A hybrid algorithm for multi-homogeneous Bézout number

Yuhui Tao, Heng Liang *, Fengshan Bai

Department of Mathematical Sciences, Tsinghua University, Beijing 100084, PR China

Abstract

The multi-homogenous homotopy continuation method can solve all isolated solutions of polynomial systems. Different variable partition yields different multi-homogenous Bézout number, which gives the upper bound of the number of isolated solutions. However, the computation of the multi-homogenous Bézout number is hard. In this paper, the permanent formulation of the multi-homogenous Bézout number is considered. The intensive and systemic computations are made for the method of row expansion with memory, the precise and the approximate permanent methods. Each of these methods has its own advantage. Hence a hybrid algorithm is naturally presented. This method works for \( n \) about 30 contrasting with 15 before, where \( n \) is the number of the variables of the polynomial system.

© 2006 Elsevier Inc. All rights reserved.

Keywords: Polynomial systems; Multi-homogenous Bézout number; Hybrid algorithm; Permanent

1. Introduction

It is very common in many fields of science and engineering to find all isolated solutions of the following system of polynomial equations in \( n \) variables.

\[
p_1(x_1, x_2, \ldots, x_n) = 0, \\
\vdots \\
p_n(x_1, x_2, \ldots, x_n) = 0,
\]

where \((x_1, x_2, \ldots, x_n) = X \in C^n\). Denote \( P = (p_1, \ldots, p_n)^T \). Multi-homogenous homotopy continuation method is one of the most efficient approaches to solve the above problem numerically [2,4,9,10,14].

By Bézout Theorem, the number of isolated solutions of a polynomial system is bounded above by the Bézout number \( D = \prod_{j=1}^{n} d_j \), where \( d_j \) is the degree of the \( j \)th equation in the system (1.1). Homotopy methods can solve all solutions of (1.1) by following \( D \) curves. However, \( D \) is often far larger than the number of isolated solutions that the system actually has, especially for problems arising from practical applications. Thus too
much time may be wasted in classical continuation methods. Morgan and Sommese [9] give the multi-homogeneous continuation method. Different partitions of variables give different multi-homogeneous Bézout numbers, which are all upper bounds for the number of isolated solutions of a polynomial system. It is desired to find the optimal partition with the minimal multi-homogeneous Bézout number. The computational cost of this method mainly comes from two aspects. One is finding a partition with the global minimal multi-homogeneous Bézout number. The other is computing the multi-homogenous Bézout number for a given partition of variables. The former has been studied relatively intensively [4,6,5,7,14]. However, with few exceptions [14], very little has been done on the latter one. In this paper, we discuss the permanent formulation of the multi-homogenous Bézout number. Combining the permanent algorithms with Wampler’s row expansion with memory method [14], a hybrid algorithm is proposed, which can work for much larger problems.

In Section 2, multi-homogenous Bézout number is introduced briefly. Then we give the permanent formulation for computing multi-homogenous Bézout number of a given variable partition. Wampler’s row expansion methods and some efficient permanent methods are described in Section 3. A plenty numerical experiments have made to compare these methods in Section 4. The practical hybrid algorithm is presented in Section 5. Finally, some discussions are given in Section 6.

2. Formulation with permanent

Let us consider the system (1.1). A partition of \( X = (x_1, x_2, \ldots, x_n) \) is defined as \( T = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} \), where

\[
x^{(j)} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}, \quad j = 1, 2, \ldots, m.
\]

(2.1)

\( K = [k_1, k_2, \ldots, k_m] \) is called the partitioning vector of \( T \). The subscripts above satisfy \( \sum_{j=1}^{m} k_j = n \), \( m \leq n \).

Assume that the degree of \( p(X) \) in (1.1) with respect to \( x^{(j)} \) is \( d_{ij} \), \( i = 1, 2, \ldots, n \), \( j = 1, 2, \ldots, m \). The corresponding degree matrix under the partition \( T \) is defined as

\[
D = \begin{pmatrix}
d_{11} & d_{12} & \cdots & d_{1m} \\
d_{21} & d_{22} & \cdots & d_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
d_{n1} & d_{n2} & \cdots & d_{nm}
\end{pmatrix}.
\]

