arXiv:2303.01185v1 [math.CO] 2 Mar 2023

计算傅里叶-戴德金和的多项式时间算法

GUOCE XIN[1], *, XINYU XU[2]

Abstract. 我们解决了 Matthias Beck 和 Sinai Robins 在《离散地计算连续对象》一书中提出的一个开放问题。具体来说，我们提出了一种计算傅里叶-戴德金和的多项式时间算法。该算法是简单的模 Barvinok 单纯锥分解。它可以很容易地整合到 De Leora 等人的 LattE 软件包中，从而实现 Barvinok 多项式时间算法。

Ma主题分类：主要分类 11F20；次要分类 11Y16、05A15、11L0 3.
关键词：戴德金和；傅里叶-戴德金和；Barvinok 算法；常数项。

1. Introduction

这份草稿是一则公告。完整版本将很快完成。

戴德金和是重要的数论对象，出现在数学的许多领域中，包括数论、几何、拓扑学、算法复杂性等。详情及更多参考文献，参见 [3] 。傅里叶-戴德金和统一了文献中出现的戴德金和的许多变体，并构成埃尔哈特拟多项式的基本组成部分。傅里叶-戴德金和定义为

$$s_n(a_1,a_2,\dots,a_d;b)=\frac{1}{b}\sum_{k=1}^{b-1}\frac{\xi^{kn}}{(1-\xi_b^{ka_1})\cdot(1-\xi_b^{ka_2})\cdots(1-\xi_b^{ka_d})}, \tag{1.1}$$

其中 a_1，a_2，……，a_d，$b\in\mathbb{N}$并且 $b>1$与每个 a_i和 $\xi_b=e^{\frac{2\pi i}{b}}$互素。 —

关于傅里叶-戴德金和的以下公开问题由 Matthias Beck 和 Sinai Robins 在 [3]中提出。

问题 1（公开问题）。已知 *[2]* 傅里叶-戴德金和可以高效计算。请找出一种可在实践中实现的快速算法。

我们通过在 [5] 中使用常数项概念，并简单应用 Barvinok 算法，给出了所需的多项式时间算法，从而解决了这一未解决问题。该算法可以轻松改编到 De Loera 等人开发的软件包 `LattE` [4] 中。

2. The polynomial time algorithm

在本节中，我们假设 a_1、a_2、……、$a_d\in\mathbb{N}$与 b互素，除非另有说明。

日期：2023年3月3日。

2.1. 简要介绍。这里需要将 Elliott 有理函数 E 写成如下形式。

$$E = \frac{L(\lambda)}{\prod_{i=1}^{n}(1-u_i\lambda^{a_i})} \tag{2.1}$$

其中， $L(\lambda)$ 是一个 Laurent 多项式， u_i 与 λ 互素，且对于所有i， a_i 均为正整数。该算法主要依赖以下已知结果。

命题 1. 假设 E 的部分分式分解为

$$E = P(\lambda) + \frac{p(\lambda)}{\lambda^k} + \sum_{i=1}^{n} \frac{A_i(\lambda)}{1-u_i\lambda^{a_i}}, \tag{2.2}$$

其中， u_i不含 λ， $P(\lambda)$ $p(\lambda)$，且 $A_i(\lambda)$均为多项式，次数 $p(\lambda) < k$，且次数 $A_i(\lambda) < a_i$ 对所有 i均成立。则有

$$\operatorname*{CT}_{\lambda} E = P(0) + \sum_{u_i\lambda^{a_i}<1} A_i(0).$$

定义 2. 我们记

$$\operatorname*{CT}_{\lambda} \frac{1}{\underline{1-u_s\lambda^{a_s}}} E(1-u_s\lambda^{a_s}) := A_s(0).$$

一般情况下，对于任意 $\varnothing \neq I \subseteq [n]$ ， 我们记

$$\operatorname*{CT}_{\lambda} \frac{1}{\underline{\prod_{s\in I}(1-u_s\lambda^{a_s})}} E \cdot \prod_{s\in I}(1-u_s\lambda^{a_s}) := \sum_{s\in I} A_s(0).$$

