

A Euclid style algorithm for MacMahon partition analysis

Guoce Xin

Department of Mathematics
Capital Normal University, Beijing 100048, PR China
guoce.xin@gmail.com

Oct. 20, 2012

Abstract

Solutions to a linear Diophantine system, or lattice points in a rational convex polytope, are important concepts in algebraic combinatorics and computational geometry. The enumeration problem is fundamental and has been well studied, because it has many applications in various fields of mathematics. In algebraic combinatorics, MacMahon partition analysis has become a general approach for linear Diophantine system related problems. Many algorithms have been developed, but “bottle neck” problems always arise when dealing with complex problems. While in computational geometry, Barvinok’s important result asserts the existence of a polynomial algorithm when the dimension is fixed. However, the implementation by the LattE package of Loera et. al. does not perform well in many situations. By combining excellent ideas in the two fields, we generalize Barvinok’s result by giving a polynomial algorithm for MacMahon partition analysis in a suitable condition. We also present a Euclid style algorithm, which might not be polynomial but is easy to implement and performs well. As applications, we contribute the generating series for magic squares of order 6.

Mathematics Subject Classification. Primary 05-04, secondary 05A15, 52B99.

Key words. MacMahon partition analysis, polytopes, lattice points, Ehrhart quasi-polynomials

1 Introduction

Linear Diophantine system is one of the most fundamental concepts in mathematics. One basic problem is to determine the set of non-negative integer solutions of a system of linear equations (or inequalities) $Ax = b$ for suitable integral matrix A and vector b . In the context of geometry, the problem is to determine lattice points in a rational convex polytope $P = \{\alpha : A\alpha = b, \alpha \geq 0\}$ specified by A and b . That is, we need to determine the set $P \cap \mathbb{Z}^n$. If $b = 0$ then the linear Diophantine system is called

homogeneous, and the corresponding P is called a rational cone. This basic problem received much attention for its wide application in many fields of mathematics. Many theories have been developed but this paper will address on the algorithmic aspect. There are many algorithms dealing with linear Diophantine system related problems. Two algorithms in two different fields have great advantages. One is Barvinok's polynomial algorithm in computational geometry [6] and the other is the author's partial fraction algorithm [18] in algebraic combinatorics.

We will use the short hand notation $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ throughout this paper. To understand the complexity of this problem, let us specify that A is a $r \times n$ matrix of rank r and consider the homogeneous case. The structure of the \mathbb{Z} -solutions $\{\alpha \in \mathbb{Z}^n : A\alpha = 0\}$ is simple, since it forms a subgroup of \mathbb{Z}^n with $n - r$ generators, which can be obtained through Hermite normal form. The structure of nonnegative integer solutions $E = \{\alpha \in \mathbb{N}^n : A\alpha = 0\}$ is only a free commutative monoid (semigroup with identity). There is no simple way to enumerate elements of E , which is equivalent to the construction of the *rational* generating series

$$E(x) = E(x; A, 0) = \sum_{\alpha \in E} x^\alpha = \frac{P(x)}{(1 - x^{e_1})(1 - x^{e_2}) \cdots (1 - x^{e_N})},$$

where e_i ranges over all extreme rays of E and $P(x)$ might be a monster polynomial. See [17, Ch. 4.6] and combinatorial theories developed there. Many practical problems only need some specializations of $E(x)$, such as $E(q, q, \dots, q)$. It is worth mentioning that there is a beautiful reciprocity theorem for rational cones due to Stanley, which gives simple connection between nonnegative solutions and positive solutions.

Our algorithms are under the framework of algebraic combinatorics, but borrow some beautiful ideas from computational geometry. The heart problem is to compute the constant term in $\Lambda = (\lambda_1, \dots, \lambda_r)$ of an Elliott-rational function, written as

$$\text{CT}_\Lambda \frac{L}{(1 - M_1)(1 - M_2) \cdots (1 - M_n)}, \quad (1)$$

where L is a Laurent polynomial and M_i are monomials for all i . Following [18], here we specify a field of iterated Laurent series, called working field, to clarify the series expansion of rational functions. It was George Andrews who found that MacMahon's partition analysis could be ideally combined with computer algebra for dealing with linear Diophantine system related problems [3]. Andrews and his coauthors has published a series of 12 papers in this topic. The first such algorithm was developed by Andrews et al. implemented by the Mathematica package Omega [4] and an improvement appears in [5]. A big progress is the author's partial fraction algorithm [18], where the field of iterated Laurent series was introduced and several bottle neck problems were solved. By using iterated Laurent series, factors like $(1 - \lambda_1/\lambda_2)^{-1}$ have series expansions and are allowed to appear in the computation. The main body of Xin's algorithm uses partial fraction decomposition and read off their constant terms separately. The algorithm is simple and implemented by the updated Maple package E112. In computing partial fraction decompositions, bottle necks resist for multiple

roots and nonlinear factors in the denominator. We find that these bottle necks can be solved using ideas from computational geometry.

A simpler model has been extensively studied earlier in computational geometry. Let P be as above specified by integral matrix $A = (a_{ij})_{r \times n}$ and nonzero vector b . Then it is well-known that the generating *polynomial* for $P \cap \mathbb{Z}^n$ can be written as a constant term:

$$E(x; A, b) = \sum_{\alpha \in P \cap \mathbb{Z}^n} x_1^{\alpha_1} \cdots x_n^{\alpha_n} = \text{CT}_{\Lambda} \frac{\lambda_1^{-b_1} \cdots \lambda_r^{-b_r}}{\prod_{j=1}^n (1 - \lambda_1^{a_{1,j}} \lambda_2^{a_{2,j}} \cdots \lambda_r^{a_{r,j}} x_j)}. \quad (2)$$

So this is just the special case of (1) when the numerator L is a monomial. Geometers are more interested with the specialization of $x_i = 1$ for all i , which gives the number of lattice points in P . The most important result in this field is due to Barvinok, who developed a polynomial algorithm when the dimension is fixed [6] in 1994. Barvinok's algorithm is hard and was implemented by the `LattE` package by Leora et al. [13] in 2004. An improvement was given in [11]. The readers are referred to [13] for related references and [7] for related applications. Xin's algorithm is not polynomial, but the `E112` package has better performance than the `LattE` package in some situations. The two algorithms are very different in nature. See Section 2. We find that we can do better if combine beautiful ideas of the two algorithms.

The paper is organized as follows. Section 1 is this introduction. Our ultimate goal is to develop a classic algorithm for this subject in the near future. Section 2 introduces and compares the ideas of Barvinok's polynomial algorithm and Xin's partial fraction algorithm. Section 3 includes one of the two major contributions in this paper. We extend Barvinok's algorithm for the multivariate case, which give rise to a polynomial algorithm for MacMahon partition analysis. We do not give implementation of this algorithm since it is not a easy task and there are much room for improvements. Section 4 includes the other major contribution. We develop a Euclid style algorithm with an easy implementation by the Maple package `CTEuclid`. This algorithm performs well and solves several bottle-neck problems in the `E112` package. In Section 5, we give an introduction for `CTEuclid` with concrete examples for the sake of clarity. We explain the flexibility of our algorithm and our strategy for benchmark problems. As an application, we give the first solution for the generating function of order 6 magic squares.

