数理解析研究所講究録
第1568巻 2007年 55-59

Comprehensiveグレブナー基底の分散計算について

井上　秀太郎
SHUTARO INOUE
東京理科大学大学院理学研究科数学専攻
DEPARTMENT OF MATHEMATICS,GRADUATE SCHOOL OF SCIENCE,TOKYO UNIVERSITY OF SCIENCE*

河元　義文
YOSHIFUMI KOMOTO
東京理科大学大学院理学研究科数学専攻
DEPARTMENT OF MATHEMATICS,GRADUATE SCHOOL OF SCIENCE,TOKYO UNIVERSITY OF SCIENCE

佐藤　洋祐
YOSUKE SATO
東京理科大学理学部数理情報科学科
DEPARTMENT OF MATHEMATICAL INFORMATION SCIENCE,TOKYO UNIVERSITY OF SCIENCE†

1　はじめに

パラメーターの空間を有限個に分割し、それぞれの分割において一様なグレブナ基底になっているものを包括的グレブナ基底系 (Comprehensive Gröbner System, 以下 CGS と記す) とよぶ.CGS の計算アルゴリズムには主に次の 3 つが挙げられる.

- Weispfenning のアルゴリズム [5]
- Montes のアルゴリズム [1]
- Suzuki,Sato のアルゴリズム [3]

そのなかで, 最近発表された Suzuki,Sato のアルゴリズムは, 体 K 上の多項式環におけるグレブナ基底の計算で構成されているので、高速計算が可能である. またプログラムが非常にシンプルであり、分散計算が容易という長所も持っている.
本研究では Suzuki,Sato のアルゴリズムの分散計算に関する研究をおこなった. 実験結果よりある程度の結果は得ることに成功したが, いくつかの問題点も明らかになった.

*j1106701@ed.kagu.tus.ac.jp
†ysato@rs.kagu.tus.ac.jp

2 分散計算アルゴリズム

CGS 分散計算アルゴリズムの元になった Suzuki,Sato のアルゴリズムは以下の通りである.

Algorithm : CGSMain

Input:F : a finite subset of $K[\bar{X},\bar{A}]$
Output:$\mathcal{H}$: a finite set of pairs(h,G) of a polynomial and a Gröbner basis in $K[\bar{X},\bar{A}]$
begin
$G :=$ ReducedGöbnerBasis$(F, <_{\bar{X},\bar{A}})$;
if $1 \in G$ then
  $\mathcal{H}:=\{(1,F)\}$;
else
  $\{h_1,\cdots,h_l\}:=\{hc_{\bar{A}}(g) : g \in G \backslash K[\bar{A}]\}$;
  $h := lcm\ \{h_1,\cdots,h_l\}$;
  $\mathcal{H} := \{(H,g)\} \cup$
    CGSMain$(G \cup \{h_1\}) \cup \cdots \cup$CGSMain$(G \cup \{h_l\})$;
end if
return $\mathcal{H}$;
end.

このアルゴリズムは始めに体 K 上の多項式環におけるグレブナ基底を計算し, その結果から適当なパラメータによる多項式を集め, それらを元々与えられた多項式の集合に加え, 再度グレブナ基底を計算する再帰アルゴリズムである. 本研究での分散計算アルゴリズムはこのアルゴリズムの再帰的な部分をそれぞれのコンピュータで計算させている. よってメインコンピュータは最初のグレブナ基底の計算を行った後は, 命令と結果回収が役割となる.
計算実験に使用したプログラムは, 公開されてる (http://kurt.scitec.kobe-u.ac.jp/~sakira/CGBusingGB/) プログラムを改造したものであり,Risa/Asir 上で実装を行った.

3 計算時間の比較

計算実験には OS: knoppix/math2006 v4.0.2,CPU: Intel PentiumM 1.7GHz,Memory: 1.25GByte のコンピュータを使用し,1 台で計算した場合と 4 台で分散計算した場合,7 台で分散計算した場合の 3 つを比較した. また使用した例題は次の問題である. ただし順序は $X > X_2 > Y > Y_2 > Z > S$ の辞書式順序とし、A,B,C,D をパラメータとする.

例 1 $\{X^3-A, Y^4-B, X+Y-Z\}$.

例 2 $\{X^4-A, Y^5-B, X+Y-Z\}$.

例 3 $\{F, G, (X-X_2)^2+(Y-Y_2)^2-S, \partial F/\partial X \cdot \partial G/\partial Y_2 - \partial F/\partial Y \cdot \partial G/\partial X_2,$
$\partial F/\partial X \cdot (Y-Y_2) - \partial F/\partial Y \cdot (X-X_2)\}$. (ただし $F = AX^2+BY, G = CY_2^2+DX_2$ とする.)