我们的算法基于以下观察。

命题 3. 假设 $F(\lambda)$是一个有理函数，并且对于 $k = 1$、2、……、 b， $F(\xi_b^k)$均存在，其中 $\xi_b = e^{\frac{2\pi i}{b}}$。那么我们有

$$\frac{1}{b}\sum_{k=1}^{b-1} F(\xi_b^k) = \operatorname*{CT}_{\lambda} \frac{1}{\underline{1-\lambda^b}} F(\lambda) - \frac{1}{b}F(1).$$

证明。 该命题可由以下已知恒等式推出。

$$\operatorname*{CT}_{\lambda} \frac{1}{\underline{1-\lambda^b}} F(\lambda) = \operatorname*{CT}_{\lambda} \frac{1}{\underline{(1-\lambda)\cdot(1-\xi_b^{-1}\lambda)\cdots(1-\xi_b^{1-b}\lambda)}} F(\lambda) = \frac{1}{b}\sum_{k=1}^{b-1} F(\xi_b^k) + \frac{1}{b}F(1).$$

∎

推论 4. 设 $d \geq 1$。那么 $s_n(a_1, a_2, \ldots, a_d; b)$ 可以写成

$$s_n(a_1, a_2, \ldots, a_d; b) = \Big(Q_z - \frac{1}{b(1-z_1)\cdots(1-z_d)}\Big)\Big|_{z_i=1}.$$

其中 $Q_z = \operatorname{CT}_{\lambda} \frac{\lambda^n}{\underline{1-\lambda^b \cdot 1-\lambda^{a_1}z_1 ()()\cdots(1-\lambda^{a_d}z_d)}}$。

证明. 将命题 3 应用于 $F(\lambda) = \frac{\lambda^n}{\underline{(1-\lambda^{a_1}z_1)\cdots(1-\lambda^{a_d}z_d)}}$. 我们得到

$$\frac{1}{b}\sum_{k=1}^{b-1} \frac{\xi_b^{kn}}{(1-\xi_b^{ka_1}z_1)\cdots(1-\xi_b^{ka_d}z_d)} = \operatorname*{CT}_{\lambda} \frac{1}{\underline{1-\lambda^b}} F(\lambda) - \frac{1}{b(1-z_1)\cdots(1-z_d)}.$$

对所有 i 令 $z_i = 1$ 趋于极限，得到

$$s_n(a_1, a_2, \ldots, a_d; b) = \Big(\operatorname*{CT}_{\lambda} \frac{\lambda^n}{\underline{(1-\lambda^b)} \cdot (1-\lambda^{a_1} z_1) \cdots (1-\lambda^{a_d} z_d)} - \frac{1}{b(1-z_1)\cdots(1-z_d)} \Big)\Big|_{z_i=1},$$

如所需。 ∎

2.2. **步骤　　算法**。 我们使用软件包 LattE 来计算 $s_n(a_1, a_2, \ldots, a_d; b)$.

首先，通过添加一个松弛变量 z_0 ，我们可以写成

$$Q_z = \Big(\operatorname*{CT}_{\lambda} \frac{\lambda^n}{\underline{(1-\lambda^b z_0)} \cdot (1-\lambda^{a_1} z_1) \cdots (1-\lambda^{a_d} z_d)} \Big)\Big|_{z_0=1}.$$

为方便起见，我们令

$$\widetilde{Q_z} := \operatorname*{CT}_{\lambda} \frac{\lambda^n}{\underline{(1-\lambda^b z_0)} \cdot (1-\lambda^{a_1} z_1) \cdots (1-\lambda^{a_d} z_d)}.$$

注意到

$$\widetilde{Q_z} = \sum_{\alpha \in P \cap \mathbb{Z}^{d+1}} z^{\alpha}$$

枚举由顶点 $v = (-\frac{n}{b}, 0, \ldots\ldots, 0)^t$ 定义的顶点单纯锥 P 中的格点，其生成元为以下列向量

$$H = \begin{pmatrix} -a_1 & -a_2 & \ldots & -a_d \\ b & 0 & \ldots & 0 \\ 0 & b & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & b \end{pmatrix}.$$

