

グレブナー基底と並列計算

岩手大学・工学部 鈴木正幸

- 変数が多くて,
- 次数が高い,
- 方程式の根を求める (逐次) アルゴリズム

コンピュータがたくさんあった場合,

- 逐次アルゴリズムから,
- 独立して計算できる部分問題を取り出し,
- それぞれの計算機で同時に計算する.
- 部分問題をどのように渡し, 結果をどうもらうか
 - 並列化した事による通信のオーバーヘッド
 - 通信量と計算量の比 (粒度)
- コンピュータ数に対し, どれくらい速くなるのか
 - 問題自体の並列度
 - 台数にたいする速度向上は? (スケーラビリティ)

1 方程式を解くとは?

- 一次方程式, $ax = b$

$$x = a^{-1}b$$

- 連立一次方程式 (系)

$$\begin{aligned}(a_{11}x_1 + a_{12}x_2 + \dots &= c_1, \\ &\dots, \\ a_{n1}x_1 + a_{n2}x_2 + \dots &= c_n)\end{aligned}$$

- 線形代数, ガウスの消去法
- 一次方程式, $ax = b$ に帰着させる

- 一変数方程式, $a_nx^n + \dots + a_0 = b$

- 根の公式, $x^n = c$ に帰着させる.
- 帰着できない時, 数値計算 (ニュートン法) で近似的に求める.

- 多変数 (代数) 方程式 (系) ($f_1(x, \dots, z) = 0, \dots, f_n(x, \dots, z) = 0$)

- 変数消去, 因数分解
- 必ず解ける方法を知っていますか?

1.1 基底とは

問題

天秤秤と, a グラムの重りと b グラムの重りが無数にあるとします. どんな重さが測れるでしょう?

あるいは, c グラムを測る事ができますか?

- この問題は, 不定方程式 $ax + by = c$ を満たす, 整数 x, y を求める問題となる.
- この解は, a と b の最大公約数を求める問題に帰着されます.
- a と b の組合わせで作れる最小の数は, 最大公約数であり, その倍数しか a と b の組合わせでは作れません.
- 最大公約数は a と b の組合わせでできる数の集合の基底となります.
- 上の問題は, a と b の最大公約数を g とすると, c は g の倍数でなければ解が存在しないこと, $ax + by = g$ の x と y は, Euclid の互除法によって求められます.

問題

二つの方程式 $f_1(x) = 0, f_2(x) = 0$ の共通根は？

それぞれの方程式の根を求めて、共通な根を求めてもいいですが、

- 上の議論から、二つの式 $f_1(x), f_2(x)$ の組み合わせでできる、最も簡単な (次数の低い) 式 (基底) を求め、その根を求める。
- 基底は、 $f_1(x)$ と $f_2(x)$ の最大公約多項式 ($g(x)$) となり、

$$A(x)f_1(x) + B(x)f_2(x) = g(x),$$

$$\deg(A(x)) < \deg(f_2(x)),$$

$$\deg(B(x)) < \deg(f_1(x))$$

1.2 多変数方程式 をどう解くか？

$$(f_1(x, \dots, z) = 0, \dots, f_n(x, \dots, z) = 0)$$

に対し、 f_1, \dots, f_n を組み合わせでできる任意の多項式

$$A_1(x, \dots, z)f_1(x, \dots, z) + \dots + A_n(x, \dots, z)f_n(x, \dots, z)$$

の集合を考えます。

この集合を (f_1, \dots, f_n) と表し、 f_1 から f_n が作るイデアル \mathcal{I} と呼ぶ。

\mathcal{I} を作ることでできる多項式の組をイデアルの基底と呼びます。

方程式を解くのに都合の良い基底を求めることは、同じ根を持つ、より簡単な方程式系への変換となる。

この基底が例えば、

$$(g_1(x, z) = 0, g_2(y, z) = 0, \dots, g_m(z) = 0)$$

という形で求まれば、多変数方程式の問題は、一変数方程式の問題に帰着される。

「このような変形はできるのか」、「変形する方針は」、「必ず求まるのか」などが問題となる。

2 パズルと基底

グラス置き換えパズル ウィスキーのグラス W , ビールのグラス B , お酒のグラス S が一列に並んでいる.

グラスは次の置き換え規則で, 置き換えて良いとする.

$$\text{置き換え規則 } G \left\{ \begin{array}{l} B \longleftrightarrow WB \\ BS \longleftrightarrow W \end{array} \right.$$

問題

1. $BSBS$ は $WWWB$ に置き換えできるか?
2. $BSBBS$ は BWW に置き換えできるか?