The multi-homogeneous Bézout number of the system (1.1) with respect to the partition \( T \), denoted by \( B_m \), is defined as the coefficient of \( a_1^{d_{11}}a_2^{d_{22}}\cdots a_m^{d_{nm}} \) in the following polynomial of \( a_1, a_2, \ldots, a_m \):

\[
\phi(a_1, a_2, \ldots, a_m) = (d_{11}a_1 + d_{12}a_2 + \cdots + d_{1m}a_m) \cdots (d_{n1}a_1 + d_{n2}a_2 + \cdots + d_{nm}a_m)
\]

where \( d_{ij} (i = 1, \ldots, n; \ j = 1, \ldots, m) \) are elements of the degree matrix \( D \). Accordingly one can prove that

\[
B_m = \sum_{(i_1, i_2, \ldots, i_n)} d_{i_1i_2} \cdots d_{i_m}, \quad j \text{ is chosen exactly } k_j \text{ times} (j = 1, \ldots, m).
\]

Morgan [9] shows that \( B_m \) also gives an upper bound of the number of isolated solutions of the system (1.1).

The permanent of an \( n \times n \) matrix \( A = [a_{ij}] \) is defined as

\[
\text{per}(A) = \sum_{\sigma \in \Lambda_n} \prod_{i=1}^{n} a_{\sigma(i)j},
\]

(2.2)

where \( \Lambda_n \) denotes the set of all possible permutations of \( \{1, 2, \ldots, n\} \). The definition of permanent is similar to that of determinant. However the permanent is a much harder problem in computation. Computing the permanent is a \# \( P \)-complete problem in counting [13], where \# \( P \) is defined as a class of functions that count the number of solutions in an NP problem. Note that, \# \( P \)-complete problem in counting is no easier than an NP-complete problem in combinatorial optimization, hence the computational cost would grow as an exponential function of the dimension of matrix.
Define an $n \times n$ matrix

$$D^* = \begin{pmatrix} d_{11} & \cdots & d_{11} & d_{12} & \cdots & d_{12} & \cdots & d_{1m} & \cdots & d_{1m} \\ d_{21} & \cdots & d_{21} & d_{22} & \cdots & d_{22} & \cdots & d_{2m} & \cdots & d_{2m} \\ \vdots \\ d_{n1} & \cdots & d_{n1} & d_{n2} & \cdots & d_{n2} & \cdots & d_{nm} & \cdots & d_{nm} \end{pmatrix}$$

which is obtained by extending the each column of degree matrix $D$ to $k_i (i = 1, \ldots, m)$ columns, respectively.

There are following relation between the multi-homogeneous Bézout number of a given variable partition and the permanent of the matrix $D^*$.

$$B_m = \sum_{(i_1,i_2,\ldots,i_n)} d_{i_11}d_{i_22}\cdots d_{i_nn} = \frac{1}{k_1!k_2!\cdots k_m!} \text{Per}(D^*).$$ (2.3)

Thus computing multi-homogeneous Bézout number $B_m$ can be reduced to evaluating the permanent of the matrix. It is no easier than an NP hard problem.

3. Computational methods

3.1. Row expansion method

The basic algorithm for computing the multi-homogeneous Bézout number is to enumerate the permissible combinations, form the corresponding degree products, and add them up. The computational cost of the method can reach to $O(n!)$ for $m = n$.

Wampler proposes two approaches, row expansion method and row expansion with memory, to reduce the computational cost [14]. Row expansion method is a recursive algorithm with $O(n)$ memory. For the most expensive case, $m = n$, the number of multiplications for the row expansion algorithm is approximately $(e - 1)/(n - 1)$ times the number required by the basic algorithm. Row expansion with memory uses the same recursion formula, but records each minor value to avoid recomputing it later. For the worst case, $m = n$, the total number of multiplications comes to $n2^{n-1}$, which is much smaller than that required by the row expansion algorithm. However, this improvement comes at the expense of a memory array that in the worst case has $2n$ elements, compared to only $O(n)$ elements for row expansion without memory.

REM(row expansion with memory method)

Input: $D$ – degree matrix, $K = [k_1,k_2,\ldots,k_m]$ – partitioning vector.

Output: $B_m$ – the multi-homogeneous Bézout number.

step 1: $b(D,K,n+1) = 1$.

step 2: For $i$ from $n$ to 1, $b(D,K,i) = \sum_{j=1,j\neq i}^m d_{ij} \times b(D,K - e_j,i + 1)$ and record all $b(D,K,i)$’s, where $e_j = (0,0,\ldots,0,1,0,\ldots,0) \in \mathbb{Z}^n$.

step 3: $B_m = b(D,K,1)$.