然后我们使用 LattE 来表示

$$\widetilde{Q_z} = \sum_i \widetilde{Q_i}(z_0, z_1, \cdots, z_d)$$

将其写成简单有理函数的短和，并计算极限

$$\Big(\sum_i \widetilde{Q_i}(z_0, z_1, \cdots, z_d) - \frac{1}{b(1-z_1)\cdots(1-z_d)} \Big)\Big|_{z_j=1}.$$

这等于所需的 $s_n(a_1, a_2, \ldots\ldots, a_d; b)$。

算法5。 现在我们将给出一个计算傅里叶-戴德金和s_n（a_1, a_2, ……, a_d; b）的算法。

1. 将松弛变量 z_0 添加到 Q_z 中，得到 $\widetilde{Q_z} = \operatorname{CT}_{\lambda} \frac{\lambda^n}{1-\lambda^b z_0 \cdot 1-\lambda^{a_1} z_1 () () \cdots (1-\lambda^{a_d} z_d)}$。

2. 我们可以使用 *LattE package* 将 $\widetilde{Q_z} = \sum Q_i(z_0, \ldots, z_d)$ 写出。3. 使用 *LattE* 或 *CTEuclid* 消去松弛变量 z_j ，以得到输出。

我们通过使用（初等）CTEuclid算法替代步骤2.3，来说明基本思路。

例6. 计算 $s_4(4,\ 3,\ 5;\ 7)$。

根据傅里叶-戴德金和的定义，我们有 $s_4(4、3、5;7) = \frac{1}{7}\sum_{k=1}^{6} \frac{\xi^{4k}}{1-\xi_7^{4k}} \underline{\frac{}{1-\xi_7^{5k}()()(1-\xi_7^{3k})}}$，其中 $\xi_7 = e^{\frac{2\pi i}{7}}$。

$$
\begin{aligned}
Q_z &= \operatorname*{CT}_{\lambda} \frac{\lambda^4}{\underline{(1-\lambda^7)} \cdot (1-\lambda^4 z_1)(1-\lambda^5 z_2)(1-\lambda^3 z_3)} \\
&= \frac{{z_1}^9}{({z_1}^3 - z_2)\,(z_1 z_3 - 1)\,({z_1}^7 - 1)} - \frac{z_3}{(z_1 z_3 - 1)\,(z_2 {z_3}^3 - 1)\,({z_3}^7 - 1)} \\
&- \frac{{z_1}^3 {z_2}^2}{(z_1 z_3 - 1)\,(z_1 {z_2}^2 - 1)\,({z_1}^3 - z_2)} - \frac{{z_2}^2 z_3}{(z_1 z_3 - 1)\,({z_2}^2 - z_3)\,(z_2 {z_3}^3 - 1)} \\
&+ \frac{{z_2}^4}{(z_1 {z_2}^2 - 1)\,({z_2}^2 - z_3)\,({z_2}^7 - 1)}.
\end{aligned}
$$

Then

$$
\begin{aligned}
s_4(4,3,5;7) &= \Big(Q_z - \frac{1}{(1-z_1)(1-z_2)(1-z_3)}\Big)\Big|_{z_i=1} \\
&= \Big(\frac{{z_1}^9}{({z_1}^3 - z_2)\,(z_1 z_3 - 1)\,({z_1}^7 - 1)} - \frac{z_3}{(z_1 z_3 - 1)\,(z_2 {z_3}^3 - 1)\,({z_3}^7 - 1)} \\
&- \frac{{z_1}^3 {z_2}^2}{(z_1 z_3 - 1)\,(z_1 {z_2}^2 - 1)\,({z_1}^3 - z_2)} - \frac{{z_2}^2 z_3}{(z_1 z_3 - 1)\,({z_2}^2 - z_3)\,(z_2 {z_3}^3 - 1)} \\
&+ \frac{{z_2}^4}{(z_1 {z_2}^2 - 1)\,({z_2}^2 - z_3)\,({z_2}^7 - 1)} - \frac{1}{(1-z_1)(1-z_2)(1-z_3)}\Big)\Big|_{z_i=1} \\
&= \frac{1}{7}.
\end{aligned}
$$