問題の難しい点

- できる場合はその置き換えを示せば良いが,
- できない事を示す事.

パズル解法への道

- 簡単な方へ置き換える (簡約化) ことにする.

$$\text{簡約規則 } R \left\{ \begin{array}{l} WB \rightarrow B \\ BS \rightarrow W \end{array} \right.$$

- これ以上簡約できないもの (正規形)
- 置き換え規則 G で置き換え可能な列の要素は簡約規則 R で同じ正規系を持つか?
この性質が成り立てば, 簡約系で正規形が同じであれば, 置き換え系で, 置き換え可能となる.
- 置き換え可能なのに, 同じ正規形を持たない場合は, そのような簡約規則を追加すればよい.

例えば, WBS は二つの

$$\left\{ \begin{array}{l} WBS \rightarrow WW \\ WBS \rightarrow BS \rightarrow W \end{array} \right.$$

置き換え系では, WW と W は, WBS を通して置き換え可能であるから, 簡約系で

$$WW \rightarrow W$$

を新しい簡約規則として採用すればいい事になる.

この追加される簡約規則を同やってみ付けるかが問題となる.

- 簡約規則の左項中で, 重なりが生ずるような二つの規則を探す. (この二つの簡約規則を危険対と呼ぶ).

今の場合, BS と WB は 重なりを持つ項, WBS を別の正規形に簡約する可能性を持つ.

- この操作を次々に繰り返し, 危険対が全て同じ簡約形を持つようになった時, 置き換え可能である物は, 全て同じ正規形を持つ事になる.

簡約系の完備化という. 完備な系とは,

- 正規系は有限ステップで求まる. (停止性)
- ある項の正規系は, 簡約順序によらず同じになる.(合流性)

パズルの答え 簡約規則 R を完備化すると,

$$\text{簡約規則 } R' \left\{ \begin{array}{l} WB \rightarrow B \\ BS \rightarrow W \\ WW \rightarrow W \end{array} \right.$$

が得られる. これで, $BSBS \rightarrow^* W$, $WWWB \rightarrow^* B$, なので, 置き換え可能ではない.

$BSBBS \rightarrow^* BW$, $BWWW \rightarrow^* BW$, なので, 置き換え可能となる.

これがどう方程式と関係しているのでしょうか?

3 グレブナー基底

与えられた方程式 f_i の最高順位項を $head(f_i)$ 、残りの項を $rest(f_i)$ とすると,

$$f_i = head(g_i) + rest(g_i) = 0$$

から

$$head(g_i) \rightarrow -rest(g_i)$$

という簡約規則を作る事ができる.

このような簡約系を作るには, 項間の順序, 簡約, 危険対の求め方を, 方程式用に決める必要がある.

3.1 項の間の順序

いくつかの順序が考えられ, 順序によって完備な簡約系が異なる.

辞書式順序: $xyz > yz^3 > z^5$

全次数辞書式順序: $x^5 > x^4y > x^3yz$

3.2 簡約

基底の先頭項を残りの項で置き換える簡約規則と見て, 項をより低順位項で置き換える操作.

例 2.1: g_1 を g_2 で M 簡約

$$g_1 = x^4yz - xyz^2 \quad (\text{head}(g_1) = x^4yz, \text{rest}(g_1) = xyz^2)$$

$$g_2 = x^3yz - xz^2 \quad (\text{head}(g_2) = x^3yz, \text{rest}(g_2) = xz^2)$$

$$\begin{aligned} g' &= g_1 - (\text{head}(g_1)/\text{head}(g_2))g_2 \\ &= g_1 - (x^4yz/x^3yz)g_2 \\ &= x^2z^2 - xyz^2 \end{aligned}$$

3.3 S 多項式

新たな簡約規則を得るための計算.

