国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第17卷第1期 1995年3月 Vol. 17 No. 1

任意长二维DFT的多项式变换算法及其并行算法*

蒋增荣 付 彬**

(国防科技大学系统工程与数学系，长沙：410073)

摘 要 本文给出任意长二维DPT的FPT算法及其并行算法，详细地讨论了$N=p^c$的情况(p为素数)。与通常二维DFT的行列算法比较，乘法量减少约50%，加法量略有增加。

关键词 离散富里叶变换(DFT)，快速多项式变换(FPT).

分类号 TN911.72

关于多项式变换的理论，人们已经深入地研究过了[1]~[3]，它在数学信号及图象信号处理中的应用也已越来越广泛[4]~[6]。多项式变换特别适合多维信号的并行处理。本文讨论当N为任意正整数时，$N\times N$二维DFT的多项式变换算法及其并行计算，为此，首先简要的讨论$N=2^t$的情况，然后较详细的讨论$N=p^c$的情况(p为素数)，最后利用Good下标映射，给出任意长度二维DFT的多项式变换算法及其并行算法。

设x_{n_1,n_2}为$N\times N$二维复序列，其二维DFT定义为

$$X_{k_1,k_2}=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}x_{n_1,n_2}W_N^{n_1k_1}W_N^{n_2k_2},\quad(k_1,k_2=0,1,\cdots,N-1)\tag{1}$$

其中$W_N=e^{-2i\pi/N}$.

1

当$N=2^t$时，(1)的FPT算法最为简单。参考文献[1]中有详细讨论。读者可仿第2节的情况写出这时的并行算法，此处从略。

2

当$N=p^c$(p为奇素数，c为正整数)，可将(1)式按$k_2=l\bmod p$，即$k_2=pu+l$($l=0,1,\cdots,p-1$；$u=0,1,\cdots,N/p-1$)分离为如下p个式子

* 国防预研基金资助项目
** 现在湖南大学工作
1994年3月21日收稿

$$X_{k_1,pu+l}=\sum_{n_1=0}^{N-1}\sum_{u=0}^{N/p-1}x^{(l)}_{n_1,n_2}W_N^{n_1k_1}W_N^{(pu+l)n_2} \tag{2}$$

$$(k_1=0,1,\cdots,N-1;u=0,1,\cdots,N/p-1;l=0,1,\cdots,p-1)$$

其中

$$x^{(l)}_{n_1,n_2}=\sum_{n=0}^{p-1}x_{n_1,n_2+nN/p}W_p^{nl} \tag{3}$$

$$(n_1=0,1,\cdots,N-1;n_2=0,1,\cdots,N/p-1;l=0,1,\cdots,p-1)$$

(3)式是 N^2/p 个 p 点一维 DFT，如设 $M_u(p)$ 和 $A_d(p)$ 是 p 点 DFT 所需的乘、加法量，那么计算 $x^{(l)}_{n_1,n_2}$ 共需 $N^2M_u(p)/p$ 和 $N^2A_d(p)/p$ 次乘法和加法。

由于

$$z^N-1=(z^{N/p}-1)(z^{(p-1)N/p}+z^{(p-2)N/p}+\cdots+z^{N/p}+1)\triangleq(z^{N/p}-1)Pe(z)$$

而

$$z^N-1=\prod_{k=0}^{N-1}(z-W_N^k)=\prod_{k\equiv 0\bmod p}(z-W_N^k)\cdot\prod_{k\not\equiv 0\bmod p}(z-W_N^k)$$

$$z^{N/P}-1=\prod_{k\equiv 0\bmod p}(z-W_N^k)$$

故

$$P_l(z)=\prod_{k\not\equiv \bmod p}(z-W_N^k)$$

即

$$P_r(z)\equiv 0\mod(z-W_N^{pu+l}).\ (l\neq 0)$$

下面以$\langle k_1\rangle$表示 k_1 关于模 N 的最少非负剩余，即$\langle k_1\rangle\triangleq_1 \bmod N$. 由于 $l\neq 0$ 时，$(pu+l,N)=1$，故当 $k_1=0,1,\cdots,N-1$ 时，$\langle k_1(pu+l)\rangle$也取值 $0,1,\cdots,N-1$，只是排列次序不同。

