Dynamic Enumeration of All Mixed Cells

Tomohiko Mizutani[†1], Akiko Takeda[†2] and Masakazu Kojima[†3]

January 2006

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. Finding the mixed cells is formulated in terms of a linear inequality system with an additional combinatorial condition. It is essential in computational efficiency how we construct an enumeration tree among a family of linear inequalities induced from it such that every mixed cell corresponds to a unique feasible leaf node. 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 a lot of subtrees which do not contain any mixed cell. Numerical results exhibit that our 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.

Key words.
Mixed Cell, Polyhedral Homotopy Method, Polynomial System, Dynamic Enumeration, Linear Programming.

† Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. †1:*mizutan8@is.titech.ac.jp.* †2:*takeda@is.titech.ac.jp.* †3:*kojima@is.titech.ac.jp.*

1 Introduction

The polyhedral homotopy continuation method [12], which is based on Bernshtein's theory [1], is known to be a powerful numerical method [5, 10, 11, 13, 14, 22] 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 the papers [8, 12] 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. 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 [7, 8, 15, 19] (or a family of systems of linear inequalities) is more efficient in computational time and memory requirement than a geometric formulation used in the papers [22].

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 are their 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 efficient implementation of enumeration of all mixed cells which are corresponding to feasible leaf nodes of an enumeration tree satisfying properties (i), (ii), (iii) and (iv). One is how we check feasibility of each node. For this purpose, the papers [7, 8, 15, 19] utilize a linear programming (LP) problem having a linear system of inequalities attached to each node as its constraint and check feasibility of the linear system. The papers [7, 8, 15] applies the primal simplex method to the LP problem while the paper [19] applies the dual simplex method to the LP problem. If we take account of 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 since the linear inequality constraints of each child node are a super set of the linear inequality constraints of its parent and they share a common linear objective function. At least, the application of the dual simplex method is popular in the field of optimization to effectively deal with such a situation [16]. This paper also applies the dual simplex method to an LP problem attached with each node to check its feasibility, which will be 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), (ii), (iii) and (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. There are lots of 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 [7, 8, 15, 19], the structure of 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 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 [16].

We now describe mixed cells in terms of systems of linear inequalities and the basic idea of our dynamic enumeration of them. For every $L \subseteq N := \{1, 2, \ldots, n\}$, define

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

The set Ω serves as candidates of 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 $\cup_{L \subseteq N,\ \#L=\ell} \Omega(L)$. For every $\boldsymbol{C} \in \Omega$, let $L(\boldsymbol{C}) = \{i \in N : C_i \neq \emptyset\}$. For every $\boldsymbol{C} \in \Omega$ and $L \subseteq N$, we denote the vector consisting of C_i $(i \in L)$ by $\boldsymbol{C}_L = (C_i : i \in L)$.

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 $\mathbb{R}$. The number $\omega_i(\boldsymbol{a})$ is called a lifting in the literature. 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 $\boldsymbol{\alpha} \in \mathbb{R}^n$:

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

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

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

Then $\Omega_*(N)$ defines 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) are satisfied. Thus, for 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 $t \in N \setminus L(\boldsymbol{C})$, define a set of child nodes of $\boldsymbol{C}$ by

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

We can build an enumeration tree if we successively choose $t \in N \setminus L(\boldsymbol{C})$ at each node $\boldsymbol{C}$ of the tree starting from the root node $\boldsymbol{C} = \emptyset^n$ with $L(\boldsymbol{C}) = \emptyset$ (see Algorithm 2.1 for more details). In the static enumeration method employed in the papers [7, 8, 15, 19], we first choose a permutation of N or a one-to-one mapping $\pi : N \to N$, and restrict nodes of enumeration trees to $\boldsymbol{C} \in \Omega(\{\pi(1), \ldots, \pi(\ell)\})$ with some $\ell \in \{0, 1, \ldots, n\}$. Note that if we take $\ell = 0$ or $\ell = n$, we have the root node $\emptyset^n \in \Omega(\emptyset)$ or the set $\Omega(N)$ of leaf nodes, respectively. Suppose that a node $\boldsymbol{C} \in \Omega(\{\pi(1), \ldots, \pi(\ell)\})$ for some $\ell < n$ has been found to be feasible. Then the static enumeration method 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 $\pi : N \to N$.