2つの多項式 f_1, f_2 の S 多項式を $Sp(f_1, f_2)$ と書き、以下のように計算する。

$$Sp(f_1, f_2) = \frac{lcm}{\text{head}(f_1)}f_1 - \frac{lcm}{\text{head}(f_2)}f_2 \quad (1)$$

例 2.2: g_1 と g_2 の S 多項式

$$g_1 = x^3yz - xz^2, \quad \text{head}(g_1) = x^3yz$$

$$g_2 = x^2y^2 - z^2, \quad \text{head}(g_2) = x^2y^2$$

$$lcm(\text{head}(g_1), \text{head}(g_2)) = x^3y^2z$$

$$\begin{aligned} Sp(g_1, g_2) &= (lcm/\text{head}(g_1))g_1 - (lcm/\text{head}(g_2))g_2 \\ &= (x^3y^2z/x^3yz)g_1 - (x^3y^2z/x^2y^2)g_2 \\ &= -xyz^2 + xz^3 \end{aligned}$$

3.4 グレブナー基底の定義

イデアル \mathcal{I} の基底を $G = \{f_1, \dots, f_n\}$ とする。

F を可能な限り M 簡約した結果を F' とし,

$$F \xrightarrow{G} F'$$

と表す。

\mathcal{I} の任意の要素 f に対し,

$$f \xrightarrow{G} 0$$

という性質を持つとき, G をグレブナー基底と呼ぶ。

G がグレブナー基底の時, $f \xrightarrow{\psi} f'$ を計算し, $f' = 0$ を調べることで, $f \in \mathcal{I}$ であるかを簡単に決定できる。

例 2. 3: f_1, f_2, f_3 のグレブナー基底を求める。(全次数辞書式順序)

$$\left\{ \begin{array}{l} f_1 = 2x_1^3x_2 + 6x_1^3 - 2x_1^2 - x_1x_2 - 3x_1 - x_2 + 3 \\ f_2 = x_1^3x_2 + 3x_1^3 + x_1^2x_2 + 2x_1^2 \\ f_3 = 3x_1^2x_2 + 9x_1^2 + 2x_1x_2 + 5x_1 + x_2 - 3 \end{array} \right.$$

(s 多項式の例)

$$\begin{aligned} Sp(f_1, f_2) &= (lcm/head(f_1))f_1 - (lcm/head(f_1))f_2 \\ &= (2x_1^3x_2/2x_1^3x_2)f_1 - (2x_1^3x_2/x_1^3x_2)f_2 \\ &= -2x_1^2x_2 - 6x_1^2 - x_1x_2 - 3x_1 - x_2 + 3 = f'_4 \end{aligned}$$

(M 簡約の例)

$$\begin{aligned} f'_4 &\xrightarrow{f_3} f'_4 - (-2x_1^2x_2/head(f_3))f_3 \\ &= x_1x_2 + x_1 - x_2 + 3 \end{aligned}$$

< f_1, f_2, f_3 のグレブナー基底 >

$$G = [x_1x_2 + x_1 - x_2 + 3, 2x_1^2 - 3x_1 + 2x_2 - 6, 2x_2^2 - 8x_1 - 5x_2 - 3]$$

4 グレブナー基底から方程式の根を求める方法

辞書式順序で基底計算を行うと、連立方程式の解が求めやすいが、基底計算に時間がかかる上に計算量が多くなる。

簡単に求まる基底から、根を求める手法として固有値法がある。

1. 任意の多項式を、グレブナー基底 G で簡約した多項式の集合 $\mathcal{P}^s/\mathcal{I}$ は、ベクトル空間をなす。
2. グレブナー基底の最高順位項で割り切れない全ての項の集合を Normal set といい、 $\mathcal{P}^s/\mathcal{I}$ ベクトル空間の基底となる。
3. Normal set により $x_i \times$ を行列で表す事ができる。
4. その行列の固有値は、 \mathcal{I} の x_i に関する根となる。

例 3. 1: 例 2. 3 の f_1, f_2, f_3 の根を求める。

< f_1, f_2, f_3 のグレブナー基底 >

$$G = [x_1x_2 + x_1 - x_2 + 3, 2x_1^2 - 3x_1 + 2x_2 - 6, 2x_2^2 - 8x_1 - 5x_2 - 3]$$

$$Normal Set = \{1, x_2, x_1\}$$

< 書き換え規則 >

$$\left\{ \begin{array}{lcl} x_1x_2 & \rightarrow & -x_1 + x_2 - 3 \\ x_1^2 & \rightarrow & \frac{3}{2}x_1 - x_2 + 3 \\ x_2^2 & \rightarrow & 4x_1 + \frac{5}{2}x_2 + \frac{3}{2} \end{array} \right.$$

$$P = c_1\vec{x}_1 + c_2\vec{x}_2 + c_3$$

< $x_1 \times$ の行列 >

$$\begin{array}{c} 1 \quad x_2 \quad x_1 \\ \begin{array}{l} 1 \\ x_2 \\ x_1 \end{array} \begin{pmatrix} 0 & 0 & 1 \\ -3 & 1 & -1 \\ 3 & -1 & 3/2 \end{pmatrix} \end{array}$$