利用上述两点，(2)式中的六个式可分别计算如下：

(1)当 $l=0$ 时，改变(2)式的计算次序，有

$$\begin{aligned}X_{\langle k_1(pu+1)\rangle,pu}&=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N/p-1}x^{(0)}_{n_1,n_2}W_N^{n_1k_1(pu+1)}W_N^{n_2pu}\\&=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N/p-1}\{x^{(0)}_{n_1,n_2}W_N^{-n_2}\}W_N^{n_1k_1(pu+1)}W_N^{n_2pu+n_2}\\&=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N/p-1}\{x^{(0)}_{n_1,n_2}W_N^{-n_2}\}z^{n_2}z^{n_1k_1}\bmod P_r(z),(z-W_N^{pu+1})\end{aligned}$$

这只要以 $z=W_N^{pu+1}$ 代入就可验证上式的正确性。关于以 $P_r(z)$为模的运算，是因为 $P_w(z)\equiv 0\bmod(z-W_N^{pu+1})$的缘故。于是，若令

$$(\mathrm{I})\quad\begin{cases}X^{(*,0)}_{n_1,n_2}=W_N^{-n_2}x^{(0)}_{n_1,n_2},(n_1=0,1,\cdots,N-1;n_2=0,1,\cdots,N/p-1)\\X^{(0)}_{n_1}(z)=\sum_{n_2=0}^{N/p-1}x^{(*,0)}_{n_1,n_2}z^{n_2},(n_1=0,1,\cdots,N-1)\\\overline{X}^{(0)}_{k_1}(z)=\sum_{n_1=0}^{N-1}X^{(0)}_{n_1}(z)z^{n_1k_1}\bmod P_r(z),(k_1=0,1,\cdots,N-1)\end{cases}\tag{4}$$

就有

$$X_{\langle k_1(P_u+1)\rangle,pu}\equiv\overline{X}^{(0)}_{k_1}(z)\bmod(z-W_N^{pu+1}),\tag{5}$$

$$(k_1=0,1,\cdots,N-1;u=0,1,\cdots,N/p-1)$$

由于$(P_r(z),z,N)$构成多项式变换[1]，故(4)式是$\{X^{(0)}_{n1}(Z)\}$的多项式变换，可用 FPT 计算，只需$(p-1)NA_d(N)/p+(p-1)N^2/p$ 次加法(这里第二项是模运算所需的加法量，

$A_d(N)$是 N 点一维 FFT 所需的加法量)。如设(4)式的计算结果为

$$\overline{X}_{k_1}^{(0)}(z)=\sum_{m=0}^{(p-1)N/p-1}y_{k_1,m}^{(0)}z^m,(k_1=0,1,\cdots,N-1)$$

则由(5)式有

$$\begin{aligned}X_{(k_1(pu+1))_N,pu}&=\sum_{m=0}^{(p-1)N/p-1}y_{k_1,m}^{(0)}W_N^{m(pu+1)}\\&=\sum_{m=0}^{N/p-1}\left[W_N^m\sum_{j=0}^{p-2}W_p^jy_{k_1,m+jN/p}^{(0)}\right]W_{N/p}^{um}\qquad(6)\\(k_1&=0,1,\cdots,N-1;u=0,1,\cdots,N/p-1)\end{aligned}$$

(6)式是以 k_1 为参数的序列 $\overline{y}_{k_1,m}^{(0)}=W_N^m\sum_{j=0}^{p-2}W_p^jy_{k_1m+jN/p}^{(0)}$的 N/p 点一维 DFT(共 N 个),可用基$-p$FFT 计算。因此由方程组(I)和(6)式计算 $X_{R_1,pu}$共需

$$M_1=NM_u\left(\frac{N}{p}\right)+N^2-N$$

$$A_1=NA_d\left(\frac{N}{p}\right)+\frac{p-1}{p}NA_d(N)+\frac{2p-3}{p}N^2$$

次乘法和加法(其中 $M_u(k)$,$A_d(k)$表示 k 点 FFT 的乘加量,下同)。

(2)$l\neq0$,改变(2)式计算次序,有

$$\begin{aligned}X_{(k_1(pu+l))_N,pu+l}&=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N/p-1}x_{n_1,n_2}^{(l)}W_N^{n_1k_1(pu+l)}W_N^{n_2(pu+l)}\\&\equiv\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N/p-1}x_{n_1,n_2}^{(l)}z^{n_2}z^{n_1k_1}\bmod P_r(z),(z-W_N^{pu+l})\end{aligned}$$

