数理解析研究所講究録
1175巻 2000年 63-71

Gröbner Bases of Acyclic Directed Graphs and Reductions in Conti-Traverso Algorithm

石関 隆幸 (Takayuki Ishizeki)
Department of Information Science, Graduate School of Science, The University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo 113–0033, Japan
ishizeki@is.s.u-tokyo.ac.jp

Abstract

Algebraic approaches to some optimization problems on graphs applying Conti-Traverso algorithm to Gröbner bases of finite graphs have been studied in recent years. On the other hand, the complexity of normal form algorithms in Conti-Traverso algorithm is not well understood. In this paper, we study the number of steps of reductions in Conti-Traverso algorithm. We especially focus on the minimum cost flow problems and the transportation problems on acyclic directed graphs, and experiment the number of reductions in Conti-Traverso algorithm.

1 Introduction

Conti-Traverso [3] showed an algorithm based on Gröbner bases for toric ideals to solve integer programs. Some approaches to network problems by applying this algorithm to the vertex-edge incidence matrices of graphs have been studied in recent years [7, 8, 10, 11]. On the other hand, for the normal form algorithm which is used in Conti-Traverso algorithm, although the worst case complexity of the number of reduction in general case has been studied [6], the number of reduction for the special case such as Gröbner bases for toric ideals are not known.

The number of elements in Gröbner bases of graphs is related to the complexity of normal form algorithm, and for the case of complete graphs, complete bipartite graphs and acyclic directed graphs, the number of elements in Gröbner bases for some term orders remain in polynomial order [7, 8, 11]. In particular, for the case of acyclic directed graphs the number of elements in Gröbner bases seems to remain in polynomial order for any term order.

In this paper, we analyze the complexity of normal form algorithm in Conti-Traverso algorithm for the case that Gröbner bases are already given. Especially we focus on the minimum cost flow problems and the transportation problems, and examine the number of reductions in normal form algorithm during Conti-Traverso algorithm when we apply some methods on negative cycle canceling method [1] for the minimum cost flow problems.

2 Preliminaries

In this section, we give basic definitions of Gröbner bases, toric ideals and Conti-Traverso algorithm. We refer to [4, 5] for the introductions of Gröbner bases, [12] for the introductions of toric ideals, and [3, 14] for the Conti-Traverso algorithm.

2.1 Gröbner Bases

Let k be a field and $k[\mathbf{x}] := k[x_1, \ldots, x_n]$ be the ring of polynomials in n variables. For a set of variables $\mathbf{x} = (x_1, \ldots, x_n)$ and a non-negative integer vector $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n$ ($\mathbb{Z}_{\geq 0}$ means the set of all non-negative integers), we denote $\mathbf{x}^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. For a fixed term order $\succ$ and $f \in k[\mathbf{x}]$, we call the largest term in f with respect to $\succ$ the *initial term* of f and write $in_\succ(f)$.

Definition 2.1 *Fix a term order $\succ$, like a lexicographic order, and let $\mathbf{c} \in \mathbb{R}_{\geq 0}^n$ be a non-negative vector. We define a* refinement $\succ_\mathbf{c}$ *of* $\mathbf{c}$ *with respect to $\succ$ such that*

$$\mathbf{x}^\alpha \succ_\mathbf{c} \mathbf{x}^\beta \Longleftrightarrow \mathbf{c} \cdot \alpha > \mathbf{c} \cdot \beta \text{ or } (\mathbf{c} \cdot \alpha = \mathbf{c} \cdot \beta \text{ and } \mathbf{x}^\alpha \succ \mathbf{x}^\beta).$$

Definition 2.2 *Let $f, g_1, \ldots, g_s \in k[\mathbf{x}]$. f is* reducible *by g_i if there exists some term f_i of f such that f_i is divisible by $in_\succ(g_i)$. Then we write $f \xrightarrow[g_i]{} r$ where $r = f - \dfrac{f_i}{in_\succ(g_i)} g_i$.*

f is reducible *by $G = \{g_1, \ldots, g_s\}$ when f is reducible by some $g_i \in G$, and then we write $f \xrightarrow[G]{} r$. In addition, if*

$$f \xrightarrow[G]{} f_1 \xrightarrow[G]{} f_2 \xrightarrow[G]{} \cdots \xrightarrow[G]{} f_k \tag{1}$$

and f_k is not reducible by G, then f_k is called a normal form *of f by G and written as $\overline{f}^G$.*

In general, the normal form of f by G is not unique. The central idea in Gröbner basis is to enlarge G so that the normal form of any polynomial becomes unique.