例 4 $\{F, G, (X-X_2)^2+(Y-Y_2)^2-S, \partial F/\partial X \cdot \partial G/\partial Y_2 - \partial F/\partial Y \cdot \partial G/\partial X_2,$
$\partial F/\partial X \cdot (Y-Y_2) - \partial F/\partial Y \cdot (X-X_2)\}$. (ただし $F = X^2+Y^2+A, G = Y_2+BX_2^2+C$ とする.)

例 5 $\{F-Z, X^2+Y^2+Z^2-S, X+\partial F/\partial X \cdot Z, Y+\partial F/\partial Y \cdot Z\}$.
(ただし $F = (X-A)^2+BY^2+B$ とする.)

例 6 $\{F-Z, X^2+Y^2+Z^2-S, X+\partial F/\partial X \cdot Z, Y+\partial F/\partial Y \cdot Z\}$.
(ただし $F=(X-A)^2+AY^2+B$ とする.)

この実験から次の結果が得られた.

計算時間の比較 (単位：sec)

	1 台で計算した場合	4 台で分散計算した場合	7 台で分散計算した場合
例 1	0.02	1.571	2.544
例 2	15.01	6.013	6.012
例 3	9.874	8.063	8.58
例 4	42.93	37.95	36.23
例 5	57.94	30.58	28.72
例 6	215.3	179.9	161.7

例 1 のような一瞬で計算できる問題は通信の負担が増えてしまい逆効果であったが, それ以外の問題では多少の分散計算効果を得ることができた. しかし期待するほどの高速化には至らず、コンピュータの台数を増やしても効果は見られなかった. これはそれぞれのコンピュータへの負担が均等ではなく, 負担の重いコンピュータの処理を待っている時間が加わるためである.

4 分散計算アルゴリズムの問題点

今回の計算実験では主に 2 つの問題点が発見された.

- それぞれのコンピュータに計算させるグレブナ基底の計算回数が偏っている.
- 極端に時間の掛かるグレブナ基底の計算が存在する.

1 番目の問題点を分かり易く説明するために例題を 2 つ与える.

例 7 $\{axt^2+bytz-x(x^2+ay^2+bz^2), ayt^2+bzxt-x(y^2+az^2+bx^2), azt^2+bxyt-x(z^2+ax^2+by^2)\}$
(ただし a, b パラメータとする.)

例 8 $\{axt^2+bytz-x(x^2+cy^2+dz^2), ayt^2+bzxt-x(y^2+cz^2+dx^2), azt^2+bxyt-x(z^2+cx^2+dy^2)\}$
(ただし a, b, c, d パラメータとする.)

次の表は,CGS を計算する過程でそれぞれのコンピュータに計算させるグレブナ基底の計算回数をまとめたものである. なお, 表中の h とは 2 節のアルゴリズム CGSMain の $h_1, \cdots, h_l$ を意味する. また 100 万以上とは最後まで計算できなかった部分である.

例7

h	グレブナ基底の計算回数	h	グレブナ基底の計算回数
b	4	a^2-b	9
a	4	$a-b^2$	9
$b+1$	5	$a^3-2ab+1$	9
$a+b+1$	5	$3ba-b^3-1$	11
b^2-b^3-1	7	$b^2a^2-3ba+b^3+1$	90
$2ba-b^3-1$	7	$a^2+(-b-1)a+b^2-b+1$	93
$ba-1$	9		

例8

h	グレブナ基底の計算回数	h	グレブナ基底の計算回数
$c+d+1$	35	d^2-d+1	133
c	39	$c^3-2dc+1$	238
d	39	$2dc-d^3-1$	308
$d+1$	42	$dc+d^3+1$	421
$c-d^2$	67	$c^2+(-d-1)c+d^2-d+1$	846
c^2-d	68	b	100万以上
a	114	$dc-1$	100万以上

この表から判るように, 今回の分散計算アルゴリズムではコンピュータへの負担が非常に偏り, 特にパラメータが増えるとその特徴が顕著になる. 当然ではあるが, 一部のコンピュータだけが計算しているという状況は非効率的である. このような問題を解決するために, 1つ1つのグレブナ基底の計算をそれぞれのコンピュータに計算させるように改良することが考えられる. これは1回の命令でグレブナ基底の計算を1回させるという意味である. この方法ならばグレブナ基底の計算回数という一面においては効果的である. しかし命令回数の増加による通信面の負担を考慮する必要はある. なぜならばパラメータの数が少なくても, グレブナ基底の計算は万単位という場合は珍しいことではなく, その数だけ計算命令を行うことになるからである.