< $x_2 \times$ の行列 >

$$\begin{matrix} & 1 & x_2 & x_1 \\ \begin{matrix} 1 \\ x_2 \\ x_1 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 \\ 3/2 & 5/2 & 4 \\ -3 & 1 & -1 \end{pmatrix} \end{matrix}$$

< x_1 の固有値 >

$$\left[0, \frac{5}{4} + \frac{1}{4}\sqrt{65}, \frac{5}{4} - \frac{1}{4}\sqrt{65} \right]$$

< x_2 の固有値 >

$$\left[3, -\frac{3}{4} + \frac{1}{4}\sqrt{65}, -\frac{3}{4} - \frac{1}{4}\sqrt{65} \right]$$

これらの固有値が f_1, f_2, f_3 の根である。

5 Buchberger 算法と並列化

以下に, f_1, \dots, f_l が作るイデアルの Gröbner 基底を計算する Buchberger 算法を示す.

Buchberger 算法

Input: $F = \{f_1, \dots, f_l\}$

Output: Gröbner 基底 G of $Ideal(F)$

$PairQ \leftarrow \phi;$

$G \leftarrow \phi;$

foreach ($f_i \in F$) {

$PairQ \leftarrow UpdatePairQ(PairQ, f_i, F);$

$G \leftarrow UpdateBase(G, f_i);$

}

while ($PairQ \neq \phi$) {

$(g_i, g_j) \leftarrow \text{select an element of } PairQ;$

$PairQ \leftarrow PairQ \setminus \{(g_i, g_j)\};$

$g_k \leftarrow SPOL(g_i, g_j) \downarrow_G;$

if $g_k \neq 0$ {

```

    PairQ ← UpdateQ(PairQ, gk, G);
    G ← UpdateBase(G, gk);
  }
}

```

算法の概要と戦術

- G は中間的な基底の集合, $PairQ$ は新たな基底を構成可能な中間基底の組 (ペア) の集合, を表している.
- $PairQ$ から一つのペア (g_i, g_j) を選ぶ. この選び方を選択戦術と呼ぶ.
- $SPO L(g_i, g_j)$ の現在の中間基底での正規形 g_k を求める. 簡約基底の選び方の順序や簡約法を簡約化戦術と呼ぶ.
- g_k が 0 でなければ,
 - ペア削除戦術により, 新たなペアの生成と, 不必要なペアの削除をおこない ($UpdateQ$),
 - 中間基底に追加し, 基底削除戦術により不必要な中間基底の削除をおこなう ($UpdateBase$),
- $PairQ$ が空になった時点で算法は停止し, G に Gröbner 基底が求まる.

5.1 Buchberger 算法の並列性

Buchberger 算法の計算上の問題点は, ペアの個数の組み合わせ的な膨張と, 中間基底の数係数の膨張である. ペアの個数の膨張を防ぐために, いくつかの選択戦術が考えられており, 選択戦術を保持したまま, ペアの個数に関する並列性の導入が必要となる [1].

野呂ら [6] は, 数係数の膨張による計算時間の増大を, 並列計算により減らせることを示した. 筆者 [8] は, 共有メモリを用いて更に高速化を行った. 一つの基底による S 多項式の簡約 ($SPO L(g_i, g_j) \downarrow_{\{g_k\}}$) を $SPO L(g_i, g_j)$ や g_k を分割し, 並列計算する. これを一簡約並列と呼ぶ. この方式では,

- 全ての戦術を保持したまま並列計算が可能であるが,
- 細粒度の並列化であり, 有効となるのは数係数が大きくなった場合に限る,
- 逐次部分が残る.

この方式は, 大規模な Gröbner 基底計算 [7] において, 並列度が中規模 (≤ 20) 程度であれば良い性能を示している [6, 8]. しかし計算の逐次部分, 通信コストのために, 性能限界を持つ.