Example 2.3 *Let $f = xy^2 - x$ and $G = \{g_1 = xy + 1, g_2 = y^2 - 1\}$ in $k[x, y]$ and $\succ$ be a lexicographic order such that $x \succ y$. Then $-x - y$ is the normal form of f since $f \xrightarrow[g_1]{} -x - y$, and 0 is also the normal form of f since $f \xrightarrow[g_2]{} 0$.*

Let enlarge G to $G' = \{g_1, g_2, g_3 = x + y\}$. Then the normal form of f is only 0. ∎

Definition 2.4 *Let $I \subseteq k[\mathbf{x}]$ be an ideal and fix a term order $\succ$. $G = \{g_1, \ldots, g_s\} \subseteq k[\mathbf{x}]$ is a* Gröbner basis *for I with respect to $\succ$ if G satisfies the following:*

1. *I is generated by $g_1, \ldots, g_s$, and*
2. *for any $f \in k[\mathbf{x}]$, the normal form $\overline{f}^G$ is unique.*

In addition, Gröbner basis G is reduced *if G satisfies the following:*

1. *for any i, the coefficient of $in_\succ(g_i)$ is 1, and*
2. *for any i, any term of g_i is not divisible by $in_\succ(g_j)$ $(i \neq j)$.*

The following is the algorithm which calculates one of the normal forms of $f \in k[\mathbf{x}]$ by a polynomial set $G = \{g_1, \ldots, g_s\}$.

Algorithm 2.5 (Normal Form Algorithm(1))
Input: $f \in k[\mathbf{x}]$, $G = \{g_1, \ldots, g_s\} \in k[\mathbf{x}]$ *and* $\succ$
Output: *One of the normal forms* $\overline{f}^G$
WHILE *there exists a term f_i in f and $g_j \in G$ such that $in_\succ(g_j) | f_i$ and $f \neq 0$* **do**

$$f := f - \frac{f_i}{in_\succ(g_j)} g_j$$

Output $\overline{f}^G := f$.

For the number of iterations of "WHILE" loop in this algorithm, only the upper bound has known [6].

Theorem 2.6 ([6]) *For a fixed term order $\succ$, $f \in k[\mathbf{x}]$ and $G = \{g_1, \ldots, g_s\}$, the number of iterations $L_G(f)$ of "WHILE" loop in Algorithm 2.5 is at most*

$$L_G(f) \leq \begin{cases} L & l = 1 \\ (1 + R_{G,\succ} U)L & l = 2 \\ 2^{R_{G,\succ} U} L & l \geq 3 \end{cases}$$

where L is the number of terms in f, l is the maximum of the number of terms in $g_i \in G$, $R_{G,\succ} \geq cd^n$ (c and $d(\geq 2)$ is a constant which depends on $\succ$ and G), and $U = O(\deg(f))$ where $\deg(f)$ is the total degree of f.

2.2 Toric Ideals

Fix a matrix $A \in \mathbb{Z}^{d \times n}$ and let $\mathbf{a}_i$ be the i-th column of A. Each vector $\mathbf{a}_i$ is identified with a monomial $\mathbf{t}^{\mathbf{a}_i}$ in the Laurent polynomial ring $k[\mathbf{t}^{\pm 1}] := k[t_1, \ldots, t_d, t_1^{-1}, \ldots, t_d^{-1}]$.

Definition 2.7 *Consider the homomorphism*

$$\pi \colon k[x_1, \ldots, x_n] \longrightarrow k[\mathbf{t}^{\pm 1}], \ x_i \longmapsto \mathbf{t}^{\mathbf{a}_i}.$$

The kernel of π is denoted I_A and called the toric ideal *of A.*

Every vector $\mathbf{u} \in \mathbb{Z}^n$ can be written uniquely as $\mathbf{u} = \mathbf{u}^+ - \mathbf{u}^-$ where $\mathbf{u}^+$ and $\mathbf{u}^-$ are non-negative and have disjoint support.