2 Comparison of Barvinok's polynomial algorithm and Xin's partial fraction algorithm

Barvinok's algorithm is in the view of computational geometry and Xin's algorithm is along the line of MacMahon partition analysis in algebraic combinatorics. In this section we compare the two algorithms and conclude that a better strategy is to combine the nice ideas of the two algorithms. It is better to give a brief description of the two algorithms. We follow notations in the introduction.

Barvinok's algorithm computes the generating polynomial $E(x; A, b)$ as a sum of *rational functions* and then evaluate at $x_i = 1$ for all i . The main body of the algorithm can be summarized in the following theorem.

Theorem 1 (Barvinok). *There is a polynomial time algorithm to write $E(x; A, 0)$ as a sum of N simple rational functions, where N is a polynomial in the bytes of A .*

The theorem can be shown by three facts. 1) Rational cones are signed-decomposed into simplicial cones in polynomial time. 2) Barvinok made a *key observation* that a simplicial cone can be decomposed in polynomial time to unimodular simplicial cones. 3) The generating function for a unimodular simplicial cone is simple.

Algorithm Barvinok for computing $E(x; A, b)|_{x_i=1}$:

1. By using Brion's theorem, we can write

$$E(x; A, b) = \sum_i x^{\nu^{(i)}} E(x; A^{(i)}, 0),$$

where the summands corresponds to vertex cones and the sum ranges over all vertices $\nu^{(i)}$ of P .

2. Apply Theorem 1 for each $E(x; A^{(i)}, 0)$.
3. Finally, take limits when $x_i \rightarrow 1$ for all i , as we shall discuss in Section 3.

The readers are referred to [6] and [13] for detailed explanation.

Xin's algorithm computes the constant term of an Elliott rational function in (1) regarded as an iterated Laurent series. The main body of the algorithm is to take constant term in a single variable.

Algorithm XinPF

1. For each Elliott-rational functions as a summand, successively choose a variable and take the constant term as in the next step. Note that after specifying a working field, we are taking constant term in a set of variables.
2. When taking constant term in λ of an Elliott-rational function, computing the partial fraction decomposition and read off the constant term in λ .

The comparison

From the above descriptions, we see that the two algorithms are very different.

1. The basic elements of Barvinok's algorithm is rational cones while that of Xin's algorithm is Elliott rational functions, which is a larger class of objects.
2. By performing on rational cones, Barvinok's algorithm avoids the convergence problem. Xin's algorithm settling the convergence problem by introducing the field of iterated Laurent series as a framework.

3. Barvinok's algorithm is polynomial while Xin's is not.
4. Xin's algorithm deals with the numerator uniformly while Barvinok' does not.
5. The application of partial fraction decompositions has much more flexibility than rational cone decompositions under our framework.

Although Barvinok's algorithm is polynomial, the application of Brion's theorem maybe very costly if the number of vertices is large. Indeed, Xin's `E112` package has better performance than the `LattE` package in many situations, such as the `Sdd5` problem. Because of the freedom of Xin's algorithm, we conclude that a better strategy is to embed some of Barvinok's ideal to Xin's frame work. This leads to a polynomial algorithm for MacMahon partition analysis in Section 3, and a Euclid style algorithm in Section 4, which solves two bottle neck problems in the step of partial fraction decomposition.

3 A polynomial algorithm for MacMahon partition analysis in theory

Let us describe the heart problem of this paper more clearly as follows. Given an Elliott rational function $E = E(x_1, x_2, \dots, x_m)$ in the form

$$E = \frac{L}{(1 - M_1)(1 - M_2) \cdots (1 - M_n)},$$

where L is a Laurent polynomial and M_i are monomials. Let $\{\lambda_1, \dots, \lambda_r\}$ be a subset of $\{x_1, \dots, x_m\}$. We need to compute the constant term of E

$$\text{CT}_{\lambda_1, \dots, \lambda_r} E = \text{a simple representation free of the } \lambda\text{'s,}$$

when E is regarded as an iterated Laurent series.

3.1 The big picture

For simplicity we assume E has the natural series expansion in this section. If use terms in the next section, E is called in its proper form. To clarify the situation, write $M_j = M_j(x)\Lambda^{c_j}$ where $\Lambda^a = \lambda_1^{a_1}\lambda_2^{a_2} \cdots \lambda_r^{a_r}$ and $M_j(x)$ is free of Λ . Let $A = (c_1, \dots, c_n)$ be the matrix with the j th column c_j . Assume the rank of A is r so that the solution space of $A\alpha = 0$ has dimension $d = n - r$. *Polynomial time* means that the running time is polynomial in the bytes $O(nr \log(\max A))$ of A , and the bytes of L , when the dimension d is fixed. Since the bytes of L is linear in the number of terms of L , we may assume L is a monomial.

The big picture of the algorithm is as follows. We are indeed extending Barvinok's algorithm to the multivariate case.

1. Add slack variables z_1, \dots, z_n so that

$$E' = \frac{L}{\prod_{j=1}^n (1 - z_j M_j(x) \Lambda^{c_j})}, \quad E'|_{z_i=1} = E.$$

This step is easy but necessary in the general situation.

2. Eliminate λ_i for all i to get a big sum of polynomial size. For instance, assume L is a monomial and write $Z_j = z_j M_j(x)$. Then $\text{CT}_\Lambda E'$ corresponds to the generating function for lattice points in a rational convex polytope. Applying Brion's theorem and then applying Theorem 1 will give a sum of N' terms where N' is polynomial in the bytes of A .
3. Eliminate all the slack variables z_i . This can also be done in polynomial time, as we shall elaborate next.

Note that the proposed second step is in the spirit of Barvinok's algorithm. It has much more room for improvements.

3.2 Eliminating the slack variables

Algorithm for eliminating the slack variables could be think as an independent subject, so let us clarify the problem. Our input is a rational function $Q(x_1, \dots, x_m; z_1, \dots, z_n)$ written as a (big) sum of simple rational functions:

$$Q(x_1, \dots, x_m; z_1, \dots, z_n) = \sum \frac{\text{simple numerator}}{\text{simple denominator}}.$$

Our output will be a representation of $Q(x_1, \dots, x_m; 1, \dots, 1)$, which is known in priori to be well defined. The z 's are called the *slack variables*. In particular the output is a number for $m = 0$ and a single rational function for $m = 1$. We do not hope a single rational function for $m \geq 2$.

It is not clear how to eliminate z_i even when $m = 0$: i) combining the terms to a single rational function will get a monster numerator that can not be handled by the computer; ii) direct substitutions of $z_i = 1$ for all i does not work for possible denominator factors like $1 - z_1 z_2$ in some of the terms.

The algorithm we present here is inspired by the work from computational geometry: the $m = 0$ case was given a nice solution as a basic step in Barvinok's algorithm for lattice point counting. Though the idea works for general rational functions, we shall concentrate on Elliott-rational functions, that is

$$Q = \sum_i \frac{L_i(x; z)}{(1 - M_{i1} z^{B_{i1}})(1 - M_{i2} z^{B_{i2}}) \cdots (1 - M_{id} z^{B_{id}})},$$

where M_{ij} are monomials in x , L_i are Laurent polynomials. Such rational functions arose from MacMahon partition analysis. In particular, the $m = 1$ case corresponds to the computation of Ehrhart series.

