $\boldsymbol{f}(\boldsymbol{x})$. Starting from zeros of every start system, we then trace all curves of zeros, so-called homotopy paths, of every polyhedral homotopy function. Bernshtein's theorem [1] guarantees that the mixed volume, which is equal to the total volume of all mixed cells and coincides with the total number of solutions of the start systems, provides an upper bound for the number of isolated solutions of the target system $\boldsymbol{f}(\boldsymbol{x})$ in $(\mathbb{C}\backslash\{0\})^n$. Especially if the target system is in general position, i.e., the coefficients of the target system are generic, this bound is tight and all homotopy paths reach isolated solutions of the target system. Some formulations were proposed for finding all mixed cells. Among others, the formulation of finding all mixed cells as a system of linear inequalities with a certain additional combinatorial condition [8], [9], [17], [21] (or a family of systems of linear inequalities) is more efficient in computational time and memory requirement than the geometric formulation used in [23].

This paper is founded on the former formulation. In this formulation, an enumeration tree is constructed among a family of systems of linear inequalities, which are induced from the system of linear inequalities with a combinatorial condition describing all mixed cells. The enumeration tree satisfies the following properties:

(i) A leaf node describes a mixed cell if and only if it is feasible (or more precisely the system of linear inequalities attached to the leaf node is feasible).
(ii) Each mixed cell is corresponding to a unique feasible leaf node.
(iii) Each node different from leaf nodes is a common subsystem of its child nodes, so that if it is infeasible then so are all of its descendant nodes.
(iv) The root node is an empty system, which is always feasible.

We then apply an enumeration method for finding all feasible leaf nodes; if a node is determined to be infeasible then so is its descendant nodes; hence the subtree having the node as a root can be pruned because it does not contain any mixed cell.

There are two important issues in the efficient implementation of the enumeration of all mixed cells which correspond to feasible leaf nodes of an enumeration tree satisfying properties (i)–(iv). One is how we check the feasibility of each node. For this purpose, [8], [9], [17] and [21] utilize a linear programming (LP) problem having a linear system of inequalities attached to each node as its constraint and check the feasibility of the linear system. Papers [8], [9] and [17] apply the primal simplex method to the LP problem while [21] employs the dual simplex method. If we take account of the effective use of information obtained at a node for its child nodes, the dual simplex method has an

advantage. Specifically, we can easily choose a feasible solution of the dual of the child LP from an optimal solution of the dual of the parent LP. At least, application of the dual simplex method is popular in the field of optimization to deal with such a situation effectively [18]. This paper also applies the dual simplex method to an LP problem attached to each node to check its feasibility, which is described as the application of the primal simplex method to the dual of the LP problem.

The other important issue is how we construct enumeration trees. Enumeration trees need to satisfy properties (i)–(iv) as we mentioned above. Specifically, the root node is fixed to be an empty system of linear inequalities by property (iv), and properties (i) and (ii) determine the collection of leaf nodes. One has much freedom in choosing and allocating systems of linear inequalities, which are induced from the system of linear inequalities with a combinatorial condition describing the mixed cells, for intermediate level nodes. In the existing works [8], [9], [17], [21], the structure of the enumeration trees is determined and fixed before enumerating mixed cells. In such a static construction of an enumeration tree, any information obtained at a node during the execution of enumeration is never utilized at all to branch the node into its child nodes because a branching rule is fixed in advance. For numerical efficiency, however, it is ideal to branch the node into its child nodes so that a larger portion of its child nodes are infeasible and are pruned. To pursue this idea, this paper proposes dynamic enumeration where branching at a node is carried out with the effective use of information which is obtained from the dual simplex method applied to some "child LP problems" at the node; hence the dual simplex method plays an essential role in this situation too. We note that dynamic enumeration is often utilized in the branch-and-bound method for integer programs [18].

Numerical results exhibit that our dynamic enumeration method works very efficiently for finding all mixed cells in comparison with existing static enumeration methods [7], [9], [10], [17], [21], [23]. For instance, our dynamic enumeration method generated all mixed cells of the cyclic-14 problem and computed the mixed volume, which had been the largest one in cyclic-n problems solved by the existing methods, in 1 hour 36 minutes, while MixedVol [9], [10], which is known as the fastest software among the existing ones, solves the same problem in 7 hours 14 minutes. Furthermore, our method computed the mixed volume for the cyclic-15 problem through finding all mixed cells of the problem in 15 hours 45 minutes. It appears that this is the first report for the mixed volume of this problem. As for noon-n and chandra-n problems, it is shown that the speedup ratio between the computational times of our method and MixedVol increases as the size of these problems becomes larger.

This paper is organized as follows. In Section 2 we describe a procedure for the construction of an enumeration tree satisfying properties (i)–(iv), and then outline our dynamic enumeration algorithm. Section 3 is devoted to technical details of the algorithm. We first show an LP formulation for checking the feasibility of each node in an enumeration tree, and then discuss the size of the primal–dual pair of LP problems. We then present how we branch each parent node into its child nodes so that the number of feasible child nodes is expected to be small. This is a core part of the dynamic enumeration algorithm. In Section 4 we show numerical results for some benchmark polynomial systems.

2. An Outline of the Dynamic Enumeration Algorithm for Finding All Mixed Cells

2.1. *The Mixed Volume and Mixed Cells of the Support Set*

Let $N := \{1, 2, \ldots, n\}$. For every $i \in N$, let $Q_i = \mathrm{conv}(\mathcal{A}_i)$ denote the convex hull of the support set $\mathcal{A}_i$ of the polynomial $f_i(\boldsymbol{x})$. For all positive number $\lambda_1, \lambda_2, \ldots, \lambda_n$, we consider the n-dimensional volume of the Minkowski sum

$$\lambda_1 Q_1 + \lambda Q_2 + \cdots + \lambda_n Q_n = \{\lambda_1 q_1 + \lambda_2 q_2 + \cdots + \lambda_n q_n \colon q_i \in Q_i,\ i \in N\}.$$

This volume is known to be a homogeneous polynomial of degree n in $\lambda_1, \lambda_2, \ldots, \lambda_n$. The mixed volume of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$ is defined as the coefficient of $\lambda_1 \lambda_2 \cdots \lambda_n$ in the polynomial.

Huber and Sturmfels [14] introduced a fine mixed subdivision of the Minkowski sum $Q = Q_1 + Q_2 + \cdots + Q_n$ satisfying the properties described below to compute the mixed volume of $\mathcal{A}$. Suppose that the *subdivision* of Q consists of n-dimensional polytopes $R_1, R_2, \ldots, R_s$, which satisfy

(i) $\dim(R_j) = n$ for every $j \in \{1, 2, \ldots, s\}$,
(ii) $\bigcup_{j=1}^{s} R_j = Q$,
(iii) $R_{j_1} \cap R_{j_2}$ is a face of both R_{j_1} and R_{j_2} for $j_1 \neq j_2$.

In addition, in order for the collection of $R_1, R_2, \ldots, R_s$ to be a *fine mixed subdivision* of Q, each piece polytope R_j needs to be represented as the Minkowski sum $\mathrm{conv}(C_1^j) + \mathrm{conv}(C_2^j) + \cdots + \mathrm{conv}(C_n^j)$ for some subset $\boldsymbol{C} = (C_1^j, C_2^j, \ldots, C_n^j)$ of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$, such that

(iv) for each $j \in \{1, 2, \ldots, s\}$, $\mathrm{conv}(C_i^j)$ $(i \in N)$ is a simplex of dimension $\#C_i^j - 1$ and $\sum_{i=1}^{n} \dim(\mathrm{conv}(C_i^j)) = n$.

Let $R_j = \mathrm{conv}(C_1^j) + \mathrm{conv}(C_2^j) + \cdots + \mathrm{conv}(C_n^j)$ be a polytope in a fine mixed subdivision of Q. Then we call it a *mixed cell* if $\dim(\mathrm{conv}(C_1^j)) = \dim(\mathrm{conv}(C_2^j)) = \cdots = \dim(\mathrm{conv}(C_n^j)) = 1$. For a fully mixed polynomial system, the set of all mixed cells in a fine mixed subdivision of Q provides the mixed volume of $\mathcal{A}$ as the summation of the volume of these cells. See [9], [16] and [17] for a detail description of the computation of the mixed volume via mixed cells.