Lemma 2.8

$$I_A = \langle \mathbf{x}^{\mathbf{u}_i^+} - \mathbf{x}^{\mathbf{u}_i^-} : \mathbf{u}_i \in \ker(A) \cap \mathbb{Z}^n, i = 1, \ldots, s \rangle$$

Furthermore, a toric ideal is generated by finite binomials.

Definition 2.9 *A binomial $\mathbf{x}^{\mathbf{u}^+} - \mathbf{x}^{\mathbf{u}^-} \in I_A$ is called* circuit *if the support of $\mathbf{u}$ is minimal with respect to inclusion in* $\ker(A)$ *and the coordinates of $\mathbf{u}$ are relatively prime. We denote the set of all circuits in I_A by $\mathcal{C}_A$.*

Definition 2.10 *A binomial $\mathbf{x}^{\mathbf{u}^+} - \mathbf{x}^{\mathbf{u}^-} \in I_A$ is called* primitive *if there exists no other binomial $\mathbf{x}^{\mathbf{v}^+} - \mathbf{x}^{\mathbf{v}^-} \in I_A$ such that both $\mathbf{u}^+ - \mathbf{v}^+$ and $\mathbf{u}^- - \mathbf{v}^-$ are non-negative. The set of all primitive binomials in I_A is called the* Graver basis *of A and written as Gr_A.*

Let $\mathcal{U}_A$ be the set which is the union of all Gröbner bases for I_A with respect to all term orders. $\mathcal{U}_A$ is a Gröbner basis for I_A with respect to any term order, and called the *universal Gröbner basis* of I_A.

Proposition 2.11 ([12, Proposition 4.11.]) *For any matrix A,*

$$\mathcal{C}_A \subseteq \mathcal{U}_A \subseteq Gr_A.$$

2.3 Conti-Traverso Algorithm

Conti and Traverso [3] introduced an algorithm based on Gröbner basis to solve integer programs. We describe the condensed version of Conti-Traverso Algorithm(See [12]). This version is useful for highlighting the main computational step involved. For the original version of Conti-Traverso Algorithm, see [3].

Let $A \in \mathbb{Z}^{d\times n}$, $\mathbf{b} \in \mathbb{Z}^d$, $\mathbf{c} \in \mathbb{R}^n_{\geq 0}$. We consider the integer program

$$IP_{A,\mathbf{c}}(\mathbf{b}) := minimize\{\mathbf{c}\cdot\mathbf{x}\colon A\mathbf{x} = \mathbf{b}, \mathbf{x} \in \mathbb{Z}^n_{\geq 0}\}.$$

Conti-Traverso Algorithm is a algorithm using the toric ideal I_A which calculates $\mathbf{x}$ such that $\mathbf{x}$ is minimum in the set $\{A\mathbf{x} = \mathbf{b}, \mathbf{x} \in \mathbb{Z}^n_{\geq 0}\}$ with respect to the term order $\succ_{\mathbf{c}}$, that is one of the optimal solution of $IP_{A,\mathbf{c}}$. Let $IP_{A,\succ_{\mathbf{c}}}(\mathbf{b})$ the problem that calculate this $\mathbf{x}$.

Algorithm 2.12 (Conti-Traverso Algorithm)
Input: $A \in \mathbb{Z}^{d\times n}$, $\mathbf{b} \in \mathbb{Z}^d$, $\mathbf{c} \in \mathbb{R}^n_{\geq 0}$
Output: *An optimal solution* $\mathbf{u}'$ *for* $IP_{A,\succ_{\mathbf{c}}}(\mathbf{b})$
1. *Compute the reduced Gröbner basis* $G_{\succ_{\mathbf{c}}}$ *of* I_A *with respect to* $\succ_{\mathbf{c}}$.
2. *For any solution* $\mathbf{u}$ *of* $IP_{A,\mathbf{c}}(\mathbf{b})$, *compute the normal form* $\mathbf{x}^{\mathbf{u}'}$ *of* $\mathbf{x}^{\mathbf{u}}$ *by* $G_{\succ_{\mathbf{c}}}$.
3. *Output* $\mathbf{u}'$. $\mathbf{u}'$ *is the unique optimal solution of* $IP_{A,\succ_{\mathbf{c}}}(\mathbf{b})$.