In the enumeration method that we propose in this paper, an enumeration tree is constructed dynamically as the enumeration of nodes proceeds. 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 $t \in N \setminus L(\boldsymbol{C})$ 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 $s \in N \setminus L(\boldsymbol{C})$. 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 [8] to find some infeasible child nodes in $W(\boldsymbol{C}, s)$.

Numerical results exhibit that our dynamic enumeration method works very efficiently for finding all mixed cells in comparison to the existing static enumeration methods [6, 8, 9, 15, 22, 19]. For instance, our dynamic enumeration method solved the cyclic-14 problem (*i.e.* generates all mixed cell of the cyclic-14 problem), which had been the largest one in cyclic-n problems solved by the existing methods, in 1 hours 36 minutes, while MixedVol [8, 9], which is known as the fastest software among the existing ones, solves the same problem

in 7 hours 14 minutes. Furthermore, our method solved the cyclic-15 problem for the first time in 15 hours 45 minutes. 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 construction of an enumeration tree satisfying properties (i), (ii),(iii) and (iv), and then outline our dynamic enumeration algorithm. Section 3 is devoted to technical details of the algorithm. We first show a LP formulation for checking feasibility of each node in an enumeration tree, and discuss the size of the primal-dual pair of LP problems. Next, we mention how to choose $t \in N \setminus L(\boldsymbol{C})$ at a parent node $\boldsymbol{C}$ so that the number of child nodes of $\boldsymbol{C}$ is as smaller as possible. 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

We first explain how to construct an enumeration tree, which satisfies properties (i), (ii), (iii) and (iv) described in the previous section, and next propose an algorithm for dynamic enumeration of all mixed cells. To explain a procedure for 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 consists of the root node $\emptyset^n$, and we define E_0 as an empty set for consistence with below discussions. The procedure for construction of a tree T with allocating nodes dynamically is written as follows:

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 $\ell \neq n$ **do**
  for all $\boldsymbol{C} \in V_\ell$ **do**
    Choose t from $N \setminus L(\boldsymbol{C})$.
    $V_{\ell+1} \leftarrow V_{\ell+1} \cup W(\boldsymbol{C}, t)$ and $E_{\ell+1} \leftarrow E_{\ell+1} \cup \{(\boldsymbol{C}, \bar{\boldsymbol{C}}) \in V_\ell \times V_{\ell+1} : \bar{\boldsymbol{C}} \in W(\boldsymbol{C}, t)\}$.
  end for
  $\ell \leftarrow \ell + 1$.
end while

This procedure for the input data $\mathcal{A}$ produces a various type of tree T depending on a choice of an index t from $N \setminus L(\boldsymbol{C})$. For instance, a static enumeration tree proposed in the existing algorithm [7, 8, 15, 19] is constructed when for any $\boldsymbol{C} \in V_\ell$ the index t is set to $\pi(\ell + 1)$ according to the given permutation π of N.

If two nodes $\boldsymbol{C} \in V_\ell$ and $\bar{\boldsymbol{C}} \in V_{\ell+1}$ of a tree are joined with a edge, we say that $\bar{\boldsymbol{C}}$ is the child node of a parent node $\boldsymbol{C}$. The descendant node of $\boldsymbol{C}$ is corresponding to any node on

all paths from $\boldsymbol{C}$ to reachable leaf nodes which are elements in V_n. Each feasible leaf nodes are corresponding to mixed cells.

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 to check feasibility of descendant nodes. Furthermore, we employ a depth-first order for applying feasibility check to all nodes of an enumeration tree in order to save memory requirement during execution of enumeration. Taking account of these factors, the depth-first search algorithm for the enumeration of all mixed cells is constructed.