Though Wampler’s methods are the best approach among all exact algorithms so far. The computation of large scale problem will be limited by time and memory cost of these methods. In this paper, we introduce an equivalence relation between the permanent of the matrix and the multi-homogenous Bézout number. Based on this, we develop the more efficient algorithm to compute multi-homogenous Bézout number.

3.2. Method with RNW for permanent

The best-known algorithm on precise evaluation of the permanent, due to Ryser [12] and later improved by Nijenhuis and Wilf [11], runs $O(n2^{n-1})$ in time with $O(n)$ memory. We call it RNW method.
Let $S$ be any set, and $|S|$ denote the number of elements of the set $S$. Ryser [12] shows that the permanent of a matrix $A$ can be computed by

$$\text{Per}(A) = (-1)^n \sum_{S} (-1)^{|S|} \prod_{i=1}^{n} \left( \sum_{j \in S} a_{ij} \right).$$

(3.1)

Here matrix $A$ is $n \times n$ and $S$ runs over all possible subsets of $\{1, 2, \ldots, n\}$. This method is based on the inclusion and exclusion principle.

The computational cost of Ryser formula is $O(n^2 \cdot 2^{n-1})$. Nijenhuis and Wilf [11] improve Ryser method from the viewpoint of effective implementation.

Note that for each subset $S$, one has to calculate

$$f(S) = \prod_{i=1}^{n} \lambda_i(S),$$

where

$$\lambda_i(S) = \sum_{j \in S} a_{ij} \quad (i = 1, 2, \ldots, n).$$

Arranging the sequence of all subsets according to Hamilton walk on an $(n - 1)$-cube [11, Chapter 1], the current subset $S$ differs from its predecessor $S'$ only by a single element, $j$. Thus

$$\lambda_i(S) = \lambda_i(S') \pm a_{ij} \quad (i = 1, 2, \ldots, n).$$

This reduces the number of additions in a subset $S$ from $n \cdot |S|$ to $n$. So the total computational cost becomes $O(n \cdot 2^{n-1})$ and the memory cost is $O(n)$. Then an algorithm of multi-homogenous Bézout number based on RNW, which is called RNW-B, is proposed as follows.

**RNW-B (Bézout number algorithm based on RNW)**

**Input:** $D$ – degree matrix, $K = [k_1, k_2, \ldots, k_m]$ – partitioning vector.

**Output:** $B_m$ – the multi-homogeneous Bézout number.

**step 1:** $D' \leftarrow D$ and $K$.

**step 2:** $X \leftarrow \text{Per}(D')$ using RNW.

**step 3:** $B_m = \frac{X}{k_1^{k_2 \cdots k_m}}$.

### 3.3. Approximate method with KKLLL for permanent

For the polynomial system, the number of all possible variable partitions will become very large when $n$ grows. As Table 3.1 shown, its growth rate is strongly exponential. Hence the exact method does not work when $n$ is large. Approximate algorithm, which can give a reasonable estimate for the optimal variable partition, is the most promising approach to reduce the computational cost for large scale problems. In the framework of approximation search, it seems reasonable to compute the multi-homogenous Bézout number only approximately. This offers a trade-off between the accuracy and the amount of work in search for large $n$.

### Table 3.1
The growth of the numbers of all possible variable partitions with $n$

<table>
<thead>
<tr>
<th>$n$</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^n$</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>512</td>
<td>1024</td>
<td>32768</td>
<td>1048576</td>
</tr>
<tr>
<td># Partition</td>
<td>52</td>
<td>203</td>
<td>877</td>
<td>4140</td>
<td>21147</td>
<td>115975</td>
<td>1382958545</td>
<td>51724158235372</td>
</tr>
</tbody>
</table>
Let an estimator $Y$ of the permanent of matrix $A$ be unbiased, i.e., $E(Y) = \text{Per}(A)$ and $E(Y^2) < \infty$. An $(\epsilon, \delta)$-approximation algorithm for estimating $\text{Per}(A)$ is a Monte Carlo algorithm which accepts an input $A$ and two parameters $\epsilon$ and $\delta$. The output $Y$ of the algorithm gives an estimate of $\text{Per}(A)$ that satisfies
\[
P[r((1 - \epsilon)\text{Per}(A) \leq Y \leq (1 + \epsilon)\text{Per}(A))] \geq 1 - \delta.
\]
(3.2)