Furthermore, [14] presents how to construct a fine mixed subdivision of Q. We choose a real-valued function $\omega_i \colon \mathcal{A}_i \to \mathbb{R}$, and call $\omega = (\omega_1, \omega_2, \ldots, \omega_n)$ a *lifting* on $\mathcal{A}$. A lifting ω lifts $\mathcal{A}_i$ to

$$\hat{\mathcal{A}}_i = \left\{ \begin{pmatrix} \boldsymbol{a} \\ \omega_i(\boldsymbol{a}) \end{pmatrix} \colon \boldsymbol{a} \in \mathcal{A}_i \right\}.$$

We use the notation $\hat{\mathcal{A}} = (\hat{\mathcal{A}}_1, \hat{\mathcal{A}}_2, \ldots, \hat{\mathcal{A}}_n)$ and $\hat{Q}_i = \mathrm{conv}(\hat{\mathcal{A}}_i)$, $\hat{Q} = \hat{Q}_1 + \cdots + \hat{Q}_n$. We consider the lifted polytope $\hat{Q} \subseteq \mathbb{R}^{n+1}$. A face of $\hat{Q}$ is said to be a *lower facet* if its inner normal has a positive last coordinate. For every $i \in N$ and every $\boldsymbol{a} \in \mathcal{A}_i$, choose a random number for $\omega_i(\boldsymbol{a})$ so that $\omega_i(\boldsymbol{a})$ is generic. Then the projection $\mathbb{R}^{n+1} \to \mathbb{R}^n$ of the set of lower facets of $\hat{Q}$ forms the fine mixed subdivision of Q.

Li and Li [17] developed an efficient algorithm for the enumeration of all lower facets, which correspond to mixed cells of the support set $\mathcal{A}$, of the lifted polytope $\hat{Q}$ via LP problems. All mixed cells of $\mathcal{A}$ are in one-to-one correspondence to the set $\boldsymbol{C} = (\{\boldsymbol{a}_1, \boldsymbol{a}_1'\}, \{\boldsymbol{a}_2, \boldsymbol{a}_2'\}, \ldots, \{\boldsymbol{a}_n, \boldsymbol{a}_n'\}) \in \prod_{i=1}^n \mathcal{A}_i$ such that the linear inequality system

$$\begin{aligned} &\langle \hat{\boldsymbol{a}}_i, \hat{\boldsymbol{\alpha}} \rangle = \langle \hat{\boldsymbol{a}}_i', \hat{\boldsymbol{\alpha}} \rangle, \\ &\langle \hat{\boldsymbol{a}}_i, \hat{\boldsymbol{\alpha}} \rangle \le \langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle, \qquad \forall \hat{\boldsymbol{a}} \in \hat{\mathcal{A}}_i \backslash \{\hat{\boldsymbol{a}}_i, \hat{\boldsymbol{a}}_i'\} \quad (i \in N), \end{aligned} \tag{1}$$

where

$$\hat{\boldsymbol{a}} = \begin{pmatrix} \boldsymbol{a} \\ \omega_i(\boldsymbol{a}) \end{pmatrix} \in \hat{\mathcal{A}}_i \quad \text{and} \quad \hat{\boldsymbol{\alpha}} = \begin{pmatrix} \boldsymbol{\alpha} \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1},$$

is feasible. The feasibility check of this system can be represented using an LP formulation.

2.2. *Enumeration Tree*

We now describe mixed cells in terms of systems of linear inequalities based on the Li–Li algorithm [17], and the basic idea of our dynamic enumeration method. For every $L \subseteq N$, define

$$\begin{aligned} \Omega(L) &= \left\{ \boldsymbol{C} = (C_1, C_2, \ldots, C_n) \colon \begin{array}{l} C_i \subseteq \mathcal{A}_i,\ \#C_i = 2\ (i \in L) \\ C_j = \emptyset\ (j \notin L) \end{array} \right\}, \\ \Omega &= \bigcup_{L \subseteq N} \Omega(L). \end{aligned}$$

The set Ω serves as candidates of the nodes of enumeration trees. Specifically, $\emptyset^n \in \Omega(\emptyset) = \{\emptyset^n\}$ is the root node, and $\Omega(N) \subset \Omega$ the leaf nodes. In general, the ℓth level nodes of an enumeration tree are chosen from $\bigcup_{L \subseteq N, \#L = \ell} \Omega(L)$. For every $\boldsymbol{C} \in \Omega$, let $L(\boldsymbol{C}) = \{i \in N \colon C_i \neq \emptyset\}$. For every $\boldsymbol{C} \in \Omega$ and $L \subseteq N$, we denote the vector consisting of C_i $(i \in L)$ by $\boldsymbol{C}_L = (C_i \colon i \in L)$.

For every $i \in N$ and every $\boldsymbol{a} \in \mathcal{A}_i$, let $\omega_i(\boldsymbol{a})$ denote a random number chosen from some bounded interval of $\mathbb{R}$. For every $\boldsymbol{C} \in \Omega$ with $C_i = \{\boldsymbol{a}^{p_i}, \boldsymbol{a}^{q_i}\}$ $(i \in L(\boldsymbol{C}))$, we consider a linear inequality system in a variable vector $\boldsymbol{\alpha} \in \mathbb{R}^n$:

$$\mathcal{I}(\boldsymbol{C}): \quad \begin{cases} \langle \boldsymbol{a}^{p_i} - \boldsymbol{a}^{q_i}, \boldsymbol{\alpha} \rangle = \omega_i(\boldsymbol{a}^{q_i}) - \omega_i(\boldsymbol{a}^{p_i}), \\ \langle \boldsymbol{a}^{p_i} - \boldsymbol{a}, \boldsymbol{\alpha} \rangle \le \omega_i(\boldsymbol{a}) - \omega_i(\boldsymbol{a}^{p_i}), \qquad (\boldsymbol{a} \in \mathcal{A}_i \backslash \{\boldsymbol{a}^{p_i}, \boldsymbol{a}^{q_i}\},\ i \in L(\boldsymbol{C})). \end{cases}$$

We say that $\boldsymbol{C} \in \Omega$ is feasible when $\mathcal{I}(\boldsymbol{C})$ is feasible. Let

$$\Omega_*(L) = \{\boldsymbol{C} \in \Omega(L) \colon \boldsymbol{C} \text{ is feasible}\}.$$

Then we know from (1) that $\Omega_*(N)$ forms the set of all mixed cells. Note that a leaf node $\boldsymbol{C} \in \Omega(N)$ is a mixed cell if and only if $\boldsymbol{C}$ is feasible, so properties (i) and (ii), which are described in the previous section, are satisfied. Thus, for the enumeration of all mixed cells, we need to find all elements in $\Omega_*(N)$.

For every $\boldsymbol{C} \in \Omega$ with a proper subset $L(\boldsymbol{C})$ of N and every $t \in N \backslash L(\boldsymbol{C})$, define a set of child nodes of $\boldsymbol{C}$ by

$$W(\boldsymbol{C}, t) = \{\bar{\boldsymbol{C}} \in \Omega(L(\boldsymbol{C}) \cup \{t\}) \colon \bar{\boldsymbol{C}}_{L(C)} = \boldsymbol{C}_{L(C)}\}.$$

We can build an enumeration tree if we successively choose $t \in N \backslash L(\boldsymbol{C})$ at each node $\boldsymbol{C}$ of the tree starting from the root node $\boldsymbol{C} = \emptyset^n$ with $L(\boldsymbol{C}) = \emptyset$ (see Algorithm 2.2 for more details). In the *static enumeration method* employed in [8], [9], [17] and [21], we first choose a permutation of N or a one-to-one mapping $\pi \colon N \to N$, and restrict nodes of the enumeration trees to $\boldsymbol{C} \in \Omega(\{\pi(1), \pi(2), \ldots, \pi(\ell)\})$ with some $\ell \in N$. Note that we have the set $\Omega(N)$ of leaf nodes if we take $\ell = n$, and the root node is a feasible node $\emptyset^n \in \Omega(\emptyset)$. First, the static enumeration method constructs the set $W(\emptyset^n, \pi(1))$ as child nodes of the root node $\emptyset^n \in \Omega(\emptyset)$. Next, if a node $\boldsymbol{C} \in \Omega(\{\pi(1), \pi(2), \ldots, \pi(\ell)\})$ for some $1 \leq \ell < n$ has been found to be feasible, the static enumeration generates $W(\boldsymbol{C}, \pi(\ell+1))$ as the set of child nodes of $\boldsymbol{C}$. Thus the structure of the static enumeration tree is completely determined by a permutation $\pi \colon N \to N$.