It is convenient to use the words "list" used in this algorithm. If A is a finite set, we denote list(A) is an ordered sequence of the elements in A, where the actual order is not relevant in our succeeding discussions but 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) $= (a, b, c, d, e)$.

Algorithm 2.1. (A general depth-first search algorithm for all mixed cells).

Input: A support $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2 \ldots, \mathcal{A}_n)$.

Output: All mixed cells $\boldsymbol{C} \in \Omega_*(N)$ and an evaluation measure ν_*.

Step 1: Let V_a be the empty list of nodes, and $V_a = \text{list}(V_a) + \text{list}(\Omega(\emptyset))$, where V_a serves as the set of active nodes during the depth-first search. Let $\nu = 1$ which serves as the counter of nodes generated; ν is used only for evaluating the efficiency of the algorithm but not essential in any step below.

Step 2: If V_a is empty, then output $\nu_* = \nu$ and stop.

Step 3: Take out the last element $\boldsymbol{C}$ of V_a and remove $\boldsymbol{C}$ from V_a.

Step 4: Check whether $\boldsymbol{C}$ is feasible or infeasible. If $\boldsymbol{C}$ is infeasible ($\boldsymbol{C} \notin \Omega_*(L(\boldsymbol{C}))$ or $\mathcal{I}(\boldsymbol{C})$ is infeasible) then go to Step 2. If $\boldsymbol{C}$ is feasible ($\boldsymbol{C} \in \Omega_*(L(\boldsymbol{C}))$ or $\mathcal{I}(\boldsymbol{C})$ is feasible) and $L(\boldsymbol{C}) = N$, then output $\boldsymbol{C}$ as a mixed cell and go to Step 2. Otherwise go to Step 5.

Step 5: Choose a t from $N \setminus L(\boldsymbol{C})$ and $\nu = \nu + \#W(\boldsymbol{C}, t)$.

Step 6: Let $V_a = \text{list}(V_a) + \text{list}(W(\boldsymbol{C}, t))$ and go to Step 2.

The total amount of works to generate all mixed cells by Algorithm 2.1 is measured by ν_* which represents the total number of nodes generated during execution of Algorithm 2.1; 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 [7, 8, 15, 19] to narrow down the number of elements in V_a. For

$\boldsymbol{C} \in V_a$ and $t \in N \setminus L(\boldsymbol{C})$, the one point test checks the feasibility of the system of linear inequalities in $\boldsymbol{\alpha} \in \mathbb{R}^n$ with fixed $\boldsymbol{a} \in \mathcal{A}_t$,

$$\mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a}) : \begin{cases} \mathcal{I}(\boldsymbol{C}), \\ \langle \boldsymbol{a} - \boldsymbol{b}, \boldsymbol{\alpha} \rangle \leq \omega_t(\boldsymbol{b}) - \omega_t(\boldsymbol{a}), \quad (\boldsymbol{b} \in \mathcal{A}_t \setminus \{\boldsymbol{a}\}). \end{cases} \tag{1}$$

If $\mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a})$ is infeasible, we can delete $\bar{\boldsymbol{C}} \in W(\boldsymbol{C}, t)$ whose $\bar{C}_t = \{\boldsymbol{a}, \boldsymbol{a}'\}$ consists of $\boldsymbol{a}$ and any $\boldsymbol{a}' \in \mathcal{A}_t \setminus \{\boldsymbol{a}\}$ from the set V_a of solution candidates, since such $\mathcal{I}(\bar{\boldsymbol{C}})$ is also infeasible. Therefore, as solution candidates we only consider

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

where $\mathcal{A}_t(\boldsymbol{C}) = \{\boldsymbol{a} \in \mathcal{A}_t : \mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a}) \text{ is feasible}\}$. 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

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

Thus we can replace Step 6 by

Step 6': Let $V_a = \mathrm{list}(V_a) + \mathrm{list}(W_1(\boldsymbol{C}, t))$ and go to Step 2.