According to Chebyshev inequality, when
\[
N \geq \frac{E[Y]}{\epsilon^2} \cdot \frac{1}{\delta^2}
\]
for $\epsilon$ and $\delta$ given, the Monte Carlo method, $Y = \frac{1}{N} \sum_{i=1}^{N} Y_i$, gives an $(\epsilon \delta)$ approximation to $\text{Per}(A)$. The dependence of $N$ on $\delta$ can be improved to $O(\ln(\frac{1}{\delta}))$ [3]. Hence the efficiency of a Monte Carlo method relies heavily on $\frac{E[Y^2]}{E[Y]^2}$, which is called the critical ratio.

A very efficient approximation for permanent is proposed by Karmarkar et al. [3], which is known as KKLLL algorithm. The algorithm is outlined as follows.

**KKLLL Method** (Karmarkar/Karp/Lipton/Lovasz/Luby)

*Input:* $A$ – an $n \times n$ nonnegative matrix.

*Output:* $X_A$ – the estimate for $\text{Per}(A)$.

**step 1:** $D^*$. For all $i, j$, $1 \leq i, j \leq n$,

- If $A_{ij} = 0$ then $B_{ij} \leftarrow 0$;
- Elseif $A_{ij} > 0$ then randomly and independently choose
  \[B_{ij} \in \{\sqrt{A_{ij}}w_0, \sqrt{A_{ij}}w_1, \sqrt{A_{ij}}w_2\}\text{ with probability } \frac{1}{3}.
\]

**step 2:** $X_A = \det(B)\det(B)$.

Karmarkar et al. prove that the estimator $X_A$ of the permanent of matrix $A$ is unbiased, i.e., $E(X_A) = \text{Per}(A)$ and $E(X_A^2) < \infty$. So a Monte Carlo algorithm can be established. Based on KKLLL, an approximation algorithm of multi-homogenous Bézout number, which is called KKLLL-B, is presented.

**KKLLL-B (Bézout number algorithm based on KKLLL)**

*Input:* $D$ – degree matrix, $K = [k_1, k_2, \ldots, k_m]$ – partitioning vector.

*Output:* $B_m$ – the multi-homogeneous Bézout number.

**step 1:** $D_e \leftarrow D$ and $K$.

**step 2:** $X \leftarrow$ the estimate for $\text{Per}(D^*)$ using KKLLL

**step 3:** $B_m = \frac{X}{k_1w_1 \cdots k_m}$.

Another practical algorithm of approximating permanent is random path method, which is a randomized Laplace expansion with column pivoting of matrix. The detail of the algorithm sees to [8].

In order to finding the variable partition with small Bézout number, we can first estimate the value by KKLLL or random path method with cheap cost. If the estimate is too large, it is safe in dismissing the variable partition. Only when the estimate is relative small, the precise evaluation is necessary. In this paper, we use KKLLL because it is simple in programming with MATLAB.

### 4. Numerical results

A large amount of computations have been done to compare the row expansion methods and permanent methods for computing multi-homogenous Bézout number. All of our computations in this paper are performed on 32-bit Intel Pentium IV personal computer (2.8 GHz), with MATLAB as programming language for convenient. The Wampler’s row expansion method with memory is denoted by REM for briefness hereafter. Because the results of REM are always better than row expansion method, the results of row expansion method are not listed in this paper. The precise algorithm with permanent method adopts the RNW-B. The approximation one adopts the KKLLL-B.
When \( n \leq 10 \), the computation of multi-homogenous Bézout number is trivial. RNW-B is adopted in this paper because it is simple in programming. Hence we only discuss the case of \( n > 10 \). In this paper, the sparse polynomial systems are generated randomly. The mechanism of generating random systems is the same as in [5], where \( d \) denote the maximum degree of the polynomial system and \( s \) denote the number of terms of the polynomial equation.