In the enumeration method that we propose in this paper, an enumeration tree is constructed dynamically as the enumeration of nodes proceeds. To explain a procedure for the construction of such a tree, we define some notation. Let $T = (V, E)$ be a rooted tree such that the vertex set V and edge set E are written as $V = \bigcup_{\ell=0}^{n} V_\ell$ and $E = \bigcup_{\ell=0}^{n} E_\ell$. V_0 corresponds to the root node $\emptyset^n$, and we define E_0 as an empty set for consistency with discussions below. The procedure for the construction of a tree T with allocating nodes dynamically is written as follows:

Algorithm 2.1. (Construction of a Tree T)

Input: A support $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$.
Output: A tree $T = (V = \bigcup_{\ell=0}^{n} V_\ell,\ E = \bigcup_{\ell=0}^{n} E_\ell)$.

$V_\ell \leftarrow \emptyset^n$ $(\ell = 0, \ldots, n)$, $E_\ell \leftarrow \emptyset$ $(\ell = 0, \ldots, n)$ and $\ell \leftarrow 0$.
while $\ell < n$ **do**
  for all $\boldsymbol{C} \in V_\ell$ **do**
    Choose t from $N \backslash L(\boldsymbol{C})$.
    $V_{\ell+1} \leftarrow V_{\ell+1} \cup W(\boldsymbol{C}, t)$ and $E_{\ell+1} \leftarrow E_{\ell+1} \cup \{(\boldsymbol{C}, \bar{\boldsymbol{C}}) \in V_\ell \times V_{\ell+1} \colon \bar{\boldsymbol{C}} \in W(\boldsymbol{C}, t)\}$.
  end for
  $\ell \leftarrow \ell + 1$.
end while

If two nodes $\boldsymbol{C} \in V_\ell$ and $\bar{\boldsymbol{C}} \in V_{\ell+1}$ of a tree are joined with an edge, we say that $\bar{\boldsymbol{C}}$ is a *child node* of the *parent node* $\boldsymbol{C}$. Any node on all paths from $\boldsymbol{C}$ to reachable leaf nodes, which are elements in V_n, is a *descendant node* of $\boldsymbol{C}$. Each feasible leaf node corresponds to mixed cells.

Given an input data $\mathcal{A}$, Algorithm 2.1 produces various types of trees T depending on the choice of an index t from $N \backslash L(\boldsymbol{C})$. For instance, a static enumeration tree proposed in the existing algorithms [8], [9], [17], [21] is constructed if we fix a permutation π of N in advance and if we choose $\pi(\ell + 1)$ for any $C \in V_\ell$ as the index t in Algorithm 2.1.

As in the static enumeration, $\emptyset^n \in \Omega(\emptyset)$ serves as the root node and $\Omega(N)$ as the set of leaf nodes. Suppose that a node $\boldsymbol{C}$ with some proper subset $L(\boldsymbol{C})$ of N has been found to

be feasible. Then we try to choose a $t \in N \backslash L(\boldsymbol{C})$ so that only a small portion of its child nodes $W(\boldsymbol{C}, t)$ are expected to be feasible. The important issue here is how inexpensively we estimate the number of feasible child nodes in $W(\boldsymbol{C}, s)$ for all $s \in N \backslash L(\boldsymbol{C})$. For this purpose, we propose a simple technique of feasibility check which applies a criterion of unboundedness detection in the simplex method. We also utilize the *relation table* given in [9] to find some infeasible child nodes in $W(\boldsymbol{C}, s)$.

2.3. *The Dynamic Enumeration Algorithm*

By deleting worthless nodes which do not contain any mixed cell, we can efficiently enumerate all feasible leaf nodes of a tree. Indeed, if a node is infeasible, all of its descendant nodes are infeasible, and thus we need not check the feasibility of the descendant nodes. Furthermore, we employ a depth-first order when applying the feasibility check to all nodes of an enumeration tree to save memory requirement during the execution of enumeration. Taking account of these factors, a depth-first search algorithm for the enumeration of all mixed cells is constructed.

It is convenient to introduce the word "list" which will be used in this algorithm. For a finite set A, we denote list(A) as an ordered sequence of the elements in A, where the actual order is not relevant in our succeeding discussions but it is fixed. For a pair of list(A) and list(B), where A and B are finite sets, list(A)+list(B) stands for the list which is generated by connecting list(B) with list(A) by "stacking" list(B) on list(A); for example, if list(A) $= (a, b, c)$ and list(B) $= (d, e)$, then list(A) + list(B) $= (a, b, c, d, e)$.

The dynamic enumeration algorithm is outlined below. V_a is a list of nodes, initialized as an empty set, and ν_* denotes the total number of nodes generated by the algorithm. The total amount of work to generate all mixed cells by the algorithm is measured by ν_*.

Algorithm 2.2. (The Dynamic Enumeration Algorithm)

Input: A lifted support $\hat{\mathcal{A}} = (\hat{\mathcal{A}}_1, \hat{\mathcal{A}}_2, \ldots, \hat{\mathcal{A}}_n)$.
Output: All mixed cells $\boldsymbol{C} \in \Omega_*(N)$ and the total number ν_* of nodes generated.

$V_a \leftarrow$ list($\Omega(\emptyset)$) and $\nu \leftarrow 1$.
while $\#V_a \neq 0$ **do**
Take out the last element $\boldsymbol{C}$ of V_a and remove $\boldsymbol{C}$ from V_a.
(A): Check whether $\boldsymbol{C}$ is feasible or infeasible.
if $\boldsymbol{C}$ is feasible **then**
if $L(\boldsymbol{C}) = N$ **then**
output $\boldsymbol{C}$ as a mixed cell.
else
(B): Choose a t from $N \backslash L(\boldsymbol{C})$ and $\nu \leftarrow \nu + \#W(\boldsymbol{C}, t)$.
(C): $V_a \leftarrow V_a +$ list($W(\boldsymbol{C}, t)$).
end if
end if
end while
$\nu_* \leftarrow \nu$.
return ν_* as the total number of nodes of the constructed tree.

Recall that for each node $\boldsymbol{C}$, the system of linear inequalities $\mathcal{I}(\boldsymbol{C})$ is solved to see whether $\boldsymbol{C}$ is feasible or infeasible. Since the efficiency of the algorithm depends on the size of V_a, we utilize the one point test [8], [9], [17], [21] to decrease the number of elements to be added to V_a. Suppose that $\boldsymbol{C} \in V_a$ and $t \in N\backslash L(\boldsymbol{C})$. For every $\boldsymbol{a} \in \mathcal{A}_t$, the one point test checks the feasibility of the system of linear inequalities in $\boldsymbol{\alpha} \in \mathbb{R}^n$,

$$\mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a})\colon \begin{cases} \mathcal{I}(\boldsymbol{C}), \\ \langle \boldsymbol{a} - \boldsymbol{b}, \boldsymbol{\alpha} \rangle \leq \omega_t(\boldsymbol{b}) - \omega_t(\boldsymbol{a}) \qquad (\boldsymbol{b} \in \mathcal{A}_t\backslash\{\boldsymbol{a}\}). \end{cases} \tag{2}$$