The algorithm consists of two steps: First reduce the number of slack variables to 1, and then use Laurent series expansion to compute separately for each term.

Reduce the number of slack variables to 1. Calculating $Q(x; 1, 1, \dots, 1)$ is equivalent to evaluating the limit as z_i goes to 1 for all i . Our first step is to reduce the number of slack variables to 1. This is done by finding a suitable integer vector λ and making the substitution $z_i \rightarrow t^{\lambda_i}$. In order to do so, λ must be picked such that there is no zero denominator in any term, i.e., for every i and j we can not have both $M_{ij} = 1$ and the inner product $\langle \lambda, B_{ij} \rangle = 0$. Barvinok showed such λ can be picked in polynomial time by choosing points on the moment curve. Loera et al. [13] suggested to use random vectors to avoid large integer entries.

Using Laurent series expansion. Now we need $Q(x; t^{\lambda_1}, \dots, t^{\lambda_n}) \Big|_{t=1}$, where

$$Q(x; t^{\lambda_1}, \dots, t^{\lambda_n}) = \sum_i \frac{L_i(x; t^{\lambda_1}, \dots, t^{\lambda_n})}{\prod (1 - t^{\langle \lambda, B_{ij} \rangle} M_{ij})}.$$

An obvious way is to make the substitution $t = 1 + s$. Then we have

$$Q \Big|_{t=1} = Q(x; (1+s)^{\lambda_1}, \dots, (1+s)^{\lambda_n}) \Big|_{s=0} = \text{CT}_s Q(x; (1+s)^{\lambda_1}, \dots, (1+s)^{\lambda_n}),$$

where we are taking constant term of a Laurent series in s . The linearity of the constant term operator allows us to compute separately:

$$Q(x; 1, \dots, 1) = \sum_i \text{CT}_s \frac{L_i(x; (1+s)^{\lambda_1}, \dots, (1+s)^{\lambda_n})}{\prod (1 - (1+s)^{\langle \lambda, B_{ij} \rangle} M_{ij})}.$$

The substitution $t = 1 + s$ seems natural and works fine for the $m = 0$ case, where we only need do polynomial division of a single variable. For $m \geq 1$, we find it better to make the exponential substitution $t = e^s$, which leads to

$$Q(x; 1, \dots, 1) = \sum_i \text{CT}_s \frac{L_i(x; e^{\lambda_1 s}, \dots, e^{\lambda_n s})}{\prod (1 - e^{s \langle \lambda, B_{ij} \rangle} M_{ij})}.$$

Proposition 2. *Let $L(x; z)$ be a Laurent polynomial, M_j monomials in x and b_j integers. The constant term*

$$\text{CT}_s \frac{L(x; e^{\lambda_1 s}, \dots, e^{\lambda_n s})}{\prod_{j=1}^d (1 - e^{b_j s} M_j)}$$

can be efficiently computed as a sum of at most $\binom{d+1}{\lfloor d/2 \rfloor}$ simple rational functions.

Proof. By rearranging the denominators, we may assume that $M_1 = M_2 = \dots = M_r = 1$ and all the other M_j are not 1. It is easy to see that

$$s^r \frac{L(x; e^{\lambda_1 s}, \dots, e^{\lambda_n s})}{\prod_{j=1}^d (1 - e^{b_j s} M_j)}$$

is a power series in s , so that we are indeed taking coefficient of s^r in a power series. To expand, we only need the following two formulas:

$$\frac{s}{1 - e^s} = \sum_{n \geq 0} -\frac{\mathcal{B}_n}{n!} s^n = -1 + \frac{1}{2}s - \frac{1}{12}s^2 + \frac{1}{720}s^4 - \frac{1}{30240}s^6 + \frac{1}{1209600}s^8 + \dots,$$

$$\frac{1}{1 - e^s M} = \frac{1}{(1 - M)(1 - \frac{M}{1-M}(e^s - 1))} = \sum_{n \geq 0} \frac{M^n}{(1 - M)^{n+1}} (e^s - 1)^n = \sum_{n \geq 0} c_n(M) s^n.$$

The \mathcal{B}_n are the well-known Bernulli numbers. The computation

$$\prod_{j=1}^r \frac{s}{1 - e^{b_j s}} = \prod_{j=1}^r \sum_{n \geq 0} -\frac{\mathcal{B}_n b_j^{n-1}}{n!} s^n = \sum_{n=0}^r c'_n s^n + \text{higher degree terms}$$

is easy since this only involves univariate polynomial multiplication. Thus we can write

$$\begin{aligned} s^r \frac{L(x; e^{\lambda_1 s}, \dots, e^{\lambda_n s})}{\prod_{j=1}^d (1 - e^{b_j s} M_j)} &= \sum_{n \geq 0} \ell_n(x) s^n \cdot \sum_{n \geq 0} c'_n s^n \prod_{j=r+1}^d c_n(M_j) b_j^{n_j} \\ &= \sum_{n \geq 0} s^n \sum_{n_0 + n_1 + n_{r+1} + \dots + n_d = n} \ell_{n_0}(x) c'_{n_1} \prod_{j=r+1}^d c_{n_j}(M_j) b_j^{n_j}. \end{aligned}$$

It follows that the desired constant term is a sum of $\ell_{n_0}(x) c'_{n_1} \prod_{j=r+1}^d c_{n_j}(M_j) b_j^{n_j}$. The number of terms we obtained is equal to $\binom{d+1}{r}$, which is the number of nonnegative integer solutions of $n_0 + n_1 + n_{r+1} + \dots + n_d = r$.

The proposition follows since $\binom{d+1}{r}$ reaches maximum at $r = \lceil d/2 \rceil$. \square

Using the exponential substitution, we only need two formulas for series expansion and we can store in advance those coefficients for $n \leq d$. This is much better than using the substitution $t = 1 + s$, especially for $m \geq 1$. In the $m = 1$ case, we can expect to obtain a single rational functions, since the computer performs well on univariate rational functions.

The only problem for this approach is the large integer problem. It seems unavoidable that some of the $b_{ij} = \langle \lambda, B_{ij} \rangle$ might be large. This results in huge numbers since our formula involves b_{ij}^n for $n \leq d$. This problem is avoidable by modular a reasonably large prime number ($> d$), provided that we know some information of the final output in advance. This is indeed the case since our problems are usually combinatorial and the final output are nice in some sense. The $m = 0$ case problem usually computes the number of lattice points in a rational convex polytope. There are methods to estimate this number. For the $m = 1$ case we usually compute the Ehrhart series as in the next subsection. Such generating functions seems always have simple denominator and its numerator has not too large integer coefficients. Thus we can do the constant term extraction modular a chosen large prime. If necessary, we can do the computation several times using different large primes and then use the Chinese remainder theorem to construct the final output.

3.3 Direct computation of the Ehrhart series

Given a rational polytope $P = \{\alpha : A\alpha = b, \alpha \geq 0\} \subset \mathbb{R}^n$, the function

$$i_P(k) := \#(kP \cap \mathbb{Z}^n) = \#\{\alpha \in \mathbb{Z}^n : A\alpha = kb, \alpha \geq 0\}$$

defined for positive integer k was first studied by E. Ehrhart [8]. It is call the Ehrhart polynomial when the vertices of P are integral and is call the Ehrhart quasi-polynomial for arbitrary rational polytopes [17, Ch. 4]. For us it is easier to describe using generating functions. The Ehrhart series of P defined by

$$I_P(q) = \sum_{k \geq 0} i_P(k)q^k$$

is an Elliott-rational function. It has close connection with Hilbert series for some graded algebra.