因此,若令

$$(\text{Ⅱ})\quad\begin{cases}X_{n_1}^{(l)}(z)=\sum_{n_2=0}^{N/p-1}x_{n_1,n_2}^{(l)}z^{n_2},(n_1=0,1,\cdots,N-1)\\\overline{X}_{k_1}^{l}(z)=\sum_{n_1=0}^{N-1}X_{n_1}^{(l)}(z)z^{n_1k_1}\bmod P_r(z),(k_1=0,1,\cdots,N-1)\end{cases}\qquad(7)$$

则有

$$\begin{aligned}X_{(k_1(Pu+1))_N,pu+l}&\equiv\overline{X}_{k_1}^{(l)}(z)\bmod(z-W_N^{pu+l}),\qquad(8)\\(k_1&=0,1,\cdots,N-1;u=0,1,\cdots,N/p-1,l\neq0)\end{aligned}$$

与(4)式一样,(7)式是多项式序列$\{X_{n_1}^{(l)}(z)\}$的多项式变换,可用相同方法计算,所需加法也相同,如设(7)式的计算结果为

$$\overline{X}_{k_1}^{(l)}(z)=\sum_{m=0}^{(p-1)N/p-1}y_{k_1,m}^{(l)}z^m,(k_1=0,1,\cdots,N-1,l\neq0)$$

则由(8)式有

$$\begin{aligned}X_{(k_1(pu+l))_N,pu+l}&=\sum_{m=0}^{(p-1)N/p-1}y_{k_1,m}^{(l)}W_N^{m(pu+l)}\\&=\sum_{m=0}^{N/p-1}\left[W_N^{ml}\sum_{j=0}^{p-2}W_p^jy_{k_1,m+jN/p}^{(l)}\right]W_{N/p}^{um}\qquad(9)\\(k_1&=0,1,\cdots,N-1;u=0,1,\cdots,N/p-1;l\neq0)\end{aligned}$$

(9)式是以 k_1 为参数的序列 $\bar{y}^{(l)}_{k_1,m}=W^{ml}_N\sum_{j=0}^{p-2}W^{jl}_p y^{(l)}_{k_1,m+jN/p}$ 的 N/p 点 DFT(共 N 个),可用基-pFFT 计算。因此用方程组(Ⅱ)和(9)式计算 $X_{k_1,pu+l}(l\neq 0)$的运算量是

$$M_2=NM_u\left(\frac{N}{p}\right)+\frac{p-1}{p}N^2$$

$$A_2=NA_d\left(\frac{N}{p}\right)+\frac{p-1}{p}NA_d(N)+\frac{2p-3}{p}N^2$$

综上所述,$p^c\times p^c$ 二维 DFT 的 FPT 算法为:

Step1 计算(3)式中 N^2/p 个 p 点 DFT.

step2 利用方程组(Ⅰ)和(5)式计算 $X_{k_1,pu}$.

step3 利用方程组(Ⅱ)和(9)式计算 $X_{k_1,pu+l}(l\neq 0)$.

step4 结束。

上述 FPT 算法的运算量为

$$\begin{aligned}M(N)&=\frac{N^2}{p}M_u(p)+M_1+(p-1)M_2\\&=pNM_u\left(\frac{N}{p}\right)+\frac{M_u(p)+p^2-p+1}{p}N^2-N\\A(N)&=\frac{N^2}{p}A_d(p)+A_1+(p-1)A_2\\&=pNA_d\left(\frac{N}{p}\right)+(p-1)NA_d(N)+\frac{A_d(p)+2p^2-3p}{p}N^2\end{aligned}\tag{10}$$

特别,当 $p=3,N=p^s$ 时,$N\times N$ 二维 DFT 的 FPT 算法运算量为

$$\begin{aligned}M(3^s)&=3NM_u\left(\frac{N}{3}\right)+3N^2-N=2sN^2+2N\\A(3^S)&=3NA_d\left(\frac{N}{3}\right)+2NA_d(N)+5N^2=(6s+3)N^2\end{aligned}\tag{11}$$

其中用到了 $M_u(3^S)=2s\cdot 3^S-3^S+1$,$A_d(3^S)=2s\cdot 3^S(s>1$,$M_u(3)=2$,$A_d(3)=6$.