If $\mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a})$ is infeasible, we can remove $\bar{\boldsymbol{C}}$ with $\bar{C}_t = \{\boldsymbol{a}, \boldsymbol{a}'\}$ for any $\boldsymbol{a}' \in \mathcal{A}_t\backslash\{\boldsymbol{a}\}$ from $W(\boldsymbol{C}, t)$ because $\mathcal{I}(\bar{\boldsymbol{C}})$ is also infeasible. Therefore, as feasible node candidates we only consider

$$W_1(\boldsymbol{C}, t) \;=\; \{\bar{\boldsymbol{C}} \in \Omega(L(\boldsymbol{C}) \cup \{t\})\colon\ \bar{\boldsymbol{C}}_L = \boldsymbol{C}_L \text{ and } \bar{C}_t \subseteq \mathcal{A}_t(\boldsymbol{C})\},$$

where $\mathcal{A}_t(\boldsymbol{C}) = \{\boldsymbol{a} \in \mathcal{A}_t\colon\ \mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a})$ is feasible$\}$. After m_t $(= \#\mathcal{A}_t)$ feasibility checks of linear inequality systems for constructing $\mathcal{A}_t(\boldsymbol{C})$, we have $W_1(\boldsymbol{C}, t)$ satisfying

$$\{\bar{\boldsymbol{C}} \in W(\boldsymbol{C}, t)\colon\ \bar{\boldsymbol{C}} \text{ is feasible}\} \subseteq W_1(\boldsymbol{C}, t) \subseteq W(\boldsymbol{C}, t).$$

Thus we can replace (C) by

$$V_a \leftarrow V_a + \text{list}(W_1(\boldsymbol{C}, t)).$$

This one point test is known to be very effective to increase the computational efficiency of enumeration [8], [9], [17], [21].

Ideally we would like to choose a $t \in N\backslash L(\boldsymbol{C})$ at (B) so that the size of $W_1(\boldsymbol{C}, t)$ is the smallest among the sizes of $W_1(\boldsymbol{C}, s)$ $(s \in N\backslash L(\boldsymbol{C}))$. Finding such a $t \in N\backslash L(\boldsymbol{C})$ exactly, however, is expensive because all $W_1(\boldsymbol{C}, s)$ $(s \in N\backslash L(\boldsymbol{C}))$ need to be constructed. Therefore, we propose to replace $W_1(\boldsymbol{C}, s)$ by another set which can be obtained easily. Utilizing a feasible solution $\boldsymbol{x}_{\text{init}}$ of $\mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a})$ which is generated from a solution of $\mathcal{I}(\boldsymbol{C})$, our method computes $\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{\text{init}})$ $(s \in N\backslash L(\boldsymbol{C}))$ satisfying

$$W_1(\boldsymbol{C}, s) \subseteq \hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{\text{init}}) \subseteq W(\boldsymbol{C}, s) \qquad (s \in N\backslash L(\boldsymbol{C})),$$

and chooses a $t \in N\backslash L(\boldsymbol{C})$ such that the size of $\hat{W}_1(\boldsymbol{C}, t, \boldsymbol{x}_{\text{init}})$ attains the minimum among the sizes of $\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{\text{init}})$ $(s \in N\backslash L(\boldsymbol{C}))$. We call this method the *dynamic enumeration method*. In Section 3.2, we explain how to generate the set $\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{\text{init}})$ $(s \in N\backslash L(\boldsymbol{C}))$. Also the relation table proposed in [9] can be used to find some infeasible child nodes in $W(\boldsymbol{C}, s)$. In our numerical experiments, the relation table is applied to remove infeasible child nodes from $W(\boldsymbol{C}, s)$ before $\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{\text{init}})$ is constructed.

3. Technical Details of the Algorithm

3.1. *Formulation for Checking the Feasibility of a System of Linear Inequalities*

The feasibility check of $\boldsymbol{C} \in \Omega$, conducted at (A) of Algorithm 2.2, can be formulated via an LP problem. Namely, we test the feasibility of the following problem in the vector

$\boldsymbol{\alpha} \in \mathbb{R}^n$ of decision variables:

$$\mathrm{P}(\boldsymbol{C}): \quad \max.\langle \boldsymbol{\gamma}, \boldsymbol{\alpha} \rangle \text{ s.t. } \mathrm{I}(\boldsymbol{C}),$$

where $\boldsymbol{\gamma} \in \mathbb{R}^n$ is some fixed vector. For every $\boldsymbol{C} \in \Omega$ with $C_i = \{\boldsymbol{a}^{p_i}, \boldsymbol{a}^{q_i}\}$ $(i \in L(\boldsymbol{C}))$, the dual problem is written as

$$\begin{aligned} \mathrm{D}(\boldsymbol{C}): \quad & \min.\ \Phi(\boldsymbol{x}; \boldsymbol{C}) \\ & \text{s.t.} \quad \Psi(\boldsymbol{x}; \boldsymbol{C}) = \boldsymbol{\gamma}, \\ & \qquad x_{\boldsymbol{a}} \geq 0 \qquad (\boldsymbol{a} \in \mathcal{A}_i \backslash \{\boldsymbol{a}^{p_i}, \boldsymbol{a}^{q_i}\}), \\ & \qquad -\infty < x_{\boldsymbol{a}^{q_i}} < +\infty \qquad (i \in L(\boldsymbol{C})). \end{aligned}$$

Here, a vector of decision variables is given by the column vector

$$\boldsymbol{x} = (x_{\boldsymbol{a}}\colon \boldsymbol{a} \in \mathcal{A}_i \backslash \{\boldsymbol{a}^{p_i}\},\ i \in L(\boldsymbol{C})) \in \mathbb{R}^{d_D}, \qquad \text{where} \quad d_D := \sum_{i \in L(C)} (m_i - 1), \tag{3}$$

and the symbols $\Phi(\boldsymbol{x}; \boldsymbol{C})$ and $\Psi(\boldsymbol{x}; \boldsymbol{C})$ are linear functions in $\boldsymbol{x}$ such that

$$\begin{aligned} \Phi(\boldsymbol{x}; \boldsymbol{C}) &= \sum_{i \in L(C)} \sum_{\boldsymbol{a} \in \mathcal{A}_i \backslash \{\boldsymbol{a}^{p_i}\}} (\omega_i(\boldsymbol{a}) - \omega_i(\boldsymbol{a}^{p_i}))\, x_{\boldsymbol{a}} \quad \text{and} \\ \Psi(\boldsymbol{x}; \boldsymbol{C}) &= \sum_{i \in L(C)} \sum_{\boldsymbol{a} \in \mathcal{A}_i \backslash \{\boldsymbol{a}^{p_i}\}} (\boldsymbol{a}^{p_i} - \boldsymbol{a})\, x_{\boldsymbol{a}}. \end{aligned}$$

Any real vector $\boldsymbol{\gamma}$ in $\mathrm{P}(\boldsymbol{C})$ can be taken for the cost vector. Accordingly we set $\boldsymbol{\gamma}$ so that $\mathrm{D}(\boldsymbol{C})$ becomes feasible. Since this primal–dual pair satisfies the duality theorem, $\mathrm{P}(\boldsymbol{C})$ is feasible if and only if $\mathrm{D}(\boldsymbol{C})$ is bounded below, and $\mathrm{P}(\boldsymbol{C})$ is infeasible if and only if $\mathrm{D}(\boldsymbol{C})$ is unbounded. Therefore, to determine the feasibility of $\boldsymbol{C}$, we need to see whether $\mathrm{D}(\boldsymbol{C})$ is bounded or not.