Conti-Traverso algorithm has given insight into the structure of integer programming by associating reduced Gröbner bases with *test sets* in integer programming [13].

3 Application to Toric Ideals of Acyclic Directed Graphs

Let $G = (V, E)$ be an acyclic directed graph such that $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. The *vertex-edge incidence matrix* $A = (a_{ij})$ of G is $n \times m$ integer matrix such that

$$a_{ij} = \begin{cases} 1 & v_i \text{ is the initial vertex of } e_j \\ -1 & v_i \text{ is the terminal vertex of } e_j \\ 0 & \text{otherwise} \end{cases}.$$

With regard to the toric ideal of the incidence matrices of graphs, applications of Conti-Traverso algorithm to complete graphs, complete bipartite graphs, directed graphs, and transportation of the incidence matrices of undirected graphs correspond to solve the minimum weight perfect f-matching problems [7], the transportation problems [8], the minimum cost flow problems [11], and the vertex covering problems [10], respectively.

3.1 Toric Ideals of Acyclic Tournament Graphs

In this section, we consider the toric ideals of acyclic tournament graphs. The toric ideals of acyclic tournament graphs are important since, as we will show in the next section, the Gröbner bases for toric ideals of acyclic directed graphs or bipartite graphs can be obtained from those for acyclic tournament graphs automatically.

Let D_n be an acyclic tournament graph with n vertices which have labels $1, 2, \ldots, n$ such that each edge (i, j) $(i < j)$ is directed from i to j. Let $m = \binom{n}{2}$ be the number of edges in D_n. We associate each edge (i, j) with a variable x_{ij} in the polynomial ring $k[\mathbf{x}] := k[x_{ij}\colon 1 \leq i < j \leq n]$. Let A_n be the vertex-edge incidence matrix of D_n.

A *walk* in D_n is the sequence of vertices $(v_1, v_2, \ldots, v_p)$ such that (v_i, v_{i+1}) or (v_{i+1}, v_i) is an arc of D_n for each $1 \leq i < p$. A *cycle* is a walk $(v_1, v_2, \ldots, v_p, v_1)$. A *circuit* is a cycle $(v_1, v_2, \ldots, v_p, v_1)$ such that $v_i \neq v_j$ for any $i \neq j$.

Definition 3.1 *Let C be a circuit of D_n. If we fix a direction of C, we can partition the edges of C into two sets C^+ and C^- such that C^+ is the set of forward edges and C^- is the set of backward edges. Then the vector $\mathbf{u} = (u_{ij})_{1\leq i<j\leq n} \in \mathbb{R}^m$ defined by*

$$u_{ij} = \begin{cases} 1 & \text{if } (i,j) \in C^+ \\ -1 & \text{if } (i,j) \in C^- \\ 0 & \text{if } (i,j) \notin C \end{cases}$$

is called the incidence vector *of C.*

Lemma 3.2 ([2, Proposition 2.17]) *A binomial $\mathbf{x}^{\mathbf{u}^+} - \mathbf{x}^{\mathbf{u}^-} \in I_{A_n}$ is a circuit if and only if $\mathbf{u}$ is the incidence vector of a circuit of D_n.*

Proposition 3.3 ([12, Exercise 4(8)]) *For the case of I_{A_n}, $\mathcal{C}_{A_n} = \mathcal{U}_{A_n} = Gr_{A_n}$.*

(Proof) If $\mathbf{x}^{\mathbf{u}^+} - \mathbf{x}^{\mathbf{u}^-} \in Gr_{A_n}$ is not a circuit of I_{A_n}, then there exists a circuit $\mathbf{x}^{\mathbf{c}^+} - \mathbf{x}^{\mathbf{c}^-} \in I_{A_n}$ such that

$$supp(\mathbf{c}^+) \subseteq supp(\mathbf{u}^+), \quad supp(\mathbf{c}^-) \subseteq supp(\mathbf{u}^-).$$