An important problem is to compute the Ehrhart quasi-polynomial of given P . Earlier method is to compute $i_P(k)$ for sufficiently many k and then use the Lagrange interpolation formula to construct $i_P(k)$. We can compute the Ehrhart series directly by the following constant term representation.

$$I_P(q) = \underset{\Lambda}{\text{CT}} \frac{1}{\prod_{j=1}^n (1 - \lambda_1^{a_{1,j}} \lambda_2^{a_{2,j}} \cdots \lambda_r^{a_{r,j}} x_j)} \times \frac{1}{1 - q\lambda_1^{-b_1} \cdots \lambda_r^{-b_r}} \Big|_{x_j=1}.$$

This corresponds to a rational cone or the homogeneous system $(A, -b)\beta = 0$. This leads to a combined way for Ehrhart series computation: Use `LattE` to do the rational cone decomposition and then use our way of eliminating the slack variables. There is no implementation for this approach yet. We remark that a similar idea was proposed in [13] to avoid the use of Brion's theorem. But they only wish to compute $i_P(k)$ for particular k , or equivalently, compute $i_P(1)$.

Many benchmark problems are related to the computation of Ehrhart series. The counting of magic squares and its variations is one of the common topic in both combinatorics and computational geometry. The definition of magic squares are different in different literature. Here an n by n nonnegative integer matrix $M = (a_{i,j})_{n \times n}$ is said to be a magic squares with magic sum m if its row sum, column sum, and two diagonal sum are all equal to m . That is, the order n magic square polytope MS_n is defined by the following linear constraints:

$$\begin{aligned} a_{i,1} + a_{i,2} + \cdots + a_{i,n} &= 1, \text{ for } 1 \leq i \leq n \\ a_{1,j} + a_{2,j} + \cdots + a_{n,j} &= 1, \text{ for } 1 \leq j \leq n \\ a_{1,1} + a_{2,2} + \cdots + a_{n,n} &= 1, \quad a_{n,1} + a_{n-1,2} + \cdots + a_{1,n} = 1. \end{aligned}$$

The determination of $I_{MS_n}(q)$ is known for $n = 3$, and for $n = 4$. Many algorithms meet trouble for the $n = 5$ case. See e.g., [2]. The first solution for the order 5 magic squares was reported in [12].

Our approach is along the line of MacMahon partition analysis. Let λ_i index the i -th row equation, let μ_j index the j -th column equation, and let ν_1 and ν_2 index the two diagonal equations. Then it is not hard to see that

$$\sum_{m=0}^{\infty} \sum_M \prod_{1 \leq i, j \leq n} x_{i,j}^{a_{i,j}} q^m = \text{CT}_{\lambda, \mu, \nu} F_n(x, q; \lambda, \mu, \nu),$$

where the second sum ranges over all magic squares M with magic sum m , and

$$F_n(x, q; \lambda, \mu, \nu) = \prod_{1 \leq i, j \leq n} \frac{1}{1 - x_{i,j} \lambda_i \mu_j \nu_1^{\chi(i=j)} \nu_2^{\chi(i+j=n+1)}} \frac{1}{1 - q(\lambda_1 \cdots \lambda_n \mu_1 \cdots \mu_n \nu_1 \nu_2)^{-1}}.$$

In particular, setting $x_{i,j} = 1$ gives the generating function for $I_{MS_n}(q)$.

$$I_{MS_n}(q) = \text{CT}_{\lambda, \mu, \nu} \prod_{1 \leq i, j \leq n} \frac{1}{1 - \lambda_i \mu_j \nu_1^{\chi(i=j)} \nu_2^{\chi(i+j=n+1)}} \times \frac{1}{1 - q(\lambda_1 \cdots \lambda_n \mu_1 \cdots \mu_n \nu_1 \nu_2)^{-1}}.$$

Though we believe that the order 6 magic squares problem should be computed fast by the proposed approach here, we would like to introduce a more elementary approach in the next section.

4 A Euclid style algorithm for MacMahon partition analysis

In the big picture of subsection 3.1, we mentioned that the second step may have different approaches. The proposed method is not easy to implement and is not ideal. In this section we provide a Euclid style algorithm. This algorithm is elementary, deal with the numerator uniformly and perform well in practice.

The Euclid style algorithm is along the line of Xin's partial fraction algorithm, so it is time to explain briefly the field of iterated Laurent series. We use the list $\mathbf{vars} = [x_1, x_2, \dots, x_n]$ to define the working field $K = Q((x_n))((x_{n-1})) \cdots ((x_1))$. See [18] for detailed explanation. Here we only need the fact that every monomial $M \neq 1$ is comparable with 1 in K by the following rule: find the smallest variable x_j appear in M , so $\deg_{x_i} M = 0$ for all $i < j$. If $\deg_{x_j} M > 0$ then we say M is small, denoted $M < 1$, otherwise we say M is large, denoted $M > 1$. Thus we can determine which of the following two series expansion holds in K .

$$\frac{1}{1 - M} = \begin{cases} \sum_{m \geq 0} M^m, & \text{if } M < 1; \\ \frac{1}{-M(1 - 1/M)} = \sum_{m \geq 0} -\frac{1}{M^{m+1}}, & \text{if } M > 1. \end{cases}$$

When expanding E as a series in K , we usually rewrite E in its *proper form*:

$$E = \frac{L}{(1 - M_1)(1 - M_2) \cdots (1 - M_n)},$$

where L is a Laurent polynomial and $M_i < 1$ for all i . Note that the proper form of E is not unique. For instance $1/(1-x) = (1+x)/(1-x^2)$ are both proper form.

Algorithm CTEuclid for the second step in subsection 3.1.

2-1 For each Elliott-rational functions as a summand, successively choose a variable, say $x = x_\ell$, and take the constant term in x . Thus we reduce to taking constant term in a single variable.

2-2 Reduce $\text{CT}_x E$ as a sum of contributions of single denominator factors.

2-3 The contribution of a single denominator factor can be recursively computed.

Both last two steps contains some new features.

4.1 A reduction to the contribution of a single factor

To take constant term in x , we shall write

$$E = \frac{L(x)}{\prod_{i=1}^n (1 - u_i x^{a_i})}, \quad (3)$$

where $L(x)$ is a Laurent polynomial, u_i are free of x and a_i are positive integers. We assume the denominator factors $1 - u_i x^{a_i}$ are coprime to each other. This is guaranteed by the first step in subsection 3.1.