First, we note that, though the computational times of different variable partitions are different with REM for fixed \((n, m)\), they concentrate very well. The computational time is mainly determined by \( n \) and \( m \) and is not sensitive to the difference of variable partitions within the same \((n, m)\). Take example for \( n = 15, m = 4–12 \), we sample 100 variable partitions for each possible \((n, m)\) pair with \( d = 3, s = n \). Let \( T \) be the running time of computing Bézout number. The maximum, minimum, average values and standard errors of \( T \), which are denoted by \( \max(T) \), \( \min(T) \), \( \text{mean}(T) \) and \( \text{std}(T) \), respectively, are listed in Table 4.1. The relative errors, \( \text{re1}(T) \) and \( \text{re2}(T) \), are also presented, which are defined as

\[
\text{re1}(T) = \frac{\max(T) - \text{mean}(T)}{\text{mean}(T)}, \quad \text{re2}(T) = \frac{\text{mean}(T) - \min(T)}{\text{mean}(T)}.
\]

Then let \( m \) fixed, we investigate the increase of the computational times with \( n \). Take \( m = 2–9, n = 11–30 \). We compute 100 samples in each \((n, m)\) pair with \( d = 3, s = n \). The average time of each set of samples is listed in Table 4.2.

When \( n \) is fixed, the computational times of precise permanent method are almost constants depending on \( m \). These constants are listed in Table 4.3. It is obvious that REM is superior to RNW-B when \( m \) is small.

<table>
<thead>
<tr>
<th>Table 4.1</th>
<th>The concentration of CPU times (s) for fixed ((n, k)) (100 samples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((n, k))</td>
<td>((15,4))</td>
</tr>
<tr>
<td>mean(T)</td>
<td>0.0049</td>
</tr>
<tr>
<td>max(T)</td>
<td>0.0058</td>
</tr>
<tr>
<td>min(T)</td>
<td>0.0025</td>
</tr>
<tr>
<td>rel1(T)</td>
<td>0.1838</td>
</tr>
<tr>
<td>std(T)</td>
<td>0.0007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4.2</th>
<th>The average CPU times (s) for ((n, m)) by REM method</th>
</tr>
</thead>
<tbody>
<tr>
<td>((n, m))</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>0.0005</td>
</tr>
<tr>
<td>12</td>
<td>0.0006</td>
</tr>
<tr>
<td>13</td>
<td>0.0006</td>
</tr>
<tr>
<td>14</td>
<td>0.0007</td>
</tr>
<tr>
<td>15</td>
<td>0.0008</td>
</tr>
<tr>
<td>16</td>
<td>0.0009</td>
</tr>
<tr>
<td>17</td>
<td>0.0010</td>
</tr>
<tr>
<td>18</td>
<td>0.0011</td>
</tr>
<tr>
<td>19</td>
<td>0.0012</td>
</tr>
<tr>
<td>20</td>
<td>0.0013</td>
</tr>
<tr>
<td>21</td>
<td>0.0014</td>
</tr>
<tr>
<td>22</td>
<td>0.0015</td>
</tr>
<tr>
<td>23</td>
<td>0.0016</td>
</tr>
<tr>
<td>24</td>
<td>0.0018</td>
</tr>
<tr>
<td>25</td>
<td>0.0019</td>
</tr>
<tr>
<td>26</td>
<td>0.0020</td>
</tr>
<tr>
<td>27</td>
<td>0.0022</td>
</tr>
<tr>
<td>28</td>
<td>0.0024</td>
</tr>
<tr>
<td>29</td>
<td>0.0025</td>
</tr>
<tr>
<td>30</td>
<td>0.0027</td>
</tr>
</tbody>
</table>
However, the advantage of the permanent method emerges gradually when \( n \) grows. The logarithmic time curves of RNW-B and REM with \( m = 2, 5, 8 \) are illustrated in Fig. 4.1.

Take \( n = 15, 20 \), the comparisons between RNW-B and REM are shown in Fig. 4.2. When \( n \) is fixed, the curve of the computational time of RNW-B is almost a horizontal line, and that of REM ascends with the \( k \) growing. From Fig. 4.2, it is observed that the critical points are (15, 7) and (20, 12), respectively when \( n = 15, 20 \). Based on a great number of computations, we give the critical points \( (n, m^*) \) empirically in Table 4.4.

