数式処理 *J.JSSAC (199X)*
Vol. XX, No. XX, pp. 1 - XXX

□

ブロックシステムを利用した可移な置換群の正規化群計算の高速化

宮本 泉

山梨大学*

G, H を $\Omega = \{1, 2, \cdots, n\}$ 上の置換群とする。G の H における正規化群は $\mathsf{Norm}(H, G) = \{\ h \in H | h^{-1}Gh = G\ \}$. で定義される。GAP システムは H が対称群 $Sym(n)$ と交代群 $Alt(n)$ のときに、特別な関数 DoNormalizerSA を使って、正規化群計算を高速化することを行っている。そこで用いられている方法は、例えば、G が可移で非原始的なとき、もし、あるサイズのブロックシステムが一つしか無ければ、正規化群はこのブロックシステムを全体として固定する群として得られる WreathProduct の群 W に含まれることを利用して、$\mathsf{Norm}(Sym(n), G)$ を求めるのに、より小さい群 W の中で、$\mathsf{Norm}(W, G)$ を計算している。可移でない場合も、同じ長さの orbit を集めて、それらを全体として固定する WreathProduct を、同様に利用している。G が可移な場合は、アソシエーションスキームを使えば、$\mathsf{Norm}(Sym(n), G)$ は G の作るアソシエーションスキーム A の自己同型群 $Aut(A)$ に含まれることが分かっている。ここで使われるアソシエーションスキームは、群 G の $\Omega \times \Omega$ 上の orbit 全体を考えているもので、その自己同型は、この G の orbit 達を全体として固定する Ω 上の置換として定義される。しかしながら、アソシエーションスキームの自己同型群の計算にはバックトラック法を使用するので、かなりの計算量が必要になる。一方、正規化群計算で利用される WreathProduct の群は、よくある種類のアソシエーションスキームの自己同型群として得られる。このことから、WreathProduct を利用する方法は、よくある種類のアソシエーションスキームのみを利用した方法と考えることもできる。

もう一つの正規化群計算を、より小さな群の中で計算する方法として、次の事実が得られている。$G \subseteq K$ であって、ある Ω の点 p を含む G の orbit が K の orbit と一致するとき、$\mathsf{Norm}(K, G) = G\mathsf{Norm}(K_p, G_p)$ が成立する。ここに、G_p は G における p の固定部分群を表す。この方法は、G_p の、ある点 q を含む orbit が、K_p の orbit と一致するようなときは、繰返し使うことができる。特に、$K = Sym(n)$ ならば、G が 2 重可移でないと繰返し使えないが、K が WreathProduct や $Aut(A)$ のときは、もっと多くの場合で繰返し使用できる。

筆者は、これらの方法を組合せて、様々な heuristic を使用して正規化群計算の高速化を試

*imiyamoto@yamanashi.ac.jp

みて来た。表にある CA-ALIAS05 の方法は、上の説明中の一般のアソシエーションスキームは使わないで WreathProduct を使った方法で、ブロックシステムも使わないが、今回の方法と比較のために引用した。実験は、GAP システムのライブラリにある 20 次から 30 次までの可移な置換群 TransitiveGroup(n,k) のすべて 36620 個の $Sym(n)$ 中での正規化群を計算するという内容で行った。すべての実験は GAP[1] システムを使って行った。また、$ParGAP$[2] を一度に複数の正規化群を計算して実験時間を節約するために使った。今回の目的は、次に述べる方法であるが、これらの群の全計算時間では、CA-ALIAS05 が、わずかの差ではあるが最速である。

GAP version 4r4 にはもう一つの特別な関数 SubgpConjSymmgp があって、対称群の中の部分群が共役であるかどうかを調べている。この関数でも、部分群があるサイズのブロックシステムを唯一つ持つときに、ブロックシステムを共役で一致させた上で、部分群の、ブロックシステム上の作用と、一つのブロックを固定するような固定部分群の作用を考慮している。今回は、この様な考え方を、次のようにして正規化群計算に利用した。B を G の block とし、 B^G が block の長さが $l=|B|$ となる G の唯一の block system であるとする。G_B を集合 B を全体として固定するような G 中の固定部分群、$G_B|B$ を G_B の B 上への作用として得られる群とする。そして、 $\bar{G}$ を G の B^G 上への作用として得られる群とする。これらの群の正規化群 $H=$Norm$(Sym(l),G_B|B)$ と $K=$Norm$(Sym(n/l),\bar{G})$ を計算して、それらの $W=$WreathProduct(H,K) を構成すれば、G の正規化群は、このより小さな WreathProduct$W$ に含まれることになるので、Norm(W,G) によって正規化群を計算することができる。$G_B|B$ や $\bar{G}$ も、ある長さの block system を唯一もつ場合は、同様な手順で計算を行う。このアルゴリズムを B-Wr で表す。B-Wr では、WreathProduct だけを使って、アソシエーションスキームを使わない。

アソシエーションスキームを使う場合は、G の作るアソシエーションスキーム A の自己同型群 $Aut(A)$ の任意の block system は G の正規化群によって、全体として固定される。したがって、$Aut(A)$ が非原始的ならば、その任意の block systemB^G を使って、上と同様に、より小さな WreathProduct$W$ を求めて、この中で正規化群を計算することができる。このアルゴリズムを B-As で表す。このアルゴリズムについては [3] で、先の 2 つのアルゴリズムについては [4] で紹介している。

これらの block system を利用するアルゴリズムでは、20 次から 30 次までの可移な置換群 TransitiveGroup(n,k) のどの群でも、正規化群計算が 30 から 40 秒程度以内で計算ができた。すべてのプログラムは GAP のプログラミング言語を使って書かれている。表の GAP4 は、GAP の関数 DoNormalizerSA による計算時間を参考のために示した。表は、一番左の列の時間内に、それぞれのアルゴリズムが何個の TransitiveGroup(n,k) の正規化群を計算できるか示している。G が block system をもつということは、G は可移で非原始的な置換群であるということもあって、現状では、表にある 3 つの考案したアルゴリズムは、どれも G が可移場合のみの計算しかできていない。

表 1: Computing times of the normalizers of transitive groups of degree n in $Sym(n)$, $20 \leq n \leq 30$

time range	GAP4	CA-ALIAS05	B-As	B-Wr
$* \leq$0.2sec	22238	956	38	85
0.2sec $< * \leq$0.5sec	5433	24213	1266	1575
0.5sec $< * \leq$1sec	2200	10377	9831	15698
1sec $< * \leq$3sec	1572	788	24380	18994
3sec $< * \leq$10sec	1162	170	462	237
10sec $< * \leq$40sec	1005	39	643	31
40sec $< * \leq$5min	1176	66	0	0
5min $< * \leq$30min	705	9	0	0
30min $< * \leq$1h	114	2	0	0
1h $< * \leq$10h	260	0	0	0
10h $< * \leq$?	755	0	0	0
total time	?	10.6h	17.0h	11.4h

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] G. Cooperman, *Parallel GAP/MPI (ParGAP/MPI)*, Version 1, College of Computer Science, Northeastern University, 1999, `http://www.ccs.neu.edu/home/gene/pargap.html`.

[3] I. Miyamoto. Performance of the GAP-function Normalizer and an attempt of its improvement II. in `http://www.ricam.oeaw.ac.at/srs/groeb/schedule\_D1.html`.

[4] I. Miyamoto. An Improvement of GAP Normalizer Function for Permutation Groups in *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pp 234–238, J-G. Dumas, ed. ACM, 2006.