The partial fraction decomposition of E in x should be

$$E = P(x) + \frac{p(x)}{x^k} + \sum_{i=1}^n \frac{A_i(x)}{1 - u_i x^{a_i}}, \quad (4)$$

where $P(x), p(x), A_i(x)$ are all polynomials and $\deg p(x) < k, \deg A_i(x) < a_i$. If written in proper form, we shall have

$$\frac{A_i(x)}{1 - u_i x^{a_i}} = \begin{cases} \frac{A_i(x)}{1 - u_i x^{a_i}}, & \text{if } u_i x^{a_i} < 1; \\ \frac{A_i(x)}{-u_i x^{a_i} (1 - \frac{1}{u_i x^{a_i}})} = \frac{x^{-a_i} A_i(x)}{-u_i (1 - \frac{1}{u_i x^{a_i}})}, & \text{if } u_i x^{a_i} > 1. \end{cases}$$

Since $x^{-a_i} A_i(x)$ contains only negative powers of x , we obtain

$$\text{CT}_x E = P(0) + \sum_{u_i x^{a_i} < 1} A_i(0), \quad (5)$$

where the sum ranges over all i such that $u_i x^{a_i}$ is small. For this reason, the denominator factor $1 - u_i x^{a_i}$ is said to be contributing if $u_i x^{a_i}$ is small and is said to be not contributing if $u_i x^{a_i}$ is large.

It will be convenient to use the following notation.

$$\langle E, 1 - u_i x^{a_i} |_x = \text{CT}_x \frac{1}{\underline{1 - u_i x^{a_i}}} E(1 - u_i x^{a_i}) = A_i(0).$$

One can think that when taking constant term in x , only the single underlined denominator factor contributes. In this view it should be clear that $\langle E, 1 - u x^a |_x = 0$ if the denominator of E is coprime to $1 - u x^a$. Thus (5) can be rewritten as

$$\text{CT}_x E = P(0) + \sum_i \chi(u_i x^{a_i} < 1) \langle E, 1 - u_i x^{a_i} |_x, \quad (6)$$

where $\chi(S)$ is 1 if the statement S is true and 0 if S is false.

Next we show that we need not compute $P(0)$. Note that $P(x)$ could be computed by polynomial division. But the polynomial division algorithm is very expensive in our situation because the number of the denominator factors and the number of variables may be large. Our first observation is that $P(x) = 0$ if E is *proper* in x , i.e., the degree in the numerator is less than the degree in the denominator. To see that we can avoid computing $P(0)$ in general, we need the following lemma.

Lemma 3. *Let E be as in (3). If E is a proper rational function, then*

$$\text{CT}_x E = \sum_{i=1}^n \chi(u_i x^{a_i} < 1) \langle E, 1 - u_i x^{a_i} |_x; \quad (7)$$

If $E|_{x=0}$ exists, then

$$\text{CT}_x E = E|_{x=0} - \sum_{i=1}^n \chi(u_i x^{a_i} > 1) \langle E, 1 - u_i x^{a_i} |_x. \quad (7')$$

Proof. The first equality is obvious. For the second equality, since $E|_{x=0}$ exists, $p(x)$ must be 0 in (4). Now setting $x = 0$ gives

$$E|_{x=0} = P(0) + \sum_i A_i(0) = \text{CT}_x E + \sum_{i=1}^n \chi(u_i x^{a_i} > 1) \langle E, 1 - u_i x^{a_i} |_x.$$

The lemma then follows. \square

Formula (7') is a kind of dual of (7). For this reason, we also call the denominator factor $1 - u_i x^{a_i}$ with $u_i x^{a_i} > 1$ dually contributing. If E is proper and has no pole at $x = 0$ then both formulas (7) and (7') applies and we can choose the simpler one.

Now we are ready to reduce the computation of $\text{CT}_x E$ to the contribution from a single denominator factor. Split $L(x)$ as $L_1(x) + L_2(x)$, where L_1 contains only positive powers in x and L_2 contains only nonpositive powers in x . Now E is written as $E_1 + E_2$ where E_j , $j = 1, 2$, are similar to E except that $L(x)$ is replaced by $L_j(x)$. Clearly $E_1|_{x=0} = 0$ and E_2 is proper. It follows from Lemma 3 that

$$\text{CT}_x E = \sum_i \chi(u_i x^{a_i} < 1) \langle E_2, 1 - u_i x^{a_i} |_x - \sum_i \chi(u_i x^{a_i} > 1) \langle E_1, 1 - u_i x^{a_i} |_x. \quad (8)$$

4.2 A recursion for the contribution of a single factor

The contribution of a linear factor is easy:

$$\langle E, 1 - ux \rangle_x = \text{CT}_x \frac{1}{1 - ux} E(1 - ux) = E(1 - ux)|_{x=1/u}.$$

However, effective computation for nonlinear factor was a long standing problem. One can factor $1 - ux^a$ into linear factors using roots of unities, but there is no simple way to get rid of the roots of unities in the final outcome. We present here a Euclid-style algorithm dealing with the nonlinear case, by repeated application of the following recursion.

Proposition 4. *Let E be as in (3) and let $1 - ux^a$ be a denominator factor. Then we can find E' such that*

$$\langle E, 1 - ux^a \rangle_x = - \sum_i \langle E', 1 - v_i x^{b_i} \rangle_x,$$

where $0 < b_i \leq a/2$ for all i and the number of terms is at most $n - 1$.

Proof. Without loss of generality, we may assume $ux^a = u_1 x^{a_1}$. We need to compute

$$\langle E, 1 - ux^a \rangle_x = \text{CT}_x \frac{1}{1 - ux^a} \frac{L(x)}{\prod_{i=2}^N (1 - u_i x^{a_i})} = A_1(0).$$

The following characterization of $A_1(x)$ is well-known:

$$A_1(x) \equiv E(1 - ux^a) \pmod{\langle 1 - ux^a \rangle}.$$

This is simply obtained by multiplying both sides of (4) by $(1 - ux^a)$ and then modulo the ideal $\langle 1 - ux^a \rangle$ generated by $1 - ux^a$. The $A_1(x)$ is the unique polynomial representation satisfying $\deg A_1(x) < a$.

To compute $A_1(x)$, we can change the terms by their simpler representations and finally take remainder when dividing by $1 - ux^a$. The following clearly holds,

$$x^m \equiv u^{-\ell} x^r \pmod{\langle 1 - ux^a \rangle} \text{ if } m = \ell a + r.$$

Particularly, the *remainder* $\text{rem}(x^m, 1 - ux^a, x)$ and the *signed remainder* $\text{srem}(x^m, 1 - ux^a, x)$ of x^m when dividing by $1 - ux^a$ is defined to be

$$\text{rem}(x^m, 1 - ux^a, x) = u^{-\ell} x^r, \text{ where } m = \ell a + r, 0 \leq r < a. \quad (9)$$

$$\text{srem}(x^m, 1 - ux^a, x) = u^{-\ell} x^r, \text{ where } m = \ell a + r, -a/2 < r \leq a/2. \quad (10)$$

These definitions linearly extends for Laurent polynomials.

The new idea is that $A_1(0)$ can be cracked out by using a *better* representative of $A_1(x) + \langle 1 - ux^a \rangle$ instead of the explicit formula of $A_1(x)$. Clearly we have

$$A_1(x) \equiv \frac{L(x)}{\prod_{i=2}^n (1 - u_i \text{srem}(x^{a_i}, 1 - ux^a, x))} \pmod{\langle 1 - ux^a \rangle}.$$

This can be rewritten in the following form:

$$A_1(x) \equiv \frac{\pm ML(x)}{\prod_{i=2}^n (1 - v_i x^{b_i})} \pmod{\langle 1 - ux^a \rangle},$$

where M is a monomial and $0 \leq b_i \leq a/2$ for all i .