References

[1] Barvinok A I，当维数固定时计算多面体中整数点数量的多项式时间算法*[J]*,《运筹学数学》，1994，19(4)：769-779。[2] Matthias Beck，戴德金余切和*[J]*,《算术学报》，2003，109(2)：109–130。[3] Matthias Beck 和 Sinai Robins。离散地计算连续对象，第 61 卷。Springer，2007。[4] J. A. De Loera、R. Hemmecke、J. Tauzer 和 R. Yoshida，有理凸多面体中的有效格点计数，《符号计算杂志》，38 (2004) 1273–1302。[5] G. Xin，麦克马洪分拆分析的欧几里得式算法，《组合理论杂志》A辑，131：32–60，2015。

3. Appendix: Computer Experiment

我们使用一种新开发的算法 LLLCTEuclid算法给出数据，该算法仍在开发中。本部分不会包含在最终版本中。

计算傅里叶-狄德金和的多项式时间算法 5

```
输出:"C:/Maple Procedure/SLPLCTEuclid.mpl":
Input:      aL :: [a[1], a[2]...a[s]]    (求a[1]那一项的贡献)
            n :: numerator x^n
```

```
XcomFD := proc(aL, n)
  local s, zL, den, L, i, FL, F, LL, Lt, LK;
  s := nops(aL);
  zL := [seq(z[i], i = 1..s - 1)];
  den := (1 - x^(aL[1])) · mul(-x^(aL[i]) · zL[i - 1] + 1, i = 2..s);
  L := x^n / den;
  FL := R2List(L, [x, zL]);
  Lt := Lflip(FL, M → degree(M, x) < 0);
  LL := ContriSingle(1, [op(Lt), [ ]], x);
  F := 1/aL[1] subs(x = 1, -L·(1 - x^(aL[1]))); F := R2List(F, zL);
  LL := [op(LL), F]; LK := Tokappa(LL, [ ], zL);
  Elikappa(LK, [ ]);
  end proc:
#FD  compute by definition, for comparison
FD_test := proc(aL, n)
  local s, F, k, out;
  s := nops(aL);
  F := x^n / (mul(-x^(aL[i]) + 1, i = 2..s));
  out := 0;
  for k from 1 to aL[1] - 1 do
  out := out + evalf(subs(x = e^(2 I k/aL[1]) π, F));
  end do;
  1/aL[1] · out;
  #[%%, %];
  end proc:
aL := [5, 2, 3];
```

$$aL := [5, 2, 3]$$

```
n := 111;
```

$$n := 111$$

```
XcomFD(aL, n);
```

$$-\frac{1}{5}$$

```
FD_test(aL, n);
```

$$-0.2000000078 + 0.\ I$$

```
ct := time( );
```

```
XcomFD(aL, n);
evalf(%);
"used time is    ", time( ) - ct;
```

$$ct := 15.303$$

$$1.101363503\ 10^{17}$$

$$\text{"used time is   "},\ 18.767$$

一个用于比较的随机示例。

```
seq(rand(11..10000)( ), i = 1..3);
```

$$260,\ 3275,\ 7367$$

```
aL := [314159, %];
```

$$aL := [314159,\ 260,\ 3275,\ 7367]$$

```
ct := time( );
XcomFD(aL, n);
evalf(%);
time( ) - ct;
```

$$ct := 9.906$$

$$-\frac{522587}{314159}$$

$$-1.663447490$$

```
ct := time( );
FD_test(aL, n);
time( ) - ct;
```

$$ct := 10.046$$

$$-1.659948513 - 0.1119267569\ I$$

XincomFD 的另外几个示例。

3 个随机数案例

```
seq(rand(101..10000)( ), i = 1..3);
```

$$2500,\ 7207,\ 3297$$

```
aL := [314159, %];
```

$$aL := [314159,\ 2500,\ 7207,\ 3297]$$

```
ct := time( );
XcomFD(aL, n);
evalf(%);
time( ) - ct;
```