By Lemma 3.2, since each element in $\mathbf{c}^+$ and $\mathbf{c}^-$ is either 0 or 1, $\mathbf{x}^{\mathbf{u}^+}$ is divisible by $\mathbf{x}^{\mathbf{c}^+}$ and $\mathbf{x}^{\mathbf{u}^-}$ is divisible by $\mathbf{x}^{\mathbf{c}^-}$. Then $\mathbf{u}$ is not primitive, which is contradiction. ∎

Corollary 3.4 *The universal Gröbner basis $\mathcal{U}_{A_n}$ is the set of binomials which correspond to all of the circuits of D_n.*

Corollary 3.5 *The number of elements in $\mathcal{U}_{A_n}$ is of exponential order with respect to n.*

3.2 Gröbner Bases for Acyclic Directed Graphs

We now consider the toric ideals of acyclic directed graphs and (undirected) bipartite graphs.

Let B_n be the vertex-edge incidence matrix of acyclic directed graph G_n with n vertices and $C_{m,n}$ that of bipartite graph $K_{m,n}$ with vertex sets V, W such that $|V| = m$, $|W| = n$.

We consider G_n as a subgraph of D_n, and let

$$E' := \{(i,j) \colon (i,j) \in E(D_n) \setminus E(G_n)\}$$

where $E(D_n)$ (resp. $E(G_n)$) is the edge set of D_n (resp. G_n).

Proposition 3.6 $I_{B_n} = I_{A_n} \cap k[x_{ij} \colon (i,j) \notin E']$.

(Proof) If $f = \mathbf{x}^{\mathbf{c}^+} - \mathbf{x}^{\mathbf{c}^-} \in I_{B_n}$, there exists a cycle C in G_n such that for a suitable orientation of C, the support of $\mathbf{c}^+$ is the set of forward edges in C and the support of $\mathbf{c}^-$ is the set of backward edges in C. Then C is also a cycle in D_n, which implies that $f \in I_{A_n} \cap k[x_{ij} \colon (i,j) \notin E']$.

Conversely, Let $f = \mathbf{x}^{\mathbf{c}^+} - \mathbf{x}^{\mathbf{c}^-} \in I_{A_n} \cap k[x_{ij} \colon (i,j) \notin E']$. Since $f \in I_{A_n}$, there exists a cycle C in D_n such that for a suitable orientation of C, the support of $\mathbf{c}^+$ is the set of forward edges in C and the support of $\mathbf{c}^-$ is the set of backward edges in C. Furthermore, since $f \in k[x_{ij} \colon (i,j) \notin E']$, C contains no edge which is contained in E'. Then C is also a cycle in G_n, which implies that $f \in I_{B_n}$. ∎

Let $G_{m,n}$ be a subgraph of D_{m+n} such that the edge set $E(G_{m,n})$ of $G_{m,n}$ is

$$E(G_{m,n}) := \{(i,j) \colon 1 \leq i \leq m \text{ and } m+1 \leq j \leq m+n\}.$$

$G_{m,n}$ is obtained from $K_{m,n}$ by orienting each edge from the vertex in V to the vertex in W. Let $C'_{m,n}$ be the vertex-edge incidence matrix of $G_{m,n}$.

Proposition 3.7 $I_{C'_{m,n}} = I_{C_{m,n}} = I_{D_{m+n}} \cap k[x_{ij} : (i,j) \in E(G_{m,n})]$.

(**Proof**) The i-th row of $C'_{m,n}$ equals the i-th row of $C_{m,n}$ for $1 \leq i \leq m$ and (-1) times the i-th row of $C_{m,n}$ for $m+1 \leq i \leq m+n$. Thus $I_{C'_{m,n}} = I_{C_{m,n}}$ since $\ker(C'_{m,n}) = \ker(C_{m,n})$.

