数理解析研究所講究録
1514巻 2006年 33-38

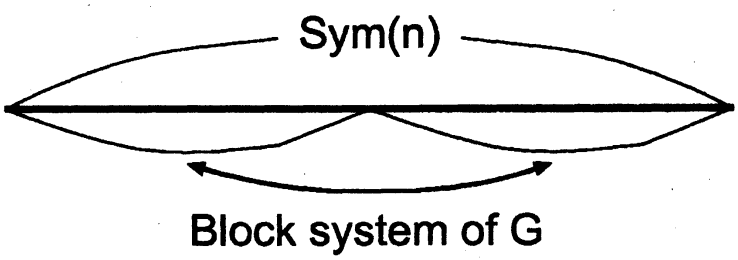
GAPの関数Normalizerの改良

宮本 泉
IZUMI MIYAMOTO
山梨大学
UNIVERSITY OF YAMANASHI

集合 $\Omega = \{1, 2, \cdots, n\}$ とし, G, H は Ω 上の置換群. G の H における正規化群 Normalizer は次で定義される.

$$\mathrm{Normalizer}(H, G) = \{\ h \in H | h^{-1}Gh = G\ \}.$$

H が 対称群 $Sym(n)$ =SymmetricGroup(Ω) と交代群のとき GAP4 - Groups, Algorithms, Programming (version 4)- a System for Computational Discrete Algebra は正規化群計算に特別な処理をしている. G が Ω 上 transitive(可移) でなかったり, 下の図で示す例のように primitive(原始的) でないとき, 正規化群を対称群 $H = Sym(n)$ より小さな群の中で計算している.



例: G が上の様な block system を一つしかもたないならば, レス積 W =WreathProduct($Sym(n/2), Sym(2)$) とおくと,

$$\mathrm{Normalizer}(Sym(n), G) \subseteq W$$

が成り立つので, GAP4 は Normalizer(W, G) を計算している.

G が原始的であっても, 2重可移でなければ, 下のように正規化群が含まれる群を制限できる.

Proposition.(AC1997[2])
G が可移ならば G の正規化群は G の $\Omega \times \Omega$ 上の orbits) によって作られるアソシエーションスキームの自己同型群に含まれる.

さらに, G が可移ならば次の Lemma を使うこともできる. もし, 可移でないときは, orbit ごとに使う様にする.

Lemma.(ISSAC2000[4, 5])
Let K be a permutation group on Ω. Let F be a tuple $[p_1, p_2, \cdots, p_r]$ of points in Ω and let G^i be the stabilizer of the subset $[p_1, p_2, \cdots, p_i]$ of F as a tuple in G for $i = 1, 2, \cdots, r$. Let I^i be the group of isomorphisms of the system of association schemes of G^i on $\Omega \backslash [p_1, p_2, \cdots, p_i]$. Set $I^0 = I$, $G^0 = G$ and

set $I^{\{0..i\}} = I^0 \cap I^1 \cap \cdots \cap I^i$. Suppose that $G^i \cap K$ is transitive on the orbit of $I^{\{0..i\}} \cap K$ containing the point p_{i+1} for $i = 0, 1, \cdots, r-1$. Then the normalizer of G in K is generated by $G \cap K$ and the normalizer of G in $I^{\{0..r\}} \cap K$.

Lemma の使い方の例：
Normalizer$(Sym(n), G) \subseteq W$ となるとき，下の固定部分群の列に対して，Lemma が成立するとする．

$$\begin{array}{cccc} W \supset & W_1 \supset & W_{1,2} \supset & W_{1,2,3} \\ G \supset & G_1 \supset & G_{1,2} \supset & G_{1,2,3} \end{array}$$

$$\text{Normalizer}(W, G) = \langle \text{Normalizer}(W_{1,2,3}, G), G \rangle$$

このとき，

$$\begin{aligned} \text{Normalizer}(W_{1,2,3}, G) &\subseteq \text{Normalizer}(W_{1,2,3}, G_1) \\ &\subseteq \text{Normalizer}(W_{1,2,3}, G_{1,2}) \\ &\subseteq \text{Normalizer}(W_{1,2,3}, G_{1,2,3}) \end{aligned}$$

が成立し，したがって，$N_{i,j} = \text{Normalizer}(W_{1,\cdots,i}, G_{1,\cdots,j})$ とおくと，

$$\text{Normalizer}(W, G) = \langle \text{Normalizer}(N_{i,j}, G), G \rangle, \quad (i \geq j)$$

が成立する．一般的には，小さい群の正規化群計算は簡単であると期待できるが，個別の群に対して，このことはまったく成り立たないことが実験でわかる．しかも，その差が 100 倍以上になることもめずらしくない．どのような群がそうなるかは，現状では i よく判らない．しかし，実験結果の傾向から heuristic を加えて今回のプログラムを作成した．

今回の実験で比較した 4 つのプログラム

- GAP4 — Normalizer in $Sym(n)$ and $Alt(n)$.
- ISSAC00 — 前ページの方法を使ったプログラム.
- AC05 — 今回のプログラムの 11 月バージョン (AC2005[3] で発表).
- CA-ALIAS05 — 今回のプログラム,
  GAP4 の Normalizer に 50 行ほど追加,
  アソシエーションスキームを使わない,
  Lemma には heuristic が伴う.