[1] では、選択戦術を忠実に守りつつ、ペアに関する簡約 ($\text{SPOL}(g_i, g_j) \downarrow_G$) を並列に行っている。 G を共有し、複数のワーカが別々の簡約を行う。以後、この並列化をペア並列と呼ぶ。ペア並列では、

- 逐次部分がないが、
- 中間基底の生成順序を保つため、 S 多項式の生成、簡約化に待ちが生ずる、
- 無駄な計算 (0 簡約される基底を用いたペア) が生ずる。

この方式では、中間基底の生成順序による待ちがボトルネックとなり、様々な問題に対して性能限界が生じることが報告されている。この論文中、斉次な基底計算の場合、生成順序による待ちが大幅に減らせ、高い並列性能を示すことが言及されているが、その性能は示されていない。

6 並列算法の組み合わせによる並列度の向上

前章の二つの並列化算法はそれぞれ性能限界を持つ。しかし、その限界を持つ原因は異なるので、二つを組み合わせることにより、並列性能向上が期待できる。

提案する算法の基本的な考え方は、

- ペア並列度を検出し、
- ペア並列度が低い場合に、一簡約並列を行う

であるが、ペア並列度の検出は計算中には行えない。そこで、まず同じ戦術の modular 計算を行い、0 簡約される基底、基底の生成順序と簡約依存性をあらかじめ求める。この手法は、[3] で用いられていて、ペア並列度は低いことが報告されている。つまり、ペア並列度だけでは高い性能向上は見込めない。そこで、

- modular 計算により基底の生成順序と簡約依存性をあらかじめ求め、並列計算可能なブロックに分ける。(これを並列計算のシナリオと呼ぶ)
- シナリオにより、ブロック内をペア並列実行するが、並列度が投入できるプロセッサ台数より小さい場合、全プロセッサが計算に参加できるように、一簡約並列を併用する。

7 d-Gröbner 基底によるペア並列度の向上

選択戦術として斉次化あるいは sugar を用いる場合には、あらかじめ決めることができるペア並列度が存在する。

7.1 d-グレブナー基底

定義 1

S 多項式の全次数 (または *sugar* 次数) d で打ち切った Buchberger 算法の結果を G_d とする .
この G_d のことを d -グレブナー基底という .

定理 2

斉次多項式 f_1, \dots, f_n に対する d -グレブナー基底は以下の性質を持つ :

1. $\deg(f) < d$ な f に対し, $\xrightarrow{G_d}^* 0$ が定義される .
2. $\forall p \in \mathcal{I} \quad \deg(p) \leq d \Rightarrow p \xrightarrow{G_d}^* 0$
3. $\forall f, g \in G_d \quad \deg(\text{HT}(f), \text{HT}(g)) \leq d$ に対し , $\text{SPOL}(f, g) \xrightarrow{G_d}^* 0$

$\forall d > d_\infty \quad G_d = G_{d_\infty}$ となる d_∞ が存在する .

系 3

任意の多項式に対し , 定理 2 の \deg を \deg_S で置き換えて , 性質 1, 2, 3 および d_{infty} の存在が成り立つ .

定理より d -グレブナー基底は ,

$$G_0 \rightarrow G_1 \rightarrow \dots \rightarrow G_d \rightarrow G_{d+1} \rightarrow \dots \rightarrow G_{d_\infty} = \dots$$

のように計算でき , $G_d = G_{d-1} + \{d\text{-次式}\}$ となる .

7.2 d-グレブナー基底の並列性

前節の定理より , G_{d-1} が求まっていて , G_d を求める場合は , 次の事が言える .

1. G_d に追加される基底は , $\text{SPOL}(g_i, g_j)$, $g_i, g_j \in G_{d-1}$, より作られ , 基底候補の S 多項式に依存性はない .
2. $\text{SPOL}(g_i, g_j) \downarrow_{G_{d-1}}$ の計算にも依存性はない .
3. 上の計算後 , $\text{SPOL}(g_i, g_j) \downarrow_{G_d}$ の計算は , 1, 2 で作られた d -次基底のみの相互簡約で求められる .

つまり , S 多項式の並列生成 , G_{d-1} に関する並列簡約 , が可能である . d -次基底の相互簡約には基底間の依存性が存在するが , これは一簡約並列実行可能である .

sugar 値	基底数	時間
11	14	21.22
12	24	89.32
13	37	359.4
14	63	2962
15	101	84620
16	168	572100
17	1	28900
18	1	12800
20	1	30000
total	442	731800

表 1: McKay 計算の sugar 毎の基底数と実行時間 (秒)

8 実装と性能 (予測)

前章により, 斉次あるいは sugar を用いた並列算法は,

- modular 計算によりシナリオを作成し,
- d -の S 多項式 s_i を並列生成し,
- $s_i \downarrow_{G_{d-1}}$ を並列計算する. ペア並列度が足りない場合に, 一簡約並列を併用する.
- $s_i \downarrow_{G_{d-1}}$ 同士の相互簡約を一簡約並列計算する.