The proof of the second equality is similar to that of Proposition 3.6. ∎

By Proposition 3.6 and Proposition 3.7, Gröbner bases of acyclic directed graphs and bipartite graphs can be obtained from those for acyclic tournament graphs using the following *elimination theorem.*

Theorem 3.8 ([4, Chapter 3, §3. Theorem 2 and Exercise 5]) *Fix an integer $1 \leq l \leq n$ and let $\succ$ be a term order on $k[x_1, x_2, \ldots, x_n]$ such that any monomial involving at least one of $x_1, \ldots, x_l$ is greater than all monomials in $k[x_{l+1}, \ldots, x_n]$. Let $\succ'$ be a term order which is the restriction of $\succ$ to $k[x_{l+1}, \ldots, x_n]$. If I is an ideal in $k[x_1, x_2, \ldots, x_n]$ and G is a Gröbner basis of I with respect to $\succ$, then $G \cap k[x_{l+1}, \ldots, x_n]$ is a Gröbner basis for $I \cap k[x_{l+1}, \ldots, x_n]$ with respect to $\succ'$.*

3.3 Conti-Traverso Algorithm for Acyclic Directed Graphs

Using Algorithm 2.12, reduced Gröbner bases for I_{A_n} can be applied to minimum cost flow problems on D_n or the subgraphs of D_n, or to transportation problem on the bipartite graphs $K_{m,n}$.

Minimum cost flow can be solved by the *cycle canceling algorithm*, that is, for a feasible flow the algorithm iteratively finds negative cost directed cycles in the residual network and augments flows on these cycles. If the residual network contains no negative cost cycle, then the flow is the minimum cost flow [1]. Since the elements of Gröbner bases for acyclic directed graphs correspond to the circuits of the graphs by Corollary 3.4, the cycle canceling algorithm corresponds to Conti-Traverso algorithm in which the reduction is done by the universal Gröbner basis. Since in the cycle canceling algorithm we augment by each cycle as much as possible, corresponding normal form algorithm in Conti-Traverso algorithm is rather different with the original normal form algorithm in Algorithm 2.5.

Algorithm 3.9 (Normal Form Algorithm(2))
Input: $f \in k[\mathbf{x}]$, $G = \{g_1, \ldots, g_s\} \in k[\mathbf{x}]$ *and* $\succ$
Output: *One of the normal forms* $\overline{f}^G$
WHILE *there exists a term* f_i *in* f *and* $g_j \in G$ *such that* $in_\succ(g_j) | f_i$ *and* $f \neq 0$ **do**
$f := f - \dfrac{f_i}{in_\succ(g_j)} g_j$
WHILE *there exists a term* f_k *in* f *such that* $in_\succ(g_j) | f_k$ *and* $f \neq 0$ **do**
$f := f - \dfrac{f_k}{in_\succ(g_j)} g_j$
Output $\overline{f}^G := f$.

We remark that, in the normal form algorithm during Conti-Traverso algorithm, the input polynomial f is a monomial, and thus intermediate polynomial f is also monomial since each $g_j \in G$ is binomial. And in Theorem 2.6, $l = 2$ since I_A is a binomial ideal, $L = 1$ since f is a monomial. Thus the worst case complexity of the number of reductions is $O(d^n \cdot \deg(f))$.

4 Experiments

To analyze the efficiency of Conti-Traverso algorithm if the Gröbner basis is already given, we measured the number of reductions during Conti-Traverso algorithm for acyclic tournament

graphs, acyclic directed graphs, and complete bipartite graphs. We examined Algorithm 3.9 to compare the number of canceling cycles during cycle canceling algorithm.

In the normal form algorithm Algorithm 3.9, we may have many choice of g_j in the first "WHILE" loops. We used four strategies to choose g_j. Let $\mathbf{c}$ be a cost vector and $g_i = \mathbf{x}^{\mathbf{a}_i} - \mathbf{x}^{\mathbf{b}_i} \in G$ ($\mathbf{x}^{\mathbf{a}_i} \succ \mathbf{x}^{\mathbf{b}_i}$).

(i) Most Leading Term Method Choose g_i such that $\mathbf{c} \cdot \mathbf{a}_i$ is the largest.

(ii) Most Improvement Method Choose g_i such that $\mathbf{c} \cdot \mathbf{a}_i - \mathbf{c} \cdot \mathbf{b}_i$ is the largest.

(iii) Minimum Mean Cycle Method Let M_i be the number of non-zero elements in $\mathbf{a}_i$ and $\mathbf{b}_i$. Then choose g_i such that $(\mathbf{c} \cdot \mathbf{a}_i - \mathbf{c} \cdot \mathbf{b}_i)/M_i$ is the largest.