当用行列算法计算 $3^S\times 3^S$ 二维 DFT 时,运算量为

$$\begin{aligned}M_0(3^S)&=2\cdot NM_u(3^S)=(4s-2)N^2+2N,\\A_0(3^S)&=2\cdot N^SA_d(3^S)=4sN^2\end{aligned}\tag{12}$$

比较(11)与(12)式可知,FPT 算法的乘法量约减少一半,加法量有所增加(可用下述改进算法使加法量减少)。

下面给出上述 FPT 算法的并行算法。

Algorithm Ⅰ 向量计算机的向量并行算法。

Step1 用多道 FFT 计算(3)式中 N 个 p 点 DFT;所需运算量是 $NMu(p)/p$ 次和 $NA_d(p)/p$ 次向量长度为 N 的并行乘法和加法。

Step2 利用方程组(*I*)和(5)式向量并行计算 $X_{k_1,pu}$:方程(Ⅰ)的第一式需 $N/p-1$ 次长度为 N 的并行乘法;用 PFPT 计算多项式变换需 $A_d(N)+N$ 次长度为 $(p-1)N/p$ 的并行加法;(5)式用多道 FFT 计算,共需 $M_u(N/p)+(p-1)N/p$ 次和 $A_d(N/p)+(p-2)N/p$ 次长度为 N 的并行乘法和加法。

Step3 利用方程组(Ⅱ)和(9)式向量并行计算 $X_{k_1,pu+l}$:这一步除 Step2 中第一步外,其余相同。

Step4 结束。

Algorithm Ⅰ所需的运算量为

$$\overline{M} = pM_u\left(\frac{N}{p}\right) + [M_u(p) + p^2 - p + 1]N/p - 1$$

$$\overline{A} = pA_d\left(\frac{N}{p}\right) + [A_d(p) + p^2 - 2p]N/p \tag{13}$$

次向量长度为 N 的并行乘法和加法,以及

$$\overline{A}^* = pA_d(N) + pN$$

次向量长度为 $(p-1)N/p$ 的并行加法。加速比约等于 N.

Algorithm Ⅱ MIMD 计算机上的向量并行算法。

Step1 //只有主进程工作//

1)主进程占有一台向量计算机,向量并行计算(3)式中的 $x^{(l)}_{n_1,n_2}$;

Step2 //并行计算//

2)创建 $p-1$ 个并行进程,各占有一台向量计算机;

3)主进程利用方程组(Ⅰ)和(5)式向量并行计算 $X_{k,pu}$,方法同 Algorithm Ⅰ的 step 2;

4)其余 $p-1$ 个进程,各利用方程组(Ⅱ)和(9)式向量并行计算 $X_{k_1,pu+l}$($l=1\sim p-1$),方法同 3.

5)终止并行进程。

Step3 结束。

Algorithm Ⅱ所需运行量为

$$\overline{M} = M_u\left(\frac{N}{p}\right) + [M_u(p) + p]N/p - 1$$

$$\overline{A} = A_d\left(\frac{N}{p}\right) + [A_d(p) + p - 2]N/p \tag{14}$$

次向量长度为 N 的并行乘法和加法,以及

$$\overline{A}^* = A_d(N) + N$$

次向量长度为 $(p-1)N/p$ 的并行加法。加速比约等于 pN.

最后指出,上述算法还可加以改进,事实上,当 $l=0$ 时,(2)式就成为

$$X_{k_1,pu} = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N/p-1} x^{(0)}_{n_1,n_2} W_N^{n_1 k_1} W_{N/p}^{n_2 k_2},$$

$$(k_1 = 0,1,\cdots,N-1; u = 0,1,\cdots,N/p-1)$$

这是一个 $N\times(N/p)$二维 DFT,可按 $k_1 = l \bmod p$ 对它进行分离。对于 $l\neq 0$ 的 $p-1$ 个式子,可利用多项式变换$(P_r(z), z^t, N/p)$和$(P_r(z), z, N)$计算,而 $l=0$ 的那一项就成为 $(N/p)\times(N/p)$二维 DFT,可用同一过程计算,直至 $p\times p$ 二维 DFT 为止。$p\times p$ 二维 DFT 也可用 FPT 计算[1]。如记 $M(N)$和 $A(N)$为用这种递归算法计算 $N\times N$ 二维 DFT 所需乘法和加法运算量,则有

$$M(N) = M(N/p) + (p^2 - 1)M_u(N/p)N/p + [p^3 - p^2 - p + 1 + pM_u(p) + M_u(p)]N^2/p^2$$

$$A(N) = A(N/p) + (p - 1)^2A_d(N/p)N/p + 2(p - 1)NA_d(N/p) + [3p^3 - 6p^2 + p + 2 + pA_d(p) + A_d(p)]N^2/p^2$$