次に2番目の問題点について説明する.Suzuki,Sato のアルゴリズムはグレブナ基底の計算を何度も行うことで CGS を計算する. 実験により, この膨大な数のグレブナ基底の計算の中には極端に時間の掛かるものが存在することが分かった. このような極端なグレブナ基底の計算は1つのコンピュータに負担を与えることになる. しかし極端なケースはごく一部であり, それら一部のために全体に対策を施せば全体的には逆効果になることが十分にありうる. そこでこの問題を解決するために重要なのが AND 並列と OR 並列である. 現在の分散計算アルゴリズムでは AND 並列を採用し, 計算結果の全てを回収している. この方法では上で述べたような問題に対処することはできない. そこで今後の方針は, プログラムは複雑になるが,OR 並列を採用して効果を得ていくことが考えられる. 1回のグレブナ基底の計算に複数のコンピュータを使うことはデメリットになるが, 速く計算できた結果だけを回収するので全体的に高速になる可能性はある. 勿論,OR 並列といってもさまざまであるが, 今のところ最も効果が得られそうなのは"gr"と"hgr"による OR 並列である. 今回の計算実験で, グレブナ基底の計算には斉次化による計算方法を採用した. なぜならば大抵のグレブナ基底の計算において斉次化は非常に有効であり, 分散計算においても問題はないと考えたからである. しかし計算実験によって次のような, 極端に負担の重い場合が発見された.

例 9

$hgr([a^2-3*b^2*a+b^4+b, b^2*a^2-3*b*a+b^3+1, -x^3+(-a*y^2-b*z^2+a*t^2)*x+b*t*z*y, -b*x^3+(-y^2-a*z^2+b*t*z)*x+a*t^2*y, -a*x^3+(-b*y^2+b*t*y-z^2)*x+a*t^2*z], [x,y,z,t,a,b], [[0,4],[0,2]]);$
$219.5sec + gc : 76.01sec(336.7sec)$
$gr([a^2-3*b^2*a+b^4+b, b^2*a^2-3*b*a+b^3+1, -x^3+(-a*y^2-b*z^2+a*t^2)*x+b*t*z*y, -b*x^3+(-y^2-a*z^2+b*t*z)*x+a*t^2*y, -a*x^3+(-b*y^2+b*t*y-z^2)*x+a*t^2*z], [x,y,z,t,a,b], [[0,4],[0,2]]);$
$1.763sec + gc : 0.5808sec(2.353sec)$

例 10

$hgr([a+b^2, c^3+(d-1)*c^2-3*d*c+3, d^2-d+1, -x^3+(-c*y^2-d*z^2+a*t^2)*x+b*t*z*y, -d*x^3+(-y^2-c*z^2+b*t*z)*x+a*t^2*y, -c*x^3+(-d*y^2+b*t*y-z^2)*x+a*t^2*z], [x,y,z,t,a,b,c,d], [[0,4],[0,4]]);$
$107sec + gc : 12.9sec(122.1sec)$
$gr([a+b^2, c^3+(d-1)*c^2-3*d*c+3, d^2-d+1, -x^3+(-c*y^2-d*z^2+a*t^2)*x+b*t*z*y, -d*x^3+(-y^2-c*z^2+b*t*z)*x+a*t^2*y, -c*x^3+(-d*y^2+b*t*y-z^2)*x+a*t^2*z], [x,y,z,t,a,b,c,d], [[0,4],[0,4]]);$
$0.661sec + gc : 0.08012sec(0.741sec)$

上の例のようにhgrによる計算が逆に負担になるケースは決して多くはない. しかし膨大な数のグレブナ基底の計算において非常に問題となる. よって"gr"と"hgr"によるOR並列は有効であると思われる.

5 まとめ

今回の研究では、Suzuki,Satoのアルゴリズムによる分散計算の基礎を作ったに過ぎない. パラメータが多い場合のCGS計算は非常に規模が大きいことから, 今後はパラメータが2つや3つといった比較的にパラメータの少ない場合に絞り, より効率的なアルゴリズムを模索していきたい. 当面は斉次化によるグレブナ基底の計算と通常のグレブナ基底の計算によるOR並列による分散計算のプログラムを実装し, 計算実験を行う予定である.

[1] Montes, A.(2005).Improving DISPGB Algorithm Using the Discriminant Ideal, J. Symb. Comp., A3L 2005 special issue,to appear.

[2] Suzuki, A. and Sato, Y.(2003).An Alternative approach to Comprehensive Gröbner Bases. J. Symb. Comp. 36/3-4,649-667.

[3] Suzuki, A. and Sato, Y.(2006).A Simple Algorithm to compute Comprehensive Gröbner Bases using Gröbner Bases. to appear in International Symposium on Symbolic and Algebraic Computation (ISSAC 2006),Proceedings.

[4] Weispfenning, V.(1989).Gröbner bases in polynomial ideals over commutative regular rings, EUROCAL'87,J.H.Davenport Ed.,Springer LNCS 378,336-347.

[5] Weispfenning, V.(1992).Comprehensive Gröbner Bases, J. Symb. Comp.14/1,1-29.

[6] Weispfenning, V.(2003).Canonical Comprehensive Gröbner Bases, J. Symb. Comp. 36,669-683.