今回のプログラムの目標と動機，および，現状.

ISSAC00 のプログラムは複雑過ぎる.

- アソシエーションスキームの自己同型群計算にはバックトラック法を使っている.
  正規化群計算でも使っている.
  自己同型群を計算して，その中でまた，正規化群を計算するのは回り道だろう.
- かえって計算時間がかかる場合がある.

• 簡単なプログラムで，ISSAC00 ほど速くなくても，いつでも GAP4 の Normalizer より速くしたい.

⟶ 一般的なアソシエーションスキームは使わないことにする.

GAP4 と同じように, レス積の群のみを利用する.

レス積の群は典型的な非原始的置換群の自己同型群になっている.

Lemma に heuristic を使わないと十分速くならない場合がある.

しかし，heuristic を使うと，返って遅くなる場合がでてくる.

結局,heuristic を使ったため, 平均的には大変速く計算できるが, 逆に, 非常に遅くなる場合もできた.

• 計算機の性能が良くなっているので，次数の大きな置換群で計算したい.

⟶ GAP4,ISSAC00 共に, 次数の小さな群でも困難があることが分った.

それは,GAP の可移な群の library は 2000 年 22 次までが 現在 30 次まで増えたことによる.

多くの実験の結果, 困難の原因は，簡単な構造の群の正規化群計算にあるらしいが, まだ, 未解決である.

次数の大きな場合は保留することにした.

実験データ

次数 20 から 30 までの合計 36620 個の可移な置換群，これらの群 G に対して，Normalizer$(Sym(n), G)$, (n は G の次数) を計算した.

次数ごとの可移な群の個数 (GAP library より)

次数	個数	次数	個数	次数	個数	次数	個数
20	1117	23	7	26	96	29	8
21	164	24	25000	27	2392	30	5712
22	59	25	211	28	1854	計	36620

実験結果

表 1. 各次数の合計計算時間

次数	個数	GAP4	ISSAC00	AC05	CA-ALIAS05	単位
20	1117	744	4.2	4.6	4.8	分
21	164	1951	1	0.6	0.6	分
22	59	60 (10)	0.2	2.6	0.2	分
23	7	86	1	0.5	0.6	秒
24	25000	39 (26)	5.4	7.3	3.2	時間
25	211	3255 (6)	1.4	81	1.2	分
26	96	10 (24)	0.07	6	0.09	時間
27	2392	200(202)	189	10	1.9	時間
28	1854	263(256)	14	2	0.9	時間
29	8	0.4	1.8	1.2	1.4	秒
30	5712	32(231)	24	1.8	0.2	日
計	36620	58(755)	32	2.8	0.5	日

注.(**) 内の個数の群では，10 時間で計算が終らず，打ち切った.

表 2. 時間区分内で計算できた群の個数 (1)

時間	GAP4	ISSAC00	AC05	CA-ALIAS05
$*\leq$0.1 秒	10510	1829	305	125
0.1 秒 $<*\leq$0.2 秒	11728	7231	1223	1220
0.2 秒 $<*\leq$0.5 秒	5433	22898	12248	24260
0.5 秒 $<*\leq$1 秒	2200	2973	16089	9947
1 秒 $<*\leq$2 秒	1098	629	3742	646
2 秒 $<*\leq$5 秒	1015	363	1921	236
5 秒 $<*\leq$10 秒	621	182	682	68
10 秒 $<*\leq$30 秒	834	232	229	39
30 秒 $<*\leq$1 分	381	126	71	14
1 分 $<*\leq$2 分	480	40	43	29
2 分 $<*\leq$5 分	486	30	32	25
5 分 $<*\leq$10 分	357	6	8	5
10 分 $<*\leq$30 分	348	9	16	4
30 分 $<*\leq$1 時間	114	12	1	2
1 時間 $<*\leq$2 時間	63	15	2	0
2 時間 $<*\leq$5 時間	112	24	3	0
5 時間 $<*\leq$10 時間	85	7	4	0
10 時間 $<*$	755	14	1	0