Now we have to split into two cases:

i) if all the b_i are 0, then we immediately obtain

$$A_1(x) = \frac{1}{\prod_{i=2}^n (1 - v_i)} \operatorname{rem}(\pm ML(x), 1 - ux^a, x)$$

and $A_1(0)$ can be easily computed.

ii) if at least one of b_i is great than 0, then rewrite

$$A_1(x) \equiv \frac{xL'(x)}{\prod_{i=2}^n (1 - v_i x^{b_i})} \pmod{\langle 1 - ux^a \rangle},$$

where

$$L'(x) = \pm \operatorname{rem}(x^{-1}ML(x), 1 - ux^a)$$

is a polynomial in x of degree less than a . Now comes the crucial observation:

$$A_1(0) = \operatorname{CT}_x \frac{1}{1 - ux^a} \frac{xL'(x)}{\prod_{i=2}^n (1 - v_i x^{b_i})}.$$

In our notation, this is just

$$\langle E, 1 - ux^a |_x = \langle E', 1 - ux^a |_x, \quad (11)$$

where

$$E' = \frac{1}{1 - ux^a} \frac{xL'(x)}{\prod_{i=2}^n (1 - v_i x^{b_i})}$$

is a proper rational function with $E'|_{x=0} = 0$. It follows by the partial fraction decomposition of E' and then setting $x = 0$ that

$$\langle E', 1 - ux^a |_x = - \sum_{i=2}^n \langle E', 1 - v_i x^{b_i} |_x. \quad (12)$$

Note that the terms for $b_i = 0$ vanish. The proposition then follows. \square

The Ell package in [18] computes $A_1(0)$ by first finding $A_1(x)$ and then setting $x = 0$. This is not a good strategy when the number of variables or n is large. The explicit formula of $A_1(x)$ may be very huge. To see this, consider the polynomial representative of $1 - vx^b$ for b coprime to a . We have

$$\frac{1}{1 - vx^b} = \frac{x^{-ab}(x^{ab}u^b - 1 + 1 - x^{ab}v^a)}{(u^b - v^a)(1 - vx^b)} = \frac{x^{-ab}(1 - (vx^b)^a)}{(u^b - v^a)(1 - vx^b)} - \frac{x^{-ab}(1 - (ux^a)^b)}{(u^b - v^a)(1 - vx^b)}.$$

Now the second term vanishes when modulo $1 - ux^a$, so we are left with

$$\frac{1}{1 - vx^b} \equiv \frac{x^{-ab}}{u^b - v^a} \sum_{j=0}^{a-1} (vx^b)^j \pmod{\langle 1 - ux^a \rangle}.$$

The $\gcd(a, b) > 1$ case needs a bit more work. See [18] for detailed information.

Now take $a = 3$ as an example. If $n = 21$ and all the other 20 a_i satisfying $\gcd(a, a_i) = 1$, then the formula for $A_1(x)$ will involve 20 three-term-factors $(1 + u_i x^{a_i} + u_i^2 x^{2a_i})$. No miracles will happen if there are more than 3 variables, and $A_1(x)$ usually contains at least about 3^{20} terms. While if we use Proposition 4, then $b_i = 1$ for all i and we only obtain 20 simple rational functions.

Repeated application of Proposition 4 will give a sum of simple rational functions. The number of terms only depends on a_i and the process is similar to Euclid's gcd algorithm. Denote this number by $f(i; a_1, a_2, \dots, a_n)$ where a corresponds to a_i . Then $f(i; a_1, a_2, \dots, a_n)$ is recursively determined by the following rules:

1. If $a_j = 0$ for all $j \neq i$ then $f(i; a_1, a_2, \dots, a_n) = 1$;
2. We have $f(i; a_1, a_2, \dots, a_n) = f(i; b_1, b_2, \dots, b_n)$, where $b_i = a_i$ and $b_j = \min(\text{rem}(a_j, a_i), a_i - \text{rem}(a_j, a_i))$ for $j \neq i$.
3. If $a_j \leq a_i/2$ for all $j \neq i$, then

$$f(i; a_1, a_2, \dots, a_n) = \sum_{j \neq i} f(j, a_1, a_2, \dots, a_n).$$

If $n = 1$ then $f(1; a_1) = 1$. If $n = 2$ we also have $f(i; a_1, a_2) = 1$, because the sum of the recursion contains a single term. For $n = 3$, computation evidence suggests that $f(1; a_1, a_2, a_3)$ is almost $O(\log a_1)^2$. For larger n , we raise the following problem:

Let $f(i; a_1, a_2, \dots, a_n)$ be defined as above. Prove or disprove that $f(i; a_1, a_2, \dots, a_n)$ is a polynomial in $\log a_i$.

If the answer is positive, then we will obtain a simple polynomial algorithm at least for one variable elimination. But this might not be the right problem, since we have much freedom to apply the partial fraction technique, and the current approach is too elementary.

5 The Maple package CTEuclid

The algorithm in Section 4 is implemented by the Maple package `CTEuclid`, which can be downloaded from the following link

<https://www.dropbox.com/sh/scepodyyn4ff7ro/ffhqmeN7ne/MPA>,

where two demo files are provided to explain how to use the package. One file works on magic squares of order up to 5 and the other file works on the Sdd problem [9] of order up to 5. Both files contain the essential idea for attacking the order 6 case. Here we only report the Ehrhart series for magic squares of order 6.

CTEuclid is the first package designed for complicated or even benchmark problems. It also performs well for simple problems. The algorithm has three main steps with the second step split into 3 small steps, as described in Section 4.1. For the sake of clarity, we explain by several Knapsack type examples.

5.1 Knapsack type examples

Let a_0, a_1, \dots, a_n be positive integers with $a = (a_1, \dots, a_n)$, $\gcd(a_1, \dots, a_n) = 1$ and $a_i \leq a_0$ for all i , and let

$$P = \{x \in \mathbb{R}^n : ax = a_0, x \geq 0\}.$$

A basic problem is to determine if P contains an integer vector, or how many integer vectors does P contain. The former is called integer programming feasibility problem. See [1] for an introduction on this topic. Here we concentrate on the second problem, which is also called Knapsack type problems. Clearly we have

$$\#P = [x^{a_0}] \frac{1}{(1-x^{a_1}) \cdots (1-x^{a_n})} = \text{CT}_x \frac{1}{x^{a_0} (1-x^{a_1}) \cdots (1-x^{a_n})}.$$