When \( n \) is large, introducing the approximation algorithm of the multi-homogenous Bézout number is reasonable. The Monte Carlo method, KLLL-B given in Section 3.2, is used. The sample size of the Monte Carlo method can be roughly taken as

\[
N = f(n) \cdot \frac{1}{\varepsilon^2 \delta},
\]

(4.1)

according to (3.2) and (3.3), where \( f(n) \) is determined by the critical ratio, \( \frac{\varepsilon^{p+1} \varepsilon \varepsilon I}{\varepsilon I^2} \). Here we set \( \varepsilon = 0.2 \) and \( \delta = \frac{1}{2} \), and thus \( N = 50 \times f(n) \). If \( f(n) = n \) in (4.1), the sample size of KLLL is \( N = 1000 \) for an \( n \times n \) matrix.

![Table 4.3](image)

<table>
<thead>
<tr>
<th>( n )</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>0.0027</td>
<td>0.0050</td>
<td>0.0087</td>
<td>0.0176</td>
<td>0.0359</td>
<td>0.0714</td>
<td>0.1434</td>
<td>0.2873</td>
</tr>
<tr>
<td>( n )</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>( T )</td>
<td>0.5867</td>
<td>1.1665</td>
<td>2.3526</td>
<td>4.7161</td>
<td>9.7004</td>
<td>19.150</td>
<td>38.4725</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4.1. The comparison of RNW-B and REM with small \( m \).

Fig. 4.2. The comparison of RNW-B and REM when \( n = 15, 20 \).
First, we test the accuracy of KKLLL-B method. 300 sets of the sparse polynomial systems generated randomly with \( n = 20, \; d = 3, \; s = 20 \) are executed. Among them, there are 100 sets for each \( k = 5, 10, 15 \), respectively. Each set of polynomial systems contains 10 samples. The variable partitions are sorted into increasing order according to their corresponding Bézout numbers. Let \( N_1 \) be the number of sets in which the orders obtained by KKLLL-B and RNW-B are absolutely identical, \( N_2 \) be the number of sets in which the first three variable partitions obtained by KKLLL-B are right, \( N_3 \) be the number of sets in which the variable partitions with the minimal Bézout number are found by KKLLL-B. Taking \( f(n) = n \) and \( f(n) = n/2 \), the computational results are listed in Table 4.5. Only 11 optimal variable partitions are missed by KKLLL-B estimator in all 300 sets of numerical experiments. Among these 11, the minimal Bézout numbers found by KKLLL-B are the second minimal in 10 and the third in 1. According to the data in Table 4.5, the improvement from \( f(n) = n/2 \) to \( f(n) = n \) is not significant. The precision achieved by \( f(n) = n/2 \) is acceptable. We also test some other \((n, m)\)'s. The similar results are obtained. Hence the KKLLL-B can distinguish the order of different variable partitions successfully. We always take \( f(n) = n/2 \) in later computations.

Table 4.4
The empirically critical points for \( n = 11–25 \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( 11 )</th>
<th>( 12 )</th>
<th>( 13 )</th>
<th>( 14 )</th>
<th>( 15 )</th>
<th>( 16 )</th>
<th>( 17 )</th>
<th>( 18 )</th>
<th>( 19 )</th>
<th>( 20 )</th>
<th>( 21 )</th>
<th>( 22 )</th>
<th>( 23 )</th>
<th>( 24 )</th>
<th>( 25 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m^* )</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 4.5
The precision of approximating Bézout number by KKLLL-B

<table>
<thead>
<tr>
<th>( k )</th>
<th>( f(n) = n/2 )</th>
<th>( f(n) = n )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( N_1 )</td>
<td>( N_2 )</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>75</td>
</tr>
<tr>
<td>10</td>
<td>35</td>
<td>80</td>
</tr>
<tr>
<td>15</td>
<td>56</td>
<td>85</td>
</tr>
</tbody>
</table>

Table 4.6
The CPU times (s) of approximating Bézout number by KKLLL-B

<table>
<thead>
<tr>
<th>( n )</th>
<th>( 11 )</th>
<th>( 12 )</th>
<th>( 13 )</th>
<th>( 14 )</th>
<th>( 15 )</th>
<th>( 16 )</th>
<th>( 17 )</th>
<th>( 18 )</th>
<th>( 19 )</th>
<th>( 20 )</th>
<th>( 21 )</th>
<th>( 22 )</th>
<th>( 23 )</th>
<th>( 24 )</th>
<th>( 25 )</th>
<th>( 26 )</th>
<th>( 27 )</th>
<th>( 28 )</th>
<th>( 29 )</th>
<th>( 30 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>0.0174</td>
<td>0.0207</td>
<td>0.0246</td>
<td>0.0290</td>
<td>0.0338</td>
<td>0.0394</td>
<td>0.0467</td>
<td>0.0535</td>
<td>0.0613</td>
<td>0.0700</td>
<td>0.0795</td>
<td>0.0901</td>
<td>0.1010</td>
<td>0.1167</td>
<td>0.1336</td>
<td>0.1496</td>
<td>0.1664</td>
<td>0.1896</td>
<td>0.2091</td>
<td>0.2311</td>
</tr>
</tbody>
</table>