$$ct := 44.203$$

$$-\frac{1155256}{314159}$$

$$-3.677297165$$

```
seq(rand(101..10000)( ), i = 1..3);
```

$$3070,\ 9622,\ 4937$$

```
aL := [314159, %];
```

$$aL := [314159,\ 3070,\ 9622,\ 4937]$$

```
ct := time( );
 XcomFD(aL, n);
 evalf(%);
 time( ) - ct;
```

$$ct := 44.312$$

```
seq(rand(101..10000)( ), i = 1..3);
```

$$1249,\ 5622,\ 5469$$

```
aL := [314159, %];
```

$$aL := [314159,\ 1249,\ 5622,\ 5469]$$

```
ct := time( );
 XcomFD(aL, n);
 evalf(%);
 time( ) - ct;
```

$$ct := 44.468$$

$$-\frac{1041468}{314159}$$

$$-3.315098406$$

4 random number cases

```
seq(rand(101..10000)( ), i = 1..4);
```

$$7180,\ 7285,\ 9919,\ 5270$$

```
aL := [314159, %];
```

$$aL := [314159,\ 7180,\ 7285,\ 9919,\ 5270]$$

```
ct := time( );
 XcomFD(aL, n);
 evalf(%);
 time( ) - ct;
```

$$ct := 44.578$$

$$-\frac{17289234}{314159}$$

$$-55.03338755$$

```
seq(rand(101..10000)( ), i = 1..4);
```

$$7370,\ 5831,\ 1883,\ 5443$$

```
aL := [314159, %];
```

(2 3
2 3)
()

```
aL := [314159, 7370, 5831, 1883, 5443]
```

```
ct := time( );
 XcomFD(aL, n);
 evalf(%);
 time( ) − ct;
```

$$ct := 45.078$$

$$-\frac{5015662}{314159}$$

$$-15.96536149$$

```
seq(rand(101..10000)( ), i = 1..4);
```

$$8924,\ 1336,\ 1455,\ 9504$$

```
aL := [314159, %];
```

$$aL := [314159,\ 8924,\ 1336,\ 1455,\ 9504]$$

```
ct := time( );
 XcomFD(aL, n);
 evalf(%);
 time( ) − ct;
```

$$ct := 45.953$$

5 *random number cases*

```
seq(rand(101..10000)( ), i = 1..5);
```

$$2948,\ 789,\ 3797,\ 8311,\ 924$$

```
aL := [314159, %];
```

$$aL := [314159,\ 2948,\ 789,\ 3797,\ 8311,\ 924]$$

```
ct := time( );
 XcomFD(aL, n);
 evalf(%);
 time( ) − ct;
```

$$ct := 46.890$$

$$-\frac{15236553}{314159}$$

$$-48.49949548$$

```
seq(rand(101..10000)( ), i = 1..5);
```

$$1900,\ 4971,\ 4493,\ 7306,\ 310$$

```
aL := [314159, %];
```

$$aL := [314159,\ 1900,\ 4971,\ 4493,\ 7306,\ 310]$$

```
ct := time( );
 XcomFD(aL, n);
```

```
evalf(%);
time() - ct;
```

$$ct := 49.125$$

```
seq(rand(101..10000)(), i = 1..5);
```

$$9322,\ 8799,\ 9223,\ 3053,\ 7907$$

```
aL := [314159, %];
```

$$aL := [314159,\ 9322,\ 8799,\ 9223,\ 3053,\ 7907]$$

```
ct := time();
XcomFD(aL, n);
evalf(%);
time() - ct;
```

$$ct := 52.109$$

$$-\frac{10390915}{314159}$$

$$-33.07533765$$

1, [2]School of Mathematical Sciences, Capital Normal University, Beijing 100048, PR China电子邮箱地址: [1]guoce xin@163.com & [2]xinyu0510x@163.com