Example 5. *Compute the following constant term:*

$$\text{CT}_x \frac{1}{x^{41} (1-x)(1-x^5)(1-x^{14})}.$$

Solution. We first add slack variables and get

$$\text{CT}_x E = \text{CT}_x \frac{1}{x^{41} (1-xz_1)(1-x^5z_2)(1-x^{14}z_3)} = \text{CT}_x \frac{1}{x^{41} \underline{(1-xz_1)} \underline{(1-x^5z_2)} \underline{(1-x^{14}z_3)}},$$

where the three underlined factors are contributing. For the last factor we have

$$\begin{aligned} \text{CT}_x \frac{1}{x^{41} (1-xz_1)(1-x^5z_2) \underline{(1-x^{14}z_3)}} &= \text{CT}_x \frac{xz_3^3}{(1-xz_1)(1-x^5z_2) \underline{(1-x^{14}z_3)}} \\ &= -\text{CT}_x \frac{xz_3^3}{\underline{(1-xz_1)} \underline{(1-x^5z_2)} \underline{(1-x^{14}z_3)}}, \end{aligned}$$

where in our notation, only the first two factors are contributing. The flexibility of our algorithm allows us to obtain the following combined form:

$$\begin{aligned} \text{CT}_x E &= \text{CT}_x \frac{1}{x^{41} \underline{(1-xz_1)} \underline{(1-x^5z_2)} \underline{(1-x^{14}z_3)}} + \text{CT}_x \frac{1}{x^{41} (1-xz_1)(1-x^5z_2) \underline{(1-x^{14}z_3)}} \\ &= \text{CT}_x \frac{x^{-41}}{\underline{(1-xz_1)} \underline{(1-x^5z_2)} \underline{(1-x^{14}z_3)}} - \text{CT}_x \frac{xz_3^3}{\underline{(1-xz_1)} \underline{(1-x^5z_2)} \underline{(1-x^{14}z_3)}} \\ &= \text{CT}_x \frac{x^{-41} - xz_3^3}{\underline{(1-xz_1)} \underline{(1-x^5z_2)} \underline{(1-x^{14}z_3)}}. \end{aligned}$$

Now the first contribution is simple:

$$\text{CT}_x \frac{x^{-41} - xz_3^3}{(1 - xz_1)(1 - x^5z_2)(1 - x^{14}z_3)} = \frac{z_1^{41} - z_1^{-1}z_3^3}{(1 - z_1^{-5}z_2)(1 - z_1^{-14}z_3)}.$$

The contribution of the second factor becomes

$$\begin{aligned} & \text{CT}_x \frac{x^{-41} - xz_3^3}{(1 - xz_1)(1 - x^5z_2)(1 - x^{-1}z_3z_2^{-3})} \\ &= \text{CT}_x \frac{x^5z_3^{-1}z_2^{12} - x^2z_3^2z_2^3}{(1 - xz_1)(1 - x^5z_2)(1 - xz_3^{-1}z_2^3)} \\ &= \text{CT}_x \frac{x^5z_2^{12}z_3^{-1} - x^2z_3^2z_2^3}{(1 - xz_1)(1 - x^5z_2)(1 - xz_3^{-1}z_2^3)} \\ &= \frac{z_1^{-5}z_2^{12}z_3^{-1} - z_1^{-2}z_3^2z_2^3}{(1 - z_1^{-5}z_2)(1 - z_1^{-1}z_3^{-1}z_2^3)} + \frac{z_3^4z_2^{-3} - z_3^4z_2^{-3}}{(1 - z_3z_2^{-3}z_1)(1 - z_3^5z_2^{-14})} \end{aligned}$$

Thus we obtain a sum of three terms and come to step 3. We need to make a substitution so that $z_1^{-5}z_2$, $z_1^{-14}z_3$, $z_1^{-1}z_3^{-1}z_2^3$, $z_3^5z_2^{-14}$ are not equal to 1. One choice is $z_1 = 1, z_2 = t, z_3 = t$. Then the constant term becomes

$$\frac{1 - t^3}{(1 - t)(1 - t)} + \frac{t^{11} - t^5}{(1 - t)(1 - t^2)} + \frac{t - t}{(1 - t^{-2})(1 - t^{-9})} = \frac{1 - t^3}{(1 - t)(1 - t)} + \frac{t^{11} - t^5}{(1 - t)(1 - t^2)}.$$

Next we let $t = 1 + s$ and take constant term in s separately to get

$$\begin{aligned} & \text{CT}_s \frac{1 - (1 + s)^3}{s^2} + \text{CT}_s \frac{(1 + s)^{11} - (1 + s)^5}{s^2(2 + s)} \\ &= [s^2](1 - (1 + 3s + 3s^2)) + [s^2](1 + 11s + 55s^2 - (1 + 5s + 10s^2)) \frac{1}{2}(1 - s/2 + s^2/4) \\ &= -3 + [s](6 + 45s)(1/2 - s/4) = -3 + \frac{45}{2} - \frac{6}{4} = 18. \end{aligned}$$

□

Next we consider a relatively complicated example, which is Example 1 of [1].

Example 6. Show that the polytope P contains no integer lattice points, where

$$P = \{x \in \mathbb{R}^3 : 12, 223x_1 + 12, 224x_2 + 36, 671x_3 = 149, 389, 505, x \geq 0\}.$$

Sketch of the Proof. The problem is equivalent to compute the following constant term:

$$\text{CT}_x \frac{1}{x^{149389505} (1 - x^{12223}) (1 - x^{12224}) (1 - x^{36671})}.$$

Our CTEuclid package will give a sum of 10 terms, which reduce by cancelation to a sum of 4 terms. By letting $z_1 = 1, z_2 = t, z_3 = t$ we reach

$$-\frac{t^{12223}}{(t-1)(t^{12223}-1)} + \frac{t^{24446}}{(t-1)(t^{12223}-1)} - \frac{t^{36670}}{(t-1)(t^{24447}-1)} + \frac{t^{12223}}{(t-1)(t^{24447}-1)}.$$

To eliminate the slack variable $t = 1$, we let $t = e^s$ and compute the constant term in s for each term separately. For instance, the first term becomes,

$$\begin{aligned} \text{CT}_s - \frac{e^{12223s}}{(e^s - 1)(e^{12223s} - 1)} &= [s^2] - e^{12223s} \times \frac{s}{(e^s - 1)} \times \frac{s}{(e^{12223s} - 1)} \\ &= [s^2] - (1 + 12223s + \frac{1}{2}12223^2s^2)(1 - s/2 + s^2/12) \\ &\quad \times (1/12223 - s/2 + 12223s^2/12) = -\frac{149365061}{146676}. \end{aligned}$$

The four constant terms sum to 0. This completes the proof. \square

Still from article [1], a very hard instance is the 4 dimensional polytope with

$$a_0 = 89643481, (a_1, \dots, a_5) = (12223, 12224, 36674, 61119, 85569).$$

Aardal and Lenstra can show that P contains no integer vectors in 0.01 seconds while the Branch and Bound method takes more than 8139 seconds. When dealing with this problem, CTEuclid gives 398 terms and returns 0 in about 0.4 seconds. The advantage of our algorithm is that we can compute the number $\#P$ for different a_0 in about the same time. For instance, if $a_0 = 89643481 \times 1001$, CTEuclid still gives 398 terms and returns 94267024658624993843 in about 0.4 seconds. It is worth noting that many of the 398 terms cancels with only 118 terms left. It might be interesting to understand how these terms cancels with each other. We also tried random examples with $100000 \leq a_i \leq 2500000$, the performance is not nice when $n \geq 5$.

The above examples show that even if the final answer is simple, the middle step may give complicated results. We make the following observation: Step 1 takes no time; Step 2 is the most important step, where we hope the number of terms nt we get is small; Step 3 of eliminating the slack variables is the most time consuming step, and its running time is almost linear to nt . This leads to the following two technical treatment when dealing with complicated or even benchmark problems.

1. In step 2, we save some data for later use: the data for every 1000 terms we obtained are saved in different files, the data for all bad denominator factors are saved in a file.
2. The running time for step 3 can be estimated according to the size of the data we saved in step 2.