(iv) Best Improvement Method Let N_i be the number of iterations of the first "WHILE" loop for g_i in Algorithm 3.9. Then choose g_i such that $N_i(\mathbf{c} \cdot \mathbf{a}_i - \mathbf{c} \cdot \mathbf{b}_i)$ is the largest.

(i) and (ii) are the methods described in [14]. (ii) and (iii) correspond to the most improvement method [1] and minimum mean cycle canceling method [9, 1], respectively, both are known as polynomial time algorithms for the minimum cost flow problems. (iv) is that we choose g_i such that the improvement after the first "WHILE" loop for g_i is the maximum.

The data types used are the following:

Acyclic tournament graphs: The data name $T[v]$ means the tournament graph with v vertices. $(4 \leq v \leq 8)$

Acyclic directed graphs: The data name $D[v]R[r]$ means that the data is generated from $T[v]$ by deleting each edge randomly by the probability $1 - r/100$. $(v = 7, 8,\ r = 50, 60, 70, 80, 90)$

Complete bipartite graphs: The data name $K[m][n]$ $(m \leq n)$ means the complete bipartite graph with the vertex sets V, W such that $|V| = m$, $|W| = n$. $((m, n) = (3, 3), (3, 4), (3, 5), (3, 6), (4, 4), (4, 5))$

For each graph, we give each edge the integer cost among 0 and 50 randomly, and give each edge the initial flow among 0 and 20 randomly. We examined each data 100 times and averaged the number of iterations of the first "WHILE" loop in Algorithm 3.9. We used G in Algorithm 3.9 as the reduced Gröbner basis with respect to the refinement of $\mathbf{c}$ and the universal Gröbner basis for toric ideal of each graph (which corresponds to the cycle canceling algorithm).

Comparing the results using reduced Gröbner bases, the strategies (ii) and (iii) which are efficient in the cycle canceling algorithm are also efficient for Conti-Traverso algorithm, and the strategy (iv) is not so much efficient than (ii) or (iii). Comparing the result using reduced Gröbner bases and that using universal Gröbner bases, Conti-Traverso algorithm is not so much efficient than cycle canceling algorithm. This shows that the number of reductions does not depend on the number of elements in Gröbner bases.

From the viewpoint of calculation time, the methods using reduced Gröbner bases seem to be more efficient. Calculating the reduced Gröbner bases takes large time, and calculating the universal Gröbner bases (enumerating all circuits in the graphs) also takes large time by Corollary 3.5. On the other hand, deciding the basis to reduce for the case of reduced Gröbner bases takes much shorter than for the case of universal Gröbner bases since the number of elements in reduced Gröbner basis is much smaller than that in universal Gröbner basis. In fact, for the data $T8$ average calculation time of most improve method using reduced Gröbner basis is 0.03 second while that using universal Gröbner basis is 168.65 seconds (these results are timed on Sun UltraSPARC-II, 450 MHz workstation with 2GB memory).

Data Name	Gröbner Basis	Reduction Strategy			
		(i)	(ii)	(iii)	(iv)
T4	Reduced	3.39	3.09	3.08	2.99
	Universal	3.29	2.73	2.74	2.71
T5	Reduced	8.11	7.10	7.09	6.71
	Universal	7.84	5.47	5.53	5.19
T6	Reduced	15.39	12.28	12.21	11.63
	Universal	18.15	8.95	9.27	8.03
T7	Reduced	28.70	20.16	20.19	18.79
	Universal	35.79	13.56	13.97	11.10
T8	Reduced	51.85	28.76	29.12	26.82
	Universal	62.67	18.73	19.40	14.12
D7R50	Reduced	6.14	5.09	5.11	5.13
	Universal	6.73	4.49	4.52	4.24
D7R60	Reduced	9.85	7.66	7.82	7.64
	Universal	11.44	6.38	6.52	5.92
D7R70	Reduced	12.10	9.86	9.87	9.63
	Universal	15.74	7.70	7.62	6.88
D7R80	Reduced	19.03	13.54	13.40	12.98
	Universal	22.55	10.14	10.38	8.91
D7R90	Reduced	21.98	16.02	15.98	15.31
	Universal	26.95	11.63	11.74	9.76