This technique called one point test is known to be very effective to increase the computational efficiency of enumeration [7, 8, 15, 19].

Ideally we would like to choose a $t \in N \setminus L(\boldsymbol{C})$ at Step 5 so that the size of $W_1(\boldsymbol{C}, t)$ is the smallest among the sizes of $W_1(\boldsymbol{C}, s)$ $(s \in N \setminus L(\boldsymbol{C}))$. Finding such a $t \in N \setminus L(\boldsymbol{C})$ exactly, however, is expensive because all $W_1(\boldsymbol{C}, s)$ $(s \in N \setminus L(\boldsymbol{C}))$ are constructed. Therefore, we propose to replace $W_1(\boldsymbol{C}, s)$ by another set which can be obtained easily. Indeed, utilizing a feasible solution $\boldsymbol{x}_{init}$ of $\mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a})$ which is generated from an solution of $\mathcal{I}(\boldsymbol{C})$, our method computes $\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{init})$ $(s \in N \setminus L(\boldsymbol{C}))$ satisfying

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

and chooses a $t \in N \setminus L(\boldsymbol{C})$ such that the size of $\hat{W}_1(\boldsymbol{C}, t, \boldsymbol{x}_{init})$ attains the minimum among the sizes of $\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{init})$ $(s \in N \setminus L(\boldsymbol{C}))$. We call this method a *dynamic enumeration method*. In Subsection 3.2, we explain how to generate the set $\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{init})$ $(s \in N \setminus L(\boldsymbol{C}))$. Also the relation table proposed in [8] 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 $\hat{W}_1(\boldsymbol{C}, s, \boldsymbol{x}_{init})$ is constructed.

3 Technical details of the algorithm

3.1 Formulation of checking feasibility of a system of linear inequalities

The feasibility check of $\boldsymbol{C} \in \Omega$, conducted at Step 4 of Algorithm 2.1, can be formulated via an LP problem. Namely, we test feasibility of the following problem in the vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ of decision variables:

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

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

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

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

$$\boldsymbol{x} = (x_{\boldsymbol{a}} : \boldsymbol{a} \in \mathcal{A}_i \setminus \{\boldsymbol{a}^{p_i}\}, \ i \in L(\boldsymbol{C})) \in \mathbb{R}^{\delta}, \quad \text{where } \delta := \sum_{i \in L(\boldsymbol{C})} (m_i - 1), \tag{2}$$

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

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

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 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

$$\begin{array}{lll} \text{min.} & \langle \boldsymbol{c}, \boldsymbol{x} \rangle & \\ \text{s. t.} & \boldsymbol{G}\boldsymbol{x} = \boldsymbol{h} & \\ & x_i \geq 0, & (i \in I), \\ & -\infty < x_j < +\infty, & (j \in J), \end{array} \tag{3}$$

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 as a nonnegative variable and a free variable, respectively. The primal-dual pair $\mathrm{P}(\boldsymbol{C})$ and $\mathrm{D}(\boldsymbol{C})$ can be transformed into (3) 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

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

On the other hand, $\mathrm{D}(\boldsymbol{C})$ has d_D variables and k_D equalities such that

$$d_D = \textstyle\sum_{i \in L(\boldsymbol{C})} (m_i - 1) \quad \text{and} \quad k_D = n.$$

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 monotonic increasing with respect to the cardinality of $L(\boldsymbol{C})$. When any polynomials

$f_i(\boldsymbol{x})$ $(i \in N)$ have at least two terms, *i.e.*, $m_i \geq 2$, there exists $L' \subseteq N$ such as $k_P \geq k_D$. Consequently for any L such as $L' \subseteq L \subseteq N$, 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 feasibility of $\boldsymbol{C}$.