となる. asir 上で逐次版の d -グレブナー基底計算を実装し, その実行過程を検討し, 並列版を現在実装中である.

表 1 に, McKay[7] 問題に対し, 選択戦術として sugar 戦術をもちいて実行した結果をしめす. 8 台の場合の一簡約並列性能は, 5.6, ほぼ 7 割である. 表中の基底数が, シナリオを用いて計算した場合のペアの並列度になる. 計算時間のもっともかかる, sugar 値 15, 16 辺りのペア並列度はかなり大きい. sugar 値 17 以上では, ペアの並列度は 1 で, ペア並列だけでは十分な性能向上ははかれないことがわかる.

表 2 に, 同じ問題の modular 基底を, d -Gröbner 基底算法を用いて計算した結果を, asir の `gr_mod_main`, F_4 の結果とともに示す. 括弧内は g.c. 時間である. この計算は並列化のシナリオを作成する部分に相当する. まだ asir F_4 の性能には及ばないが, `gr_mode_main` に比べて数割早くなっていることがわかる.

表 3 に, d -グレブナー基底計算中の各 S 多項式の G_{d-1} に関する簡約時間, d -次の基底間の相互簡約にかかる時間を示す. G_{d-1} に関する簡約時間が支配的であり, 並列化した場合, ペア並列度が実行時間に大きく寄与することがわかる.

d -Gröbner	gr_mod_main	asir F_4
180 (409)	240 ()	126 (432)

表 2: McKay 計算の modular 計算時間 (秒) 比較

sugar 次数 d	全実行時間		SPOL(\cdot) $\downarrow_{G_{d-1}}$		$\downarrow_{G_d \setminus G_{d-1}}$	
total	180.7	(409.3)	148.1	(321.2)	32.2	(87.3)
11	0.8	(3.3)	0.7	(3.1)	0.1	(0.2)
12	2.5	(9.6)	2.2	(8.4)	0.3	(1.2)
13	7.6	(27.7)	6.8	(24.5)	0.8	(3.2)
14	19.4	(58.7)	16.6	(49.4)	2.7	(9.1)
15	48.7	(134.4)	39.6	(104.3)	9.0	(29.8)
16	74.6	(150.3)	54.9	(106.2)	19.4	(43.6)
17	25.2	(22.7)	25.2	(22.7)	0.0	(0.0)
18	1.0	(0.9)	1.0	(0.9)	0.0	(0.0)
19	0.1	(0.0)	0.1	(0.0)	0.0	(0.0)
20	0.3	(0.2)	0.3	(0.2)	0.0	(0.0)
21	0.4	(0.3)	0.4	(0.3)	0.0	(0.0)

表 3: d -Gröbner 基底計算時間 (秒) の内訳

一簡約並列算法 (共有メモリ版) の性能は, 12 のプロセッサで 8 程度の並列性能を得ている [8]. d -グレブナー基底計算の並列版は実装中であるので, 算法の組合わせによる全体性能を示すことはできないが, 相互簡約の部分の並列化, ペア並列性の低い部分, が高速化でき, 良い性能が得られることは明らかだろう.

参 考 文 献

- [1] Attardi, G., Tracerso, C.: Strategy–Accurate Parallel Buchberger Algorithms, *J.Symb. Comp.*, 21/4-6 (1997), 411–426
- [2] Beker, T., Weispfenning, V.: Gröbner Bases. GTM bf 141, Springer-Verlag, 1993
- [3] Faugère, J.C.: Parallelization of Gröbner basis *Proc. PASCO'94*, 1994, 124–132
- [4] Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F_4), *Journal of Pure and Applied Algebra* 139(1–3), 1999, 61–88
- [5] Giovini, A., Mora, T., Niesi, G., Robbiano, L., Traverso, C.: “One sugar cube, please” OR Slection strategies in the Buchberger algorithm, *Proc. ISSAC'91*, 1991, 49–54
- [6] Noro, M., Kando, T., Takeshima, T.: Solving a large scale problem by parallel algebraic computation on AP3000, *Research Report ISIS-RR-97*, FUJITSU LABS, 1997
- [7] Noro, M., Mckay, J.: Computation of replicable functions on Risa/Asir, *Proc. PASCO'97*, ACM Press, 1997, 130–138
- [8] 鈴木正幸: 分散共有メモリを用いた並列 Gröbner 基底計算の性能評価, 第 8 回数式処理大会, 1999