Now we consider a formulation of an LP as

$$\begin{aligned} & \min.\ \langle \boldsymbol{c}, \boldsymbol{x} \rangle \\ & \text{s.t.} \quad \boldsymbol{G}\boldsymbol{x} = \boldsymbol{h}, \\ & \qquad x_i \geq 0 \qquad (i \in I), \\ & \qquad -\infty < x_j < +\infty \qquad (j \in J), \end{aligned} \tag{4}$$

where a coefficient matrix $\boldsymbol{G} \in \mathbb{R}^{k \times d}$, cost vector $\boldsymbol{c} \in \mathbb{R}^d$ and constant vector $\boldsymbol{h} \in \mathbb{R}^k$ are given, and $\boldsymbol{x} \in \mathbb{R}^d$ is a vector of decision variables. These index sets I and J of decision variables satisfy $I \cap J = \emptyset$ and $I \cup J = \{1, 2, \ldots, d\}$. Here, x_i $(i \in I)$ and x_j $(j \in J)$ are called the *nonnegative variable* and a *free variable*, respectively. The primal–dual pair $\mathrm{P}(\boldsymbol{C})$ and $\mathrm{D}(\boldsymbol{C})$ can be transformed into (4) by introducing slack variables to the inequalities of $\mathrm{P}(\boldsymbol{C})$ and replacing the cost vector $\boldsymbol{\gamma}$ of $\mathrm{P}(\boldsymbol{C})$ by $-\boldsymbol{\gamma}$. In consequence of these transformations, $\mathrm{P}(\boldsymbol{C})$ has d_P variables and k_P equalities such that

$$d_P = n + \sum_{i \in L(C)} (m_i - 2) \quad \text{and} \quad k_P = \sum_{i \in L(C)} (m_i - 1).$$

On the other hand, $\mathrm{D}(\boldsymbol{C})$ has d_D variables in (3) and k_D equalities such that $k_D = n$. Here d_D is not greater than d_P for any $L(\boldsymbol{C}) \subseteq N$. Also k_D is constant whereas k_P is

monotonically increasing with respect to the cardinality of $L(\boldsymbol{C})$. From the definition of a mixed cell of the support set, $\mathcal{A}_i$ $(i \in N)$ has at least two elements, i.e., $m_i \geq 2$. Therefore there exists $L' \subseteq N$ such that $k_P \geq k_D$. Consequently for any L such that $L' \subseteq L \subseteq N$, the number of constraints and that of variables in $\mathrm{D}(\boldsymbol{C})$ are not greater than those of $\mathrm{P}(\boldsymbol{C})$. Therefore, it is reasonable to observe whether $\mathrm{D}(\boldsymbol{C})$ is bounded for checking the feasibility of $\boldsymbol{C}$.

We formulate the one point test, stated in the previous section, via an LP problem. For every $\boldsymbol{C} \in \Omega,\ t \in N \backslash L(\boldsymbol{C})$ and $\boldsymbol{a} \in \mathcal{A}_t$, checking the feasibility of (2) can be written as the following LP problem in the vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ of decision variables:

$$\mathrm{P}_1(\boldsymbol{C}, t, \boldsymbol{a}): \quad \max.\langle \boldsymbol{\gamma}, \boldsymbol{\alpha} \rangle \text{ s.t. } \mathrm{I}(\boldsymbol{C}, t, \boldsymbol{a}),$$

where $\boldsymbol{\gamma} \in \mathbb{R}^n$ is some fixed vector. As the one point test, we check the feasibility of this problem for all $\boldsymbol{a} \in \mathcal{A}_t$. The dual problem of $\mathrm{P}_1(\boldsymbol{C}, t, \boldsymbol{a})$ is given by

$$\begin{aligned}
\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a}): \quad \min.\ & \Phi(\boldsymbol{x}; \boldsymbol{C}) + \sum_{\boldsymbol{b} \in \mathcal{A}_t \backslash \{\boldsymbol{a}\}} (\omega_t(\boldsymbol{b}) - \omega_t(\boldsymbol{a})) y_{\boldsymbol{b}} \\
\text{s.t.}\ & \Psi(\boldsymbol{x}; \boldsymbol{C}) + \sum_{\boldsymbol{b} \in \mathcal{A}_t \backslash \{\boldsymbol{a}\}} (\boldsymbol{a} - \boldsymbol{b}) y_{\boldsymbol{b}} = \boldsymbol{\gamma}, \\
& x_{\boldsymbol{a}} \geq 0 \qquad (\boldsymbol{a} \in \mathcal{A}_i \backslash \{\boldsymbol{a}^{p_i}, \boldsymbol{a}^{q_i}\}) \quad \text{and} \\
& \qquad\qquad -\infty < x_{\boldsymbol{a}^{q_i}} < +\infty, \qquad \text{for} \quad i \in L(\boldsymbol{C}), \\
& y_{\boldsymbol{b}} \geq 0 \qquad (\boldsymbol{b} \in \mathcal{A}_t \backslash \{\boldsymbol{a}\}).
\end{aligned}$$

Using $\boldsymbol{x} \in \mathbb{R}^{d_D}$ of (3) and the column vector $\boldsymbol{y} = (y_{\boldsymbol{b}}\colon\ \boldsymbol{b} \in \mathcal{A}_t \backslash \{\boldsymbol{a}^{p_t}\}) \in \mathbb{R}^{(m_t - 1)}$, the vector of decision variables in this problem is represented as

$$\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \in \mathbb{R}^{\bar{d}_D}, \qquad \text{where} \quad \bar{d}_D := \sum_{i \in L(C) \cup \{t\}} (m_i - 1).$$

Since we can choose $\boldsymbol{\gamma}$ such that $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ is feasible, this primal–dual pair satisfies the duality theorem. Similar to $\mathrm{P}(\boldsymbol{C})$ and $\mathrm{D}(\boldsymbol{C})$, we can say that the size of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ is not larger than that of $\mathrm{P}_1(\boldsymbol{C}, t, \boldsymbol{a})$ for any $\boldsymbol{C} \in \Omega,\ t \in N \backslash L(\boldsymbol{C})$ and $\boldsymbol{a} \in \mathcal{A}_t$. Therefore, we deal with $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ as the one point test, and check whether this problem is bounded or not. From the results of the one point test, we can generate the set $W_1(\boldsymbol{C}, t)$ which satisfies $W_1(\boldsymbol{C}, t) \subseteq W(\boldsymbol{C}, t)$.

We refer to how to fix a right-hand constant vector $\boldsymbol{\gamma}$ on $\mathrm{D}(\boldsymbol{C})$ and $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$. For $\boldsymbol{C} \in \Omega(L)$ with $C_i = \{\boldsymbol{a}^{p_i}, \boldsymbol{a}^{q_i}\}$ $(i \in L(\boldsymbol{C}))$ and a proper subset $L(\boldsymbol{C})$ of N, we consider the problem $\mathrm{D}(\boldsymbol{C})$. Using the arbitrary nonnegative vector $\hat{\boldsymbol{x}} \in \mathbb{R}^{d_D}$, we compute

$$\hat{\boldsymbol{\gamma}} = \Psi(\hat{\boldsymbol{x}}; \boldsymbol{C})$$

and set this $\hat{\boldsymbol{\gamma}}$ as a right-hand vector $\boldsymbol{\gamma}$ of the problem $\mathrm{D}(\boldsymbol{C})$ and $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ for some $\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \backslash L(\boldsymbol{C})$. As a result, $\mathrm{D}(\boldsymbol{C})$ is feasible. We suppose that the problem $\mathrm{D}(\boldsymbol{C})$ is bounded, and denote an optimal solution of this problem by $\boldsymbol{x}_* \in \mathbb{R}^{d_D}$. Then the vector

$$\boldsymbol{x}_{\mathrm{init}} = \begin{pmatrix} \boldsymbol{x}_* \\ \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{\bar{d}_D} \tag{5}$$

is feasible solution in $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N\backslash L(\boldsymbol{C})$) because $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ with fixed $\boldsymbol{y} = (y_{\boldsymbol{b}}\colon\ \boldsymbol{b} \in \mathcal{A}_t\backslash\{\boldsymbol{a}^{p_t}\}) = \boldsymbol{0}$ is equivalent to $\mathrm{D}(\boldsymbol{C})$. Also, we consider the problem $\mathrm{D}(\bar{\boldsymbol{C}})$ ($\bar{\boldsymbol{C}} \in \Omega(L(\boldsymbol{C}) \cup \{t\})$) with $\bar{\boldsymbol{C}}_L = \boldsymbol{C}_L$, and use $\hat{\boldsymbol{\gamma}}$ as a right-hand vector $\boldsymbol{\gamma}$ of this problem. We easily see that this problem is feasible. Furthermore, an optimal solution of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ is a feasible solution of $\mathrm{D}(\bar{\boldsymbol{C}})$. Since the simplex method is suitable for solving many LP problems with a similar structure, we employ the method to solve problems arising from checking the feasibility of linear inequality systems.