We formulate the one point test, stated in the previous section, via an LP problem. For every $\boldsymbol{C} \in \Omega,\ t \in N \setminus L(\boldsymbol{C})$ and $\boldsymbol{a} \in \mathcal{A}_t$, checking feasibility of (1) can be written as the following LP problem in the vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ of decision variables:

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

where $\boldsymbol{\gamma} \in \mathbb{R}^n$ is some fixed vector. As the one point test, we check 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

$\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$:

$$\begin{array}{ll} \text{min.} & \Phi(\boldsymbol{x}; \boldsymbol{C}) + \displaystyle\sum_{\boldsymbol{b} \in \mathcal{A}_t \setminus \{\boldsymbol{a}\}} (\omega_t(\boldsymbol{b}) - \omega_t(\boldsymbol{a}))\, y_{\boldsymbol{b}} \\ \text{s. t.} & \Psi(\boldsymbol{x}; \boldsymbol{C}) + \displaystyle\sum_{\boldsymbol{b} \in \mathcal{A}_t \setminus \{\boldsymbol{a}\}} (\boldsymbol{a} - \boldsymbol{b})\, y_{\boldsymbol{b}} = \boldsymbol{\gamma} \\ & x_{\boldsymbol{a}} \geq 0 \ (\boldsymbol{a} \in \mathcal{A}_i \setminus \{\boldsymbol{a}^{p_i}, \boldsymbol{a}^{q_i}\}) \text{ and } -\infty < x_{\boldsymbol{a}^{q_i}} < +\infty, \text{ for } i \in L(\boldsymbol{C}), \\ & y_{\boldsymbol{b}} \geq 0 \ (\boldsymbol{b} \in \mathcal{A}_t \setminus \{\boldsymbol{a}\}). \end{array}$$

Using $\boldsymbol{x} \in \mathbb{R}^\delta$ of (2) and the column vector $\boldsymbol{y} = (y_{\boldsymbol{b}} : \boldsymbol{b} \in \mathcal{A}_t \setminus \{\boldsymbol{a}^{p_t}\}) \in \mathbb{R}^{(m_t - 1)}$, the vector of decision variables in this problem is represented as

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

This primal-dual pair satisfies the duality theorem because we set $\boldsymbol{\gamma}$ so that $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ is feasible. 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 $\boldsymbol{C} \in \Omega,\ t \in N \setminus L(\boldsymbol{C})$ 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, let us consider the problem $\mathrm{D}(\boldsymbol{C})$. Using the arbitrary nonnegative vector $\hat{\boldsymbol{x}} \in \mathbb{R}^\delta$, 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 $t \in N \setminus L(\boldsymbol{C})$. As a result, $\mathrm{D}(\boldsymbol{C})$ is feasible. Let us suppose that the problem $\mathrm{D}(\boldsymbol{C})$ is bounded, and denote an optimal solution of this problem by $\boldsymbol{x}_* \in \mathbb{R}^\delta$. Then, the vector

$$\boldsymbol{x}_{init} = \begin{pmatrix} \boldsymbol{x}_* \\ \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{\bar{\delta}} \tag{4}$$

is feasible solution in $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \setminus L(\boldsymbol{C})$) because $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ with fixed $\boldsymbol{y} = (y_{\boldsymbol{b}} : \boldsymbol{b} \in \mathcal{A}_t \setminus \{\boldsymbol{a}^{p_t}\}) = \boldsymbol{0}$ 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 a lot of LP problems with a similar structure, we employ the method to solve problems arising from checking feasibility of linear inequality systems.

3.2 How to choose an index t from $N \setminus L(\boldsymbol{C})$

Choice of a $t \in N \setminus L(\boldsymbol{C})$ at Step 5 of Algorithm 2.1 has a major effect on computational efficiency of this algorithm. As stated in Section 2, we want to choose a $t \in N \setminus L(\boldsymbol{C})$ such that the size of $W_1(\boldsymbol{C}, s)$ is the smallest among $s \in N \setminus L(\boldsymbol{C})$. However, this task is expensive in general because we need to check feasibility of $\mathcal{I}(\boldsymbol{C}, t, \boldsymbol{a})$ for every $\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \setminus L(\boldsymbol{C})$ at Step 5 additionally. We will employ a less expensive technique to choose a $t \in N \setminus L(\boldsymbol{C})$ so that an evaluation measure ν_* of efficiency of our dynamic enumeration method becomes smaller.