Fig. 4.3. The comparison of KKLLL-B and REM.
The average computational times of KKLLL-B for \( n = 11-30 \) are shown in Table 4.6. Comparing Tables 4.3 and 4.6, KKLLL-B is faster than RNW-B when \( n > 16 \).

In order to compare REM method with KKLLL-B, we generate 100 sparse polynomial systems randomly with each \( n = 11-30 \). The average computational times of KKLLL-B is roughly between that of \( m = 5 \) and \( m = 6 \) with REM, as shown in Fig. 4.3.

5. Hybrid algorithm

A plentiful numerical computations show that REM (row expansion with memory method), RNW-B and KKLLL-B, are all efficient methods for Multi-homogeneous Bézout number. Each of these three methods has its own advantage. Take \( n \) and \( m \) as parameters, the preponderant domain of each method can be illustrated by Fig. 5.1, which naturally suggests the following hybrid algorithm for multi-homogeneous Bézout number computation.

\[
\text{Hybrid algorithm for multi-homogeneous Bézout number}
\]

**Input:** \( D \) – degree matrix, \( K = [k_1, k_2, \ldots, k_m] \) – partitioning vector.

**Output:** \( B_m \) – the multi-homogeneous Bézout number.

**Case 1:** When \( n \leq 10 \), use RNW-B;

**Case 2:** When \( 10 < n \leq 15 \), if \( m \leq m^* \) use REM, otherwise use RNW-B, where the value of \( m^* \) is listed in Table 4.4;

**Case 3:** When \( n > 15 \) and \( m < 6 \), use REM;

**Case 4:** When \( n > 15 \) and \( m \geq 6 \), use KKLLL-B.

6. Conclusion and discussion

The multi-homogenous homotopy continuation method is one of the most common approaches to solve all isolated solutions of polynomial systems. In this method, the computation of the multi-homogeneous Bézout number for a given variable partition is a very difficult problem. It is the main limitation to make the size of the computable problems larger. Hence an efficient algorithm on computing multi-homogeneous Bézout number is considered as a very challenging problem [5].

Wampler’s row expansion with memory method is the best algorithm for this problem so far. In this paper, we formulate the permanent of matrix for computing the Bézout number. Notice that only ordinal comparisons for Bézout numbers corresponding to different variable partitions are needed here. Hence the permanent formulation can fully make use of this important property. The intensive and systemic comparisons are made between the Wampler’s method and the precise and approximation permanent methods. The results show that
each of these methods has its advantage. Take \( n \) and \( m \) as parameters, we give the domains in which each method predominates. Then the hybrid algorithm is established. When \( n \leq 15 \), the determinate methods, REM for small \( m \) and RNW-B for large \( m \), is appropriate to be used. When \( m \) is small (\(<6\)), REM works up to \( n = 30 \). For \( n > 15 \) and \( m \geq 6 \), the approximation method has to be used. The estimate obtained by KKLll-B is good enough to distinguish the order of different variable partitions when \( n \leq 30 \) at least.

If we want to solve any practical problems whose number of variables is up to 50 or so, the search of optimal variable partition and the computation of Bézout number are both needed improving. Randomized algorithms, which have been used to find the optimal variable partition [6], provide a possible way for dealing with the large scale problems. But for large scale problems, such as \( n > 30 \), the computational efficiency of Bézout number with fixed variable partition must be improved further. Barvinok proposed a strongly polynomial time algorithm for computing permanent of the rank fixed matrix in [1]. Consider that the rank of degree matrix is no more than \( m \) for an \( n \) partition, Barvinok’s method maybe inspire useful enlightenment for further improving the computational efficiency of Bézout number. Another promising approach is to combine the search of optimal variable partition with approximation comparison of Bézout number and establish the incorporated algorithm with global convergence. These are valuable research problems in the future.

References