3.2. *How to Choose an Index t from $N\backslash L(\boldsymbol{C})$*

The choice of a $t \in N\backslash L(\boldsymbol{C})$ at (B) of Algorithm 2.2 has a major effect on the computational efficiency of this algorithm. As stated in Section 2.3, we want to choose a $t \in N\backslash L(\boldsymbol{C})$ such that the size of $W_1(\boldsymbol{C}, s)$ is the smallest among $s \in N\backslash L(\boldsymbol{C})$. However, this task is expensive in general because we need to check the feasibility of $\mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a})$ for every $\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N\backslash L(\boldsymbol{C})$ at (B) additionally. We employ a less expensive technique to choose a $t \in N\backslash L(\boldsymbol{C})$ so that an evaluation measure ν_* of the efficiency of our dynamic enumeration method becomes smaller.

To check the feasibility of $\boldsymbol{C} \in \Omega(L)$ with a proper subset L of N, we have solved the dual problem $\mathrm{D}(\boldsymbol{C})$, and in (B) we have an optimal solution $\boldsymbol{x}_* \in \mathbb{R}^{d_D}$ of $\mathrm{D}(\boldsymbol{C})$. As stated in the previous subsection, if we set $\boldsymbol{x}_{\mathrm{init}} \in \mathbb{R}^{\bar{d}_D}$ by (5) using $\boldsymbol{x}_*$, the vector $\boldsymbol{x}_{\mathrm{init}}$ is a feasible solution of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ for any $\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N\backslash L(\boldsymbol{C})$. Because the structures of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ and $\mathrm{D}(\boldsymbol{C})$ are similar to each other, we usually require only a few iterations to solve $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ by the simplex method when using $\boldsymbol{x}_{\mathrm{init}}$ as an initial feasible solution. Thus we expect that $\boldsymbol{x}_{\mathrm{init}}$ is incident to an unbounded direction if $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ is unbounded. Accordingly, instead of applying the simplex method to check the feasibility of $\mathrm{P}_1(\boldsymbol{C}, t, \boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N\backslash L(\boldsymbol{C})$) , we propose to test whether the feasible solution $\boldsymbol{x}_{\mathrm{init}}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ has unbounded directions or not.

At (B) we consider

$$\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{\mathrm{init}}) = \{\bar{\boldsymbol{C}} \in \Omega(L(\boldsymbol{C}) \cup \{s\})\colon\ \bar{\boldsymbol{C}}_L = \boldsymbol{C}_L \text{ and } \bar{C}_s \subseteq \hat{\mathcal{A}}_s(\boldsymbol{C}, \boldsymbol{x}_{\mathrm{init}})\},$$

where

$$\hat{\mathcal{A}}_t(\boldsymbol{C}, \boldsymbol{x}_{\mathrm{init}}) = \{\boldsymbol{a} \in \mathcal{A}_t\colon\ \boldsymbol{x}_{\mathrm{init}} \text{ of } \mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a}) \text{ has no unbounded direction}\}$$

and we choose an index $\hat{t} \in N\backslash L(\boldsymbol{C})$ which attains

$$\min_{s \in N\backslash L(C)} \#\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{\mathrm{init}}).$$

In general, this index $\hat{t}$ does not coincide with the index t which achieves the minimum number of elements in $W_1(\boldsymbol{C}, s)$ ($s \in N\backslash L(\boldsymbol{C})$). We will observe from the numerical results, however, that the evaluation measure ν_* is much smaller for our dynamic enumeration method than for the static enumeration method, and the total computational time for finding all mixed cells is reduced dramatically.

We next explain how to compute elements in $\hat{W}_1(\boldsymbol{C}, t, \boldsymbol{x}_{\mathrm{init}})$ ($t \in N\backslash L(\boldsymbol{C})$) more precisely, using an LP form of (4) instead of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ for simplicity of notation. For

(4), we assume that the number of variables d is not less than that of constraints k and the matrix $\boldsymbol{G} = (\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_d)$ has full row rank. If this problem (4) is feasible, there exists a vertex $\boldsymbol{x} \in \mathbb{R}^d$ on the feasible region which consists of two components: the vector of basic variables $\boldsymbol{x}_B = \boldsymbol{G}_B^{-1}\boldsymbol{h} \in \mathbb{R}^k$ and nonbasic variables $\boldsymbol{x}_N = \boldsymbol{0} \in \mathbb{R}^{d-k}$ where $\boldsymbol{G}_B \in \mathbb{R}^{k\times k}$ is a basic matrix. Note that $\boldsymbol{x}_B = (x_{b_1}, x_{b_2}, \ldots, x_{b_k})^T$ and $\boldsymbol{x}_N = (x_{n_1}, x_{n_2}, \ldots, x_{n_{d-k}})^T$. In particular, the set of basic variables and nonbasic variables are called a basis and nonbasis. An adjacent vertex $\tilde{\boldsymbol{x}}$ of $\boldsymbol{x}$ is represented as

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \theta \boldsymbol{d}$$

by using a nonnegative scalar θ and a direction vector $\boldsymbol{d}$. Let $D = \{1, 2, \ldots, d\}$, and we denote the set of basic indices $B = \{b_1, b_2, \ldots, b_k\} \subset D$. Note that $(d - k)$ extreme rays extend from a vertex $\boldsymbol{x}$ and one direction $\boldsymbol{d}$ is chosen by fixing an index $j \in D\backslash B$. The direction $\boldsymbol{d}$ is composed of two component vectors $\boldsymbol{d}_B$ and $\boldsymbol{d}_N$ such that

$$\boldsymbol{d}_B = -\boldsymbol{G}_B^{-1}\boldsymbol{g}_j \in \mathbb{R}^k \quad \text{and} \quad \boldsymbol{d}_N = \begin{cases} d_i = 1, i = j, \\ d_i = 0, i \in D\backslash(B \cup \{j\}), \end{cases} \in \mathbb{R}^{d-k},$$

where d_i represents a component of $\boldsymbol{d}$. When we move from $\boldsymbol{x}$ to $\tilde{\boldsymbol{x}}$, the cost change per unit θ is $\langle \boldsymbol{c}, \boldsymbol{d}\rangle$. Using a component $\boldsymbol{c}_B$ of a cost vector $\boldsymbol{c}$ which corresponds to a basis B, this amount can be written as

$$\langle \boldsymbol{c}, \boldsymbol{d}\rangle = c_j - \boldsymbol{c}_B^T \boldsymbol{G}_B^{-1}\boldsymbol{g}_j \tag{6}$$

and called a reduced cost for $j \in D$. To obtain an adjacent vertex $\tilde{\boldsymbol{x}}$ of $\boldsymbol{x}$ so that the value of a cost function decreases from $\boldsymbol{x}$, we search for a direction $\boldsymbol{d}$ with $j \in D$ such that its reduced cost is negative, and determine the step size $\theta \geq 0$ such that a new vertex $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \theta\boldsymbol{d}$ satisfies its constraints; if components d_i of a direction vector $\boldsymbol{d}$ are nonnegative for all $i \in B \cap I$ where I is the index set of nonnegative variables in (4), this problem is unbounded and we say that $\boldsymbol{x}$ has an unbounded direction. Otherwise, we compute the largest θ allowed by constraints for variables.