To check 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 Step 5 we have an optimal solution $\boldsymbol{x}_* \in \mathbb{R}^{\delta}$ of $\mathrm{D}(\boldsymbol{C})$. As stated in the previous subsection, if we set $\boldsymbol{x}_{init} \in \mathbb{R}^{\bar{\delta}}$ by (4) using $\boldsymbol{x}_*$ of $\mathrm{D}(\boldsymbol{C})$, the vector $\boldsymbol{x}_{init}$ is a feasible solution of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ for any $\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \setminus L(\boldsymbol{C})$. Because the structure of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ and $\mathrm{D}(\boldsymbol{C})$ is similar to each other, we usually require only a few iterations to solve $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ when using $\boldsymbol{x}_{init}$ as an initial feasible solution of the simplex method. Thus we expect that $\boldsymbol{x}_{init}$ is incident to unbounded directions in cases where corresponding problems are unbounded. Accordingly, instead of applying the simplex method to check the feasibility of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \setminus L(\boldsymbol{C})$) , we propose to test whether the feasible solution $\boldsymbol{x}_{init}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ has unbounded directions or not.

At Step 5 we consider

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

where

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

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

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

In general, this index $\hat{t}$ does not coincide with the index t which achieves the minimum number of elements in $W_1(\boldsymbol{C}, s)$ ($s \in N \setminus L(\boldsymbol{C})$). We will observe from numerical results, however, that the evaluation measure ν_* is much smaller for our dynamic enumeration method than 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 $\hat{W}_1(\boldsymbol{C}, t, \boldsymbol{x}_{init})$ ($t \in N \setminus L(\boldsymbol{C})$) more precisely, using an LP form of (3) instead of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ for simplicity of notation. For (3), let us 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 (3) is feasible, there exists a vertex $\boldsymbol{x} \in \mathbb{R}^d$ on the feasible region which consists of two components of the vector of basic variables $\boldsymbol{x}_B = \boldsymbol{G}_B^{-1}\boldsymbol{h} \in \mathbb{R}^k$ and of 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 $j \in D \setminus B$. The direction $\boldsymbol{d}$ is composed of two components vectors $\boldsymbol{d}_B$ and $\boldsymbol{d}_N$ such that

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

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

$$\langle \boldsymbol{c}, \boldsymbol{d} \rangle = c_j - \boldsymbol{c}_B^T \boldsymbol{G}_B^{-1} \boldsymbol{g}_j \tag{5}$$

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 (3), 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 $\boldsymbol{x}_{init}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \setminus L(\boldsymbol{C})$) 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 $\boldsymbol{x}_{init}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ for any $\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \setminus L(\boldsymbol{C})$) because the number of constraints is equal to each other, and a feasible solution $\boldsymbol{x}_{init}$ can be represented as (4) 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 (5) 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 $t \in N \setminus L(\boldsymbol{C})$, reduced costs on $\boldsymbol{x}_{init}$ 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

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

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 $\boldsymbol{b} \in \mathcal{A}_i \setminus \{\boldsymbol{a}^{p_i}\}$ and $i \in L(\boldsymbol{C})$. Consequently, if there are $\boldsymbol{b} \in \mathcal{A}_t \setminus \{\boldsymbol{a}^{p_t}\}$ 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 nonnegative variables in the basis, are nonnegative,

we see that the feasible solution $\boldsymbol{x}_{init}$ 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 $\boldsymbol{x}_{init}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a}^{p_t})$ has no such $\boldsymbol{b} \in \mathcal{A}_t \setminus \{\boldsymbol{a}^{p_t}\}$, $\boldsymbol{a}^{p_t}$ is added to $\hat{\mathcal{A}}_t(\boldsymbol{C}, \boldsymbol{x}_{init})$.