与前面一样，$M_u(k)$，$A_d(k)$表示 k 点一维 FFT 的乘、加量。特别，当 $p=3$，$N=3^S$ 时，有

$$M(N) = M(N/3) + \frac{16}{9}sN^2 + \frac{8}{3}N$$

$$A(N) = A(N/3) + \frac{32}{9}sN^3 + \frac{8}{3}N^2$$

上述算法反复使用，直至 3×3 二维 DFT 为止。由 $M(3)=8$，$A(3)=36$，最后得

$$M(N) = \left(2s - \frac{1}{4}\right)N^2 + 4N - \frac{79}{4}$$

$$A(N) = \left(4s + \frac{5}{2}\right)N^2 - \frac{45}{2} \tag{15}$$

与(11)式和(12)式比较，运算量有所减少，特别加法量减少更多。

3

现在来给出任意长二维 DFT 的 FDT 算法及其并行算法。

当 $N=N_1 \cdot N_2$，$(N_1,N_2)=1$，利用简单映射和孙子定理映射[1]-[4]，可将 $N\times N$ 二维 DFT(1)式映射成$(N_1\times N_1)\times(N_2\times N_2)$四维 DFT，当应用素因子算法计算这个四维 DFT 时，运算量为

$$M = N_1^2M_2 + N_2^2M_1; A = N_1^2A_2 + N_2^2A_1$$

当用 Winograd 嵌套算法计算这个四维 DFT 时，运算量为

$$M = M_1M_2; \quad A = N_2^2A_1 + M_1A_2$$

其中 M_i，A_i 为 $N_i\times N_i$ 二维 DFT 的运算量。当 $N_i=2^t$ 或 p_c 时，$N_i\times N_i$ 二维 DFT 如前述可用 FPT 算法及其并行算法计算，因此 N 有双因子时，$N\times N$ 二维 DFT 可用 FPT 算法及其并行算法计算。

当 $N=N_1N_2\cdots N_r$，$(N_i,N_j)=1(i\neq j)$时，同样用简单映射和孙子定理映射将(1)式映射为$(N_1\times N_1)\times(N_2\times N_2)\times\cdots\times(N_r\times N_r)2r$ 维 DFT，当用素因子算法时，运算量为

$$M = \sum_{i=1}^{r} M_i \frac{N^2}{N_i^2}, \quad A = \sum_{i=1}^{r} A_i \frac{N^2}{N_i^2} \tag{16}$$

当用 Winograld 嵌套算法时，运算量为

$$M = \prod_{i=1}^{r} M_i, \quad A = A_1 \frac{N^2}{N_1^2} + M_1A_2 \frac{N^2}{N_1^2N_2^2} + \cdots + M_1M_2\cdots M_{r-1}A_r \tag{17}$$

当 $N_i=2^t$ 或 $p_i^{t_i}$ 时，各个 $N_i\times N_i$ 均可用前述 FPT 算法及其并行算法计算。

参 考 文 献

1 蒋增荣，曾泳泓，多项式变换及其应用. 长沙：国防科技大学出版社，1989

2 蒋增荣. 模 $Z^{2q}-Z^{q}+1$ 的多项式变换，湖南数学年刊，1987，(2)

3 H J Nussbaumer，P Quandalle Fast computation of discrede Fourxer transforms using polymomial transforms，IEEE Trans.(1979)Assp－27:169－181

Fast Polynomail Transform Algorithms and Parallel Algorithms for Two－Dimensional DFT of Arbitrary Length

Jiang Zengrong　Fu Bin

(Department of System Enginaering and Mathematics)

Abstract

In this paper，We present FPT algorithms and their parallel algorithms for $N\times N$ 2D－DFF，where N is any positive integer. Especially we describe the case $N=p^{c}$ in detail (where p is a prime number). As compared to the ordinary column－row algorithm of 2D－DFT，the number of multiplications of the FPT algorithm is reduced about 50%，while the number of additions increases a little.

Key words　Discrete Foutier Transform (DFT)，Fast Polynomial Transform(FPT)