Now we provide the criteria for detecting that the feasible solution $\boldsymbol{x}_{\text{init}}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N\backslash L(\boldsymbol{C})$) has an unbounded direction. Notice that the optimal basic matrix $\boldsymbol{G}_B \in \mathbb{R}^{n\times n}$ of $\mathrm{D}(\boldsymbol{C})$ is equal to the basic matrix on $\boldsymbol{x}_{\text{init}}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ for any $\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N\backslash L(\boldsymbol{C})$ because the number of constraints is equal to each other, and a feasible solution $\boldsymbol{x}_{\text{init}}$ can be represented as (5) using an optimal solution $\boldsymbol{x}_*$ of $\mathrm{D}(\boldsymbol{C})$. Also, note that the vector $(\boldsymbol{G}_B^{-1})^T\boldsymbol{c}_B$ in (6) is an optimal solution of its dual problem when the duality theorem holds for this primal–dual pair. Let $\boldsymbol{\alpha}_* \in \mathbb{R}^n$ be an optimal solution of $\mathrm{P}(\boldsymbol{C})$. For some $\boldsymbol{a}^{p_t} \in \mathcal{A}_t$ and $t \in N\backslash L(\boldsymbol{C})$, reduced costs on $\boldsymbol{x}_{\text{init}}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a}^{p_t})$ with $C_i = \{\boldsymbol{a}^{p_i}, \boldsymbol{a}^{q_i}\}$ ($i \in L(\boldsymbol{C})$) are written as

$$\omega_i(\boldsymbol{b}) - \omega_i(\boldsymbol{a}^{p_i}) - \langle \boldsymbol{a}^{p_i} - \boldsymbol{b},\ \boldsymbol{\alpha}_*\rangle, \qquad \text{for} \quad i \in L(\boldsymbol{C}) \cup \{t\} \quad \text{and} \quad \boldsymbol{b} \in \mathcal{A}_i\backslash\{\boldsymbol{a}^{p_i}\}.$$

Since $\boldsymbol{\alpha}_*$ is an optimal solution of $\mathrm{P}(\boldsymbol{C})$ which has $\mathcal{I}(\boldsymbol{C})$ as a constraint, these reduced costs are nonnegative for every $\boldsymbol{b} \in \mathcal{A}_i\backslash\{\boldsymbol{a}^{p_i}\}$ and $i \in L(\boldsymbol{C})$. Consequently, if there are $\boldsymbol{b} \in \mathcal{A}_t\backslash\{\boldsymbol{a}^{p_t}\}$ such that

(i) $\omega_t(\boldsymbol{b}) - \omega_t(\boldsymbol{a}^{p_t}) - \langle \boldsymbol{a}^{p_t} - \boldsymbol{b},\ \boldsymbol{\alpha}_*\rangle < 0$,
(ii) all components of a vector $-\boldsymbol{G}_B^{-1}(\boldsymbol{a}^{p_t} - \boldsymbol{b})$, which corresponds to the nonnegative variables in the basis, are nonnegative,

then we see that the feasible solution $\boldsymbol{x}_{\text{init}}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a}^{p_t})$ for $\boldsymbol{a}^{p_t} \in \mathcal{A}_t$ has an unbounded direction. Conversely, if the feasible solution $\boldsymbol{x}_{\text{init}}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a}^{p_t})$ has no such $\boldsymbol{b} \in \mathcal{A}_t \backslash \{\boldsymbol{a}^{p_t}\}$, then $\boldsymbol{a}^{p_t}$ is added to $\hat{\mathcal{A}}_t(\boldsymbol{C}, \boldsymbol{x}_{\text{init}})$.

The problem $\mathrm{D}(\boldsymbol{C})$ with $\#L(\boldsymbol{C}) = \ell$ has ℓ free variables, and the optimal basis contains all free variables if this problem is bounded. Therefore, since the basis on $\boldsymbol{x}_{\text{init}}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \backslash L(\boldsymbol{C})$) has ℓ free variables, it is enough to check $(n - \ell)$ components of a vector $-\boldsymbol{G}_B^{-1}(\boldsymbol{a}^{p_t} - \boldsymbol{b})$ in (ii).

4. Numerical Results

The proposed algorithm has been implemented and coded in C++ language. All numerical experiments were executed on a 2.4 GHz Opteron 850 with 8 GB memory, running Linux. First, we observe the total number of nodes ν_* generated by the static and dynamic enumeration, described in Section 2.3, for the cyclic-n [2] and noon-n [20] problems. In the cyclic-n problem, one polynomial has two monomials and others have n monomials such as

$$\begin{aligned}
&x_1 + x_2 + \cdots + x_{n-1} + x_n,\\
&x_1x_2 + x_2x_3 + \cdots + x_{n-1}x_n + x_nx_1,\\
&x_1x_2x_3 + x_2x_3x_4 + \cdots + x_{n-1}x_nx_1 + x_nx_1x_2,\\
&\quad\vdots\\
&x_1x_2\cdots x_n - 1.
\end{aligned}$$

In the noon-n problem, all polynomials have $(n + 1)$ monomials such as

$$\begin{aligned}
&x_1x_2^2 + x_1x_3^2 + \cdots + x_1x_n^2 - 1.1x_1 + 1,\\
&x_2x_1^2 + x_2x_3^2 + \cdots + x_2x_n^2 - 1.1x_2 + 1,\\
&\quad\vdots\\
&x_nx_1^2 + x_nx_2^2 + \cdots + x_nx_{n-1}^2 - 1.1x_n + 1.
\end{aligned}$$

For each polynomial system, we denote the support set of the ith polynomial from the top as $\mathcal{A}_i$, and for every $i \in N$ and $\boldsymbol{a} \in \mathcal{A}_i$ choose $\omega_i(\boldsymbol{a})$ randomly from some bounded interval of $\mathbb{R}$. We set $\hat{\mathcal{A}} = (\hat{\mathcal{A}}_1, \hat{\mathcal{A}}_2, \ldots, \hat{\mathcal{A}}_n)$ as the input data of Algorithm 2.2. When the static enumeration method is employed, we choose a one-to-one mapping $\pi\colon N \to N$ such that $\pi(i) = i$ $(i \in N)$.

Table 1 shows the total number of nodes ν_* generated by the static and dynamic enumeration (abbreviated by "Static enum." and "Dynamic enum.", respectively) for the cyclic-n and noon-n problem, and the next row "Ratio" indicates the ratio between ν_* given by these two methods. For these systems, this table reveals the efficiency of the dynamic enumeration method in comparison with the static one, and we can see that the ratio increases as the size of each system becomes larger.

Next, in terms of the computational time, we compare our dynamic enumeration algorithm with some existing ones. The following software packages have been developed for enumerating all mixed cells of a polynomial equation system: MVLP [7], MixedVol [9], [10], PHCpack [23], PHoM [21] and mvol [17]. In particular, [9] and [10] report

Table 1. The total number of nodes ν_* generated by the static and dynamic enumeration methods for the cyclic-n and noon-n problem

	Cyclic-n				
	$n = 10$	$n = 11$	$n = 12$	$n = 13$	$n = 14$
Static enum.	2.46×10^7	2.36×10^8	1.73×10^9	1.47×10^{10}	1.39×10^{11}
Dynamic enum.	6.60×10^6	5.19×10^7	3.27×10^8	2.69×10^9	1.91×10^{10}
Ratio	3.75	4.55	5.29	6.46	7.29
	Noon-n				
	$n = 14$	$n = 15$	$n = 16$	$n = 17$	$n = 18$
Static enum.	2.13×10^9	8.85×10^9	3.15×10^{10}	1.25×10^{11}	4.20×10^{11}
Dynamic enum.	5.77×10^7	1.88×10^8	5.13×10^8	1.40×10^9	3.50×10^9
Ratio	36.87	47.09	61.35	89.04	120.07

the superiority of MixedVol, which is coded in C++ language, in computational time over the other existing software packages. Therefore, we compare our algorithm with MixedVol for the benchmark systems: the economic-n [19], chandra-n [4] and katsura-n [3] problems in addition to the cyclic-n and noon-n problems. Notice that the katsura-n problem consists of $(n + 1)$ polynomials with $(n + 1)$ variables, and the others n polynomials with n variables. Here, we have omitted the description of the economic-n, chandra-n and katsura-n problems, which are found on the web site [22]. We summarize these computational times in Table 2. The column "Mixed volume" presents the mixed volume of the support $\mathcal{A}$ of the system, and "Mixed cells" is the total number of mixed cells generated by our algorithm. The column "Speed-up ratio" indicates the ratio between the cpu time of our algorithm and MixedVol, and "—" means that the software is not applied to the corresponding system. The last column "CPU time/cells" is the cpu time of our algorithm divided by the total number of mixed cells generated by the algorithm. For all tested systems, "CPU time/cells" increases as their sizes become larger. The increasing rate is more than linear and varies depending on each system. From the column "Speed-up ratio", we can see that our dynamic enumeration method improves the cpu time necessary for finding all mixed cells considerably, and solves large-scale polynomial systems such as the cyclic-15, noon-19,20,21, economic-20, chandra-20,21,22 and katsura-15,16 problems. This table shows the first numerical results for enumerating mixed cells of these large-scale problems.