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 $\boldsymbol{x}_{init}$ of $\mathrm{D}_1(\boldsymbol{C}, t, \boldsymbol{a})$ ($\boldsymbol{a} \in \mathcal{A}_t$ and $t \in N \setminus L(\boldsymbol{C})$) 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 2.4GHz Opteron 850 with 8 GB memory, running Linux. First, let us observe an evaluation measure ν_* generated by the static and dynamic enumeration, described in Section 2, for the cyclic-n [2] and noon-n [18] problems. In the cyclic-n problem, one polynomial has 2 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 top as $\mathcal{A}_i$ and set the support set $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$ as the input data of Algorithm 2.1.

Table 1 shows an evaluation measure 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 below row "Ratio" indicates the ratio between ν_* given by these two methods. For these systems, this table reveals efficiency of the dynamic enumeration method by comparison with the static one, and we can see that the ratio increases as the size of each system becomes larger.

Table 1: An evaluation measure ν_* generated by the static and dynamic enumeration method 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

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 [6], MixedVol [8, 9], PHCpack [22], PHoM [19] and mvol [15]. In particular, the papers [8, 9] report the superiority of MixedVol, which is coded in C++ language, in the computational time over the other existing software packages. Therefore, we compare our algorithm with MixedVol for the benchmark systems: the economic-n [17], chandra-n [4] and katsura-n [3] problems in addition to the cyclic-n and noon-n problems. Here, we have omitted the description of the economic-n, chandra-n and katsura-n problems, which are found in the web site [20]. We summarize these computational times in Table 2. The right-hand 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 column "Mixed Volume" presents the mixed volume of the support $\mathcal{A}$ of the system, which provides an upper bound for the number of isolated zeros of the polynomial system in $(\mathbb{C} \setminus \{0\})^n$. See [1, 12] for a detail description of the mixed volume, and also [8, 14, 15] for the computation of the mixed volume. In Table 2 we can see that our dynamic enumeration method improves cpu time necessary for finding all mixed cells considerably, and solves large scale polynomial system 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 efficiency, 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 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 the feasibility of a linear inequality system by solving the induced dual LP problem with effective use

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

System	Size (n)	Mixed Volume	Our Algorithm	MixedVol	Speed-up Ratio
Cyclic-n	12	500,352	1m8.8s	4m43.0s	4.11
	13	2,704,156	10m54.7s	49m57.4s	4.58
	14	8,795,976	1h36m37.1s	7h14m24.1s	4.50
	15	35,243,520	15h45m26.0s	-	
Noon-n	16	43,046,689	1m4.9s	33m54.8s	31.38
	17	129,140,129	3m13.1s	2h25m20.8s	45.15
	18	387,420,453	7m38.3s	8h23m19.6s	65.90
	19	1,162,261,429	28m1.0s	-	
	20	3,486,784,361	1h8m49.6s	-	
	21	10,460,353,161	3h59m44.1s	-	
Economic-n	17	32,768	4m56.1s	20m41.8s	4.19
	18	65,536	19m31.8s	1h17m56.0s	3.99
	19	131,072	1h21m30.4s	4h56m4.6s	3.63
	20	262,144	5h41m54.4s	-	
Chandra-n	17	65,536	1m14.4s	33m13.4s	26.80
	18	131,072	3m37.0s	2h14m15.3s	37.13
	19	262,144	10m24.3s	8h19m6.3s	47.97
	20	524,288	35m24.1s	-	
	21	1,048,576	1h30m37.5s	-	
	22	2,097,152	4h34m18.1s	-	
Katsura-n	12	4,096	1m4.2s	14m3.5s	13.13
	13	8,192	7m37.7s	1h21m19.4s	10.66
	14	16,384	37m21.5s	7h54m29.4s	12.70
	15	32,768	3h9m9.3s	-	
	16	65,536	20h43m39.4s	-	

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 to the existing algorithms which utilizes 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, 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 computes all solutions, which are not available so far, for large scale polynomial systems.