表 3. 時間区分内で計算できた群の個数 (2)

time	GAP4	ISSAC00	AC05	CA-ALIAS05
$*\leq$0.1 秒	10510	1829	305	125
$*\leq$0.2 秒	22238	9060	1528	1345
$*\leq$0.5 秒	27671	31958	13776	25605
$*\leq$1 秒	29871	34931	29865	35552
$*\leq$2 秒	30969	35560	33607	36198
$*\leq$5 秒	31984	35923	35528	36434
$*\leq$10 秒	32605	36105	36210	36502
$*\leq$30 秒	33439	36337	36439	36541
$*\leq$1 分	33820	36463	36510	36555
$*\leq$2 分	34300	36503	36553	36584
$*\leq$5 分	34786	36533	36585	36609
$*\leq$10 分	35143	36539	36593	36614
$*\leq$30 分	35491	36548	36609	36618
$*\leq$1 時間	35605	36560	36610	36620
$*\leq$2 時間	35668	36575	36612	36620
$*\leq$5 時間	35780	36599	36615	36620
$*\leq$10 時間	35865	36606	36619	36620
–	36620	36620	36620	36620

表 4. いくつかの悪い結果の例 TransitiveGroup(n,i) (単位：秒)

n	i	GAP4	ISSAC00	AC05	CA-ALIAS05
27	554	2.3	0.4	0.9	201
27	593	3.5	0.3	0.8	197
30	390	128	0.2	16	642
30	392	210	0.2	27	644
30	834	14.7	1	3.6	3087
30	841	16	1.2	3.7	3149
27	583	43	13	7065	195
27	620	5.8	16	3480	105
27	863	1.6	1.5	787	80
30	300	98	108	34712	97
30	545	82	34	20713	216
30	563	215	56	42935	185
30	826	9.2	7.3	13619	325
30	840	58	51	13479	368
28	1075	16	69	0.3	0.4
30	1149	29	86	1.8	1.7
30	1367	39	118	0.6	0.5

実験結果について：

表 1 では，各プログラムの全般的な全般的な性能が比較されている．表 2 および表 3 では，各プログラムの特長を示している．今回のプログラムでは，どの可移な置換群の対称群における正規化群でも 1 時間以内に計算可能となっているが，他のプログラムでは 10 時間以上かかる場合が残っている．しかし，今回のプログラムでも，次数が小さい群の計算でも，30 分以上かかる場合がある．表 3 では，今回の目標の 1 つであるように簡単なプログラムであるにもかかわらず，0.5 秒以内で計算できる群の個数が複雑な ISSAC00 のプログラムより少なくなっている．このことは，次数が大きな群になって，どの場合もある程度の時間がかかるようになるであろうこと考えると，今回のプログラムはどの群も長時間かかってしまうかもしれないという欠点がある．

表 4 は，GAP4 で簡単に計算できるのに，他のプログラムでは時間がかかる場合を挙げてある．表 4 を見ると，むしろ複雑な処理をしている ISSAC00 プログラムが GAP4 に比較して，かえって極端に悪くなる場合が少ないという意外な結果がでている．したがって，簡単なプログラムで GAP4 の計算が速くなるように改良して，しかも，かえって時間がかかるという不運な場合が無いようにしたいという今回の目標は，残念ながら，達成されていない．

GAP の Normalizer と AutomorphismGroupPermGroup

GAP の関数 AutomorphismGroupPermGroup が置換群の正規化群の計算を直接している．関数 Normalizer は，対称群と交代群の中で計算するときは，適当な部分群 W を求めた後，AutomorphismGroupPermGroup を呼び出して，W の中で正規化群を計算している．しかし，これが返って計算を遅くしていることがある．その様な例を表 5 に挙げた．いずれも可移でない群である．

表 5. Normalizer と AutomorphismGroupPermGroup の比較 (単位：秒)

G	H	Normalizer	Automorphism.
Stabilizer(PrimitiveGroup(81,123),81)	$Sym(80)$	37	0.2
Stabilizer(PrimitiveGroup(100,3),100)	$Sym(99)$	77	0.3
Stabilizer(PrimitiveGroup(105,9),105)	$Sym(104)$	12	0.3
Stabilizer(PrimitiveGroup(112,1),112)	$Sym(111)$	19652	0.4
Stabilizer(PrimitiveGroup(120,12),120)	$Sym(119)$	46	0.5

参 考 文 献

[1] The GAP Groups. Gap - groups, algorithms and programming, version 4. *Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany and School of Mathematical and Computational Sciences, Univ. St. Andrews, Scotland*, 2000.

[2] 宮本泉: Association scheme を利用した対称群における Normalizer の計算. ftp://tnt.math.metro-u.ac.jp/pub/ac97/PROCEEDINGS/miyamoto/

[3] 宮本泉: GAP の Normalizer 関数の性能の報告と改良の試み. http://tnt.math.metro-u.ac.jp/ac/2005/

[4] I. Miyamoto: Computing normalizers of permutation groups efficiently using isomorphisms of association schemes. In *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation*, pp 220–224, C. Traverso, ed. ACM, 2000.

[5] I. Miyamoto: Computing isomorphisms of association schemes and its applications. *J. Symbolic Comp.*, 32:133–141, 2001.