5. Concluding Remarks

For finding all mixed cells of a polynomial system efficiently, the following two issues are essential: (1) how we construct an enumeration tree among a family of linear inequalities induced from the polynomial system, and also (2) how we check the feasibility of a linear inequality system. In this paper we proposed a depth-first search algorithm for a dynamic construction of an enumeration tree. At each iteration, the algorithm checks

Table 2. CPU time for some benchmark systems (2.4 GHz and 8 GB)

Size (n)	Mixed volume	Mixed cells	Our algorithm	MixedVol	Speed-up ratio	CPU time/cells
			Cyclic-n			($\times 10^{-3}$)
12	500,352	30,147	1m8.8s	4m43.0s	4.11	2.28
13	2,704,156	146,982	10m54.7s	49m57.4s	4.58	4.45
14	8,795,976	418,199	1h36m37.1s	7h14m24.1s	4.50	13.86
15	35,243,520	1,471,923	15h45m26.0s	—		38.53
			Noon-n			($\times 10^{-3}$)
16	43,046,689	47,773	1m4.9s	33m54.8s	31.38	1.36
17	129,140,129	94,945	3m13.1s	2h25m20.8s	45.15	2.03
18	387,420,453	179,418	7m38.3s	8h23m19.6s	65.90	2.55
19	1,162,261,429	367,993	28m1.0s	—		4.57
20	3,486,784,361	677,542	1h8m49.6s	—		6.09
21	10,460,353,161	1,542,363	3h59m44.1s	—		9.33
			Economic-n			($\times 10^{-2}$)
17	32,768	7,321	4m56.1s	20m41.8s	4.19	4.05
18	65,536	12,660	19m31.8s	1h17m56.0s	3.99	9.26
19	131,072	23,975	1h21m30.4s	4h56m4.6s	3.63	20.40
20	262,144	45,562	5h41m54.4s	—		45.03
			Chandra-n			($\times 10^{-3}$)
17	65,536	41,415	1m14.4s	33m13.4s	26.80	1.80
18	131,072	82,830	3m37.0s	2h14m15.3s	37.13	2.62
19	262,144	154,343	10m24.3s	8h19m6.3s	47.97	4.04
20	524,288	365,896	35m24.1s	—		5.81
21	1,048,576	650,515	1h30m37.5s	—		8.36
22	2,097,152	1,286,825	4h34m18.1s	—		12.79
			Katsura-n			($\times 10^{-1}$)
12	4,096	387	1m4.2s	14m3.5s	13.13	1.66
13	8,192	678	7m37.7s	1h21m19.4s	10.66	6.76
14	16,384	1,179	37m21.5s	7h54m29.4s	12.70	19.01
15	32,768	2,135	3h9m9.3s	—		53.17
16	65,536	3,366	20h43m39.4s	—		221.69

the feasibility of a linear inequality system by solving an LP problem with the effective use of information obtained at the previous iteration. Our numerical results show that the proposed algorithm considerably decreases the number of the LP problems to be solved, compared with the existing algorithms which utilize a static construction of an enumeration tree. In consequence, finding all mixed cells of large-scale polynomial systems becomes possible. Indeed, the proposed algorithm generated all mixed cells of the cyclic-15 problem for the first time in less than 16 hours.

Enumeration of all mixed cells of a polynomial system, which is the subject of this paper, plays an essential role in the polyhedral homotopy method, known as a powerful numerical method for computing all isolated zeros of a polynomial system. We expect that

the polyhedral homotopy method utilizing the proposed dynamic enumeration technique successfully solves large-scale polynomial systems which have not been processed.

Acknowledgments

The author is grateful to the referees for many helpful comments.

References

1. D. N. Bernshtein, The number of roots of a system of equations, *Funct. Anal. Appl.* **9**, 183–185 (1975).
2. G. Björk and R.Fröberg, A faster way to count the solutions of inhomogeneous systems of algebraic equations, *J. Symbolic Comput.* **12**(3), 329–336 (1991).
3. W. Boege, R. Gebauer and H. Kredel, Some examples for solving systems of algebraic equations by calculating Groebner bases, *J. Symbolic Comput.* **2**, 83–98 (1986).
4. S. Chandrasekhar, *Radiative Transfer*, Dover, New York, 1960.
5. V. Chvátal, *Linear Programming*, Freeman, San Francisco, CA, 1983.
6. Y. Dai, S. Kim and M. Kojima, Computing all nonsingular solutions of a cyclic-*n* polynomial using polyhedral homotopy continuation methods, *J. Comput. Appl. Math.* **152**, 83–97 (2003).
7. I. Z. Emiris and J. F. Canny, Efficient incremental algorithms for the sparse resultant and the mixed volume, *J. Symbolic Comput.* **20**, 117–149 (1995). Software available at http://cgi.di.uoa.gr/∼emiris/index-eng.html.
8. T. Gao and T. Y. Li, Mixed volume computation via linear programming, *Taiwanese J. Math.* **4**, 599–619 (2000).
9. T. Gao and T. Y. Li, Mixed volume computation for semi-mixed systems, *Discrete Comput. Geom.* **29**(2), 257–277 (2003).
10. T. Gao, T. Y. Li and M. Wu, MixedVol: a software package for mixed volume computation, *ACM Trans. Math. Software* **31**(4), (2005). Software available at http://www.csulb.edu/∼tgao/.
11. T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa and T. Mizutani, PHoM—a polyhedral homotopy continuation method, *Computing* **73**(1), 57–77 (2004).
12. T. Gunji, S. Kim, K. Fujisawa and M. Kojima, PHoMpara—parallel implementation of the polyhedral homotopy continuation method, *Computing* **77**(4), 387–411 (2006).
13. L. Gurvits and A. Samorodnitsky, A deterministic algorithm for approximating the mixed discriminant and mixed volume, and a combinatorial corollary, *Discrete Comput. Geom.* **27**(4), 531–550 (2002).
14. B. Huber and B. Sturmfels, A polyhedral method for solving sparse polynomial systems, *Math. Comp.* **64**, 1541–1555 (1995).
15. S. Kim and M. Kojima, Numerical stability of path tracing in polyhedral homotopy continuation methods, *Computing* **73**, 329–348 (2004).
16. T. Y. Li, Solving polynomial systems by polyhedral homotopies, *Taiwanese J. Math.* **3**, 251–279 (1999).
17. T. Y. Li and X. Li, Finding mixed cells in the mixed volume computation, *Found. Comput. Math.* **1**, 161–181 (2001). Software available at http://www.math.msu.edu/∼li/.
18. J. T. Linderoth and M. W. P. Savelsbergh, A computational study of branch and bound search strategies for mixed integer programming, *INFORMS J. Comput.* **11**, 173–187 (1999)
19. A. Morgan, *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1987.
20. V. W. Noonburg, A neural network modeled by an adaptive Lotka–Volterra system, *SIAM J. Appl. Math.* **49**, 1779–1792 (1989).
21. A. Takeda, M. Kojima and K. Fujisawa, Enumeration of all solutions of a combinatorial linear inequality system arising from the polyhedral homotopy continuation method, *J. Oper. Res. Soc. Japan* **45**, 64–82 (2002). Software available at http://www.is.titech.ac.jp/∼kojima/index.html.
22. J. Verschelde, The database of polynomial systems is on his web site: http://www.math.uic.edu/∼jan/.

23. J. Verschelde, Algorithm 795: PHCPACK: a general-purpose solver for polynomial systems by homotopy continuation, *ACM Trans. Math. Software* **25**, 251–276 (1999). Software available at http://www.math.uic.edu/∼jan/.
24. J. Verschelde, P. Verlinden and R. Cools, Homotopies exploiting Newton polytopes for solving sparse polynomial systems, *SIAM J. Numer. Anal.* **31**, 915–930 (1994).

Received January 30, 2006, *and in revised form August* 25, 2006. *Online publication March* 9, 2007.