References

[1] D. N. Bernshtein, "The number of roots of a system of equations," *Functional Analysis and 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 Computation* **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 Computation* **2**, 83–98 (1986).

[4] S. Chandrasekhar, "Radiative Transfer," Dover, NY, 1960.

[5] Y. Dai, S. Kim and M. Kojima, "Computing all nonsingular solutions of cyclic-n polynomial using polyhedral homotopy continuation methods", *J. Computational and Applied Mathematics* **152**, 83-97 (2003).

[6] I. Z. Emiris and J. F. Canny, "Efficient incremental algorithms for the sparse resultant and the mixed volume", *J.Symbolic Computation* **20**, 117–149 (1995). Software available at http://cgi.di.uoa.gr/~emiris/index-eng.html.

[7] T. Gao and T. Y. Li, "Mixed volume computation via linear programming," *Taiwan Journal of Mathematics* **4**, 599–619 (2000).

[8] T. Gao and T. Y. Li, "Mixed volume computation for semi-mixed systems," *Discrete and Computational Geometry* **29**, 257–277 (2003).

[9] T. Gao, T. Y. Li and M. Wu "MixedVol: A Software Package for Mixed Volume Computation," To appear in *ACM Transactions on Math. Software* **31**(4), (2005). Software available at http://www.csulb.edu/~tgao/.

[10] T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa and T. Mizutani, "PHoM – a Polyhedral Homotopy Continuation Method." *Computing* **73**, 53–77 (2004).

[11] T. Gunji, S. Kim, K. Fujisawa and M. Kojima, "PHoMpara – Parallel implementation of the polyhedral homotopy continuation method", Research Report B-419, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan, October 2005.

[12] B. Huber and B. Sturmfels, "A Polyhedral method for solving sparse polynomial systems," *Mathematics of Computation* **64**, 1541–1555 (1995).

[13] S. Kim and M. Kojima, "Numerical Stability of Path Tracing in Polyhedral Homotopy Continuation Methods", *Computing* **73**, 329–348 (2004).

[14] T. Y. Li, "Solving polynomial systems by polyhedral homotopies", *Taiwan Journal of Mathematics* **3**, 251–279 (1999).

[15] T. Y. Li and X. Li, "Finding Mixed Cells in the Mixed Volume Computation," *Foundation of Computational Mathematics* **1**, 161–181 (2001). Software available at http://www.math.msu.edu/~li/.

[16] J. T. Linderoth and M. W. P. Savelsbergh, "A Computational Study of Branch and Bound Search Strategies for Mixed Integer Programming," *INFORMS Journal on Computing* **11**, 173–187 (1999)

[17] A. Morgan, "Solving polynomial systems using continuation for engineering and scientific problems," Pentice-Hall, New Jersey, 1987.

[18] V. W. Noonburg, "A neural network modeled by an adaptive Lotka-Volterra system," *SIAM J. Appl. Math.* **49**, 1779–1792 (1989).

[19] 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. of Operations Society of Japan* **45**, 64–82 (2002). Software available at http://www.is.titech.ac.jp/~kojima/index.html.

[20] J. Verschelde, The database of polynomial systems is in his web site: "http://www.math.uic.edu/~jan/."

[21] J. Verschelde, P. Verlinden and R. Cools, "Homotopies exploiting Newton polytopes for solving sparse polynomial systems," *SIAM J. Numerical Analysis* **31**, 915–930 (1994).

[22] J. Verschelde, "Algorithm 795: PHCPACK: A general-purpose solver for polynomial systems by homotopy continuation," *ACM Transactions on Mathematical Software* **25**, 251–276 (1999). Software available at http://www.math.uic.edu/~jan/.

[23] V. Chvátal, "LINEAR PROGRAMMING," W.H.Freeman, 1983.