Data Name	Gröbner Basis	Reduction Strategy			
		(i)	(ii)	(iii)	(iv)
D8R50	Reduced	10.30	7.89	7.93	7.85
	Universal	11.41	6.50	6.57	6.01
D8R60	Reduced	16.50	11.72	11.98	11.47
	Universal	20.25	8.99	8.99	7.90
D8R70	Reduced	26.08	16.09	16.19	15.26
	Universal	29.42	11.83	11.91	9.74
D8R80	Reduced	32.16	19.96	19.57	18.87
	Universal	39.30	13.88	14.22	11.28
D8R90	Reduced	39.58	23.90	23.65	22.35
	Universal	48.74	16.23	16.37	12.68
K33	Reduced	5.22	3.79	3.82	3.89
	Universal	5.12	3.72	3.71	3.68
K34	Reduced	8.03	5.61	5.94	5.56
	Universal	9.04	5.34	5.75	5.10
K35	Reduced	12.63	7.71	7.71	7.53
	Universal	14.24	7.17	7.33	6.65
K36	Reduced	16.61	9.55	9.67	9.36
	Universal	17.84	9.07	9.25	8.17
K44	Reduced	15.04	8.98	8.93	8.94
	Universal	19.68	7.92	8.37	7.22
K45	Reduced	21.58	11.67	11.86	11.49
	Universal	27.74	10.84	11.27	9.25

5 Conclusions

In this paper, we have studied the reductions in Conti-Traverso algorithm and have experimented for the minimum cost flow problems and the transportation problems. The strategies which are used in cycle canceling algorithm are also efficient for Conti-Traverso algorithm. The calculation time is almost same as the cycle canceling algorithm.

Although the theoretical calculation time is not known, Conti-Traverso algorithm will be useful if the fast algorithms to calculate Gröbner bases for toric ideals are constructed.

Acknowledgement

The author thanks Hiroshi Imai and Fumihiko Takeuchi for useful comments and Ayumu Nagai, Hisashi Koga, and Hirotada Kobayashi for their help about implementing the algorithms.

References

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, New Jersey, 1993.

[2] A. Bachem and W. Kern. *Linear Programming Duality.* Springer-Verlag, Berlin, 1991.

[3] P. Conti and C. Traverso. Buchberger algorithm and integer programming. In *Proc. AAECC-9*, Springer, LNCS **539**(1991), pp. 130–139.

[4] D. A. Cox, J. B. Little and D. B. O'Shea. *Ideals, Varieties, and Algorithms.* Second Edition, Springer-Verlag, New York, 1996.

[5] D. A. Cox, J. B. Little and D. B. O'Shea. *Using Algebraic Geometry.* Springer-Verlag, New York, 1998.

[6] T. Dubé, B. Mishra and C. K. Yap. Admissible Orderings and Bounds for Gröbner Bases Normal Form Algorithm.

[7] J. A. de Loera, B. Sturmfels and R. R. Thomas. Gröbner bases and triangulations of the second hypersimplex. *Combinatorica*, **15**(1995), pp. 409–424.

[8] P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *Annals of Statistics*, **26**(1998), pp. 363–397.

[9] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *J. ACM.* **36**(1989), pp. 873–886.

[10] M. Hayer and W. Hochstättler. Test Sets for Vertex Cover Problems. In *Proc. 6th Twente Workshop on Graphs and Combinatorial Optimization*, Electronic Notes in Discrete Mathematics, **3**, 1999.

[11] T. Ishizeki and H. Imai. Complexity of Gröbner bases for toric ideals of acyclic tournament graphs. *RIMS Kokyuroku: Foundations of Computer Science*, **1148**(2000), pp. 134–139.

[12] B. Sturmfels. *Gröbner Bases and Convex Polytopes.* AMS University Lecture Series,**8**, Providence, RI, 1995.

[13] B. Sturmfels and R. R. Thomas. Variation of Cost Functions in Integer Programming. *Mathematical Programming*, **77**(1997), pp. 357–387.

[14] R. R. Thomas. A Geometric Buchberger Algorithm for Integer Programming. *Mathematics of Operations Research*, **20**(1995), pp. 864–884.