5.2 Computation for magic squares of order 6

In subsection 3.3 we have convert the Ehrhart series for order n magic squares polytope to a constant term. Applying the package CTEuclid will give the desired Ehrhart series. There is no difficulty for the cases $n = 3, 4$. Indeed the author's E112 package is faster in these two cases but meet memory problem for the $n = 5$ case. Our CTEuclid computes the $n = 5$ case in about 2700 seconds of cpu time. The $n = 6$ case is much

more complicated, and is not known before. The Ehrhart series for order 6 magic square has been put at Sloane's integer sequence website [15, A216039]. It looks like

$$I_{MS_6}(q) = \frac{(1-q)^3 N}{(1-q^3)^5 (1-q^4)^5 (1-q^5)^4 (1-q^6)^6 (1-q^7)^3 (1-q^8)^2 (1-q^9) (1-q^{10})}$$

$$= 1 + 96q + 14763q^2 + 957936q^3 + 33177456q^4 + 718506720q^5 + \dots$$

where

$$N = q^{138} + 99q^{137} + 15057q^{136} + \dots$$

$$+ 21382798694422310755770332936q^{69} + \dots + 15057q^2 + 99q + 1.$$

This result is obtained by modulo three large primes. The total cpu time is about 70×3 days. The author would like to thank his officemates for running these computations on their computers.

For the order 6 magic squares problem, we have to save data for the results obtained in the second step. We split the data into different files, with each file containing 1000 terms. The most time consuming step is the third step, especially when the dimension is large. The good news is that we can estimate the running time. In the third step we need to eliminate the slack variables. When the data files are too large, we must do the computation modulo a large prime. Here we choose $p_1 = 636, 286, 597$ for our first computation. Our estimate time for the third step is about 108 days of cpu time based on the observation that each file takes about 5 minutes.

The flexibility of our algorithm allows us to reduce the number of terms obtained in the second step. The idea is that we can delay the adding of slack variables. That is, we can directly eliminate several variables as long as all terms in the outcome are valid, i.e., with no zero in the denominator. This is done by several tries and we need to control the size of the output. Next we apply our package to each term in the output. In this way, we reduced the size of the data files by one third and now the running time for the third step is about 70 days. These data can be reused for different modulo computations. Indeed we also did the computation modulo $p_2 = 460, 710, 223, 302, 903, 961$ and $p_3 = 1, 073, 129, 417, 747, 493, 923$.

Finally we use the Chinese remainder theorem to reconstruct the generating function N/D . We conclude that this is the desired solution because the maximum coefficient in N is about $6.797227759 \times 10^{-17} p_1 p_2 p_3$. Of course, if one needs a rigorous proof, one needs a bound for $N|_{q=1}$, which should not be a hard problem.

5.3 Possible improvements

In summary, Step 1 is simple but necessary for general problems, but we shall consider delay this step in practical problems. Step 3 is of independent interest. Give a faster algorithm is possible but we will not discuss here. Step 2 is the crucial step, and the number of terms obtained in this step is dominant. We shall concentrate on improvements to this step.

Let us consider the linear Diophantine system $A\alpha = b$ with augmented matrix (A, b) . Clearly elementary row operation will not change the solution space. So it is possible to find a simpler matrix (A', b') with the same solution space. This step may be achieved by the well-known Lenstra Lenstra Lovasz's (LLL) basis reduction algorithm [10, 14]. The author is considering upgrade the package by using this idea.

There are many ideas to improve the algorithm. An interaction with known theories will give hints for improvements. For instance, Stanley's monster reciprocity theory contains some algorithmic ideas. See [16, 19]. Our ultimate goal is to develop a classic algorithm in this subject. We believe that such an algorithm should contain the following features.

1. We shall deal with the inhomogeneous case directly, avoid using Brion's theorem, which is too expensive when the number of vertices is large.
2. We shall give a decomposition dealing with Laurent polynomial numerator in a uniform way. The outcome will be analogous of simplicial cones.
3. We shall apply Barvinok's decomposition of simplicial cones into unimodular simplicial cones or the like, which we believe to be the genuine essence of Barvinok's algorithm.

Acknowledgements: This work was supported by the Natural Science Foundation of China (11171231).

References

- [1] K. Aardal and A. K. Lenstra, Hard equality constrained integer knapsacks, *Math. Operations Research*, 29 (2004) 724–738.
- [2] M. Ahmed, J. D. Loera, R. Hemmecke, Polyhedral cones of magic cubes and squares, *Discrete and Computational Geometry*, 25–41, *Algorithms Combin.*, 25, Springer, Berlin, 2003.
- [3] G. E. Andrews, MacMahon's partition analysis. I. The lecture hall partition theorem, *Mathematical essays in honor of Gian-Carlo Rota* (Cambridge, MA, 1996), *Progr. Math.*, vol. 161, Birkhauser Boston, Boston, MA, 1998, pp.1–22.
- [4] G. E. Andrews, P. Paule, and A. Riese, MacMahon's partition analysis III: the Omega package, *Europ. J. Combin.* 22 (2001) 887–904.
- [5] G. E. Andrews, P. Paule, and A. Riese, MacMahon's partition analysis VI: A new reduction algorithm, *Ann. Comb.*, 5 (2001) 251–270.
- [6] A. I. Barvinok, Polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed, *Math. Operations Research* 19 (1994) 769–779.

- [7] M. Beck, S. Robins, *Computing the Continuous Discretely: Integer–Point Enumeration in Polyhedra*, Undergrad. Texts Math., Springer, New York, 2007.
- [8] E. Ehrhart, *Polynomes arithmétiques et methode des polyédres en combinatoire*, International Series of Numerical mathematics, vol 35, Birkhäuser, Basel 1977.
- [9] A. Garsia, G. Musiker, N. Wallach, and G. Xin, Invariants, Kronecker products, and combinatorics of some remarkable Diophantine systems, *Adv. in Appl. Math.*, 42 (2009) 392–421.
- [10] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithm and Combinatorial Optimization*, second edition. Algorithms and combinatorics, 2, Springer-Verlag, Berlin, 1993.
- [11] M. Köppe, A primal Barvinok algorithm based on irrational decompositions. *SIAM J. Discrete Math.* 21 (2007), 220–236 (electronic).
- [12] J. A. D. Loera, The many aspects of counting lattice points in polytopes, *Mathematische Semesterberichte* 52 (2005), 175–195.
- [13] J. A. D. Loera, R. Hemmecke, J. Tauzer, and R. Yoshida, Effective Lattice point counting in Rational Convex polytopes, *J. Symbolic Comput.*, 38 (2004) 1273–1302.
- [14] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley-Interscience, 1986.
- [15] N.J.A. Sloane, The On-Line Encyclopedia of Integer Sequences. Published electronically at <http://oeis.org>, 2012.
- [16] R. P. Stanley, Combinatorial reciprocity theorems, *Adv. Math.* 14 (1974) 194–253.
- [17] R. P. Stanley, *Enumerative Combinatorics*, Vol 1, Cambridge University Press, Cambridge, 1997.
- [18] G. Xin, A fast algorithm for MacMahon’s partition analysis, *Electron. J. Combin.* 11 (2004), R58 (electronic).
- [19] G. Xin, Generalization of Stanley’s monster reciprocity theorem, *JCTA* 114 (2007) 1526–1544.