COMPUTATIONAL ASPECTS OF THE AYAHASTA ALGORITHM

SUBHAS KAK

Department of Electrical and Computer Engineering
Louisiana State University
Baton Rouge, LA 70803

(Revised 30 March 1985)

This paper investigates the Ayahasta algorithm from the point of view of computational complexity.
It is also shown that this algorithm is as efficient as the popularity run Chinese Remainder algorithms in solving a system of congruences.

INTRODUCTION

The solution to the problem of a system of congruences where the moduli are pairwise prime is normally obtained by the Chinese Remainder (CR) algorithm. This algorithm is of particular interest to computer and communication scientists because systems of congruences arise frequently in coding, cryptography, signal processing, and in computer design. The significance of an alternate, efficient algorithm cannot be overstated especially since one may obtain an elegant implementation structure even if the complexity remained unchanged. An algorithm to solve congruences in a manner different to the CR method is presented in Ayahasta. Recently I have brought this method to the attention of computer scientists. I believe the reason this algorithm has not been described earlier in computer literature is because history books often state it only in the context of the solution of the linear indeterminate equation as—by

In Ayahasta, this method is called jagat (the pulverizer). Even though I have sometimes called it the Ayahasta algorithm, to conform to the convention of associating a person with a result, the traditional name is very appropriate. Considering that Ayahasta's system was a modification of the earlier Paandita Siddharta it is likely that this method, like other results in his book, was already well known before him.

Commentaries on this method were given by Bhaskara I (522 A.D.), Brahmagupta (628 A.D.), Mahavira (850 A.D.), Bhaskara II (1150 A.D.) and others. This algorithm for the solution of a linear indeterminate equation appears to be the earliest recorded anywhere. Diophantus of Alexandria (250 A.D.) had described determinate and indeterminate problems but provided no methods of solutions; furthermore, he was only concerned with rational solutions.
The Aryabhata Algorithm

The Aryabhata problem, in its simplest form, is: Find a number \( x \) less than \( d_1d_2\ldots d_n \) which, when divided by \( d_1 \) and \( d_2 \ldots d_n \), leaves the residues \( r_1 \) and \( r_2 \ldots r_n \), respectively, where \( x = r_1 = r_2 \ldots r_n \), and if \( d_1 \) and \( d_2 \) are relatively prime. In other words, find \( x \) so that:

\[
\begin{align*}
x \mod d_1 &= r_1 \\
x \mod d_2 &= r_2 = r_3 = \ldots = r_n
\end{align*}
\]  

The pulverizer to solve this problem is given in A2.32-33. There are several difficulties in translating these mantras described in the literature. One was the commentary and the methods of the later mathematicians to help in the translation. The following is the translation by Clark based on Paramewar's interpretation and of Brahamagupta's pulverizer:

A2.32-33. Divide the divisor which gives the greatest agna (agu: remainder) by the divisor which gives the smaller agna. The remainder is reciprocally divided that is to say, the remainder becomes the divisor of the original divisor, and the remainder of this second division becomes the divisor of the second divisor, etc. (The quotients are placed below each other in the so-called chain.) (The last remainder) is multiplied by an assumed number and added to the difference between the agnas. Multiply the penultimate number by the number above it and add the number which is below it. (Continue this process to the top of the chain). Divide (the top number) by the divisor which gives the smaller agna. Multiply the remainder by the divisor which gives the greater agna. And this product to the greater agna. The result is the number which will satisfy both divisors and both agnas.

Clark quotes another translation by Gangoly that leads to a method differing somewhat in detail. Since our objective is to emphasize the essential method we express it in the following modern form:

Theorem : Let \( d_1 > d_2 \) and let

\[
\begin{align*}
d_1 &= a_1 + r_1, \quad r_1 < d_1 \\
d_2 &= a_2 + r_2 \\
r_1 &= a_3 + r_3 \\
r_2 &= a_4 + r_4 + r_5 + r_6 + \ldots
\end{align*}
\]  

(2)
and \( a_{n+1} = 1 \), then write the \( a \)'s in a column, appending \( e = x_n - x_i \), and reduce this as shown:

\[
\begin{align*}
   a & \rightarrow a & \rightarrow a' \\
   a_1 & \rightarrow a_1 & \rightarrow a_1' \\
   a_2 & \rightarrow a_2 & \rightarrow a_2' \\
   \vdots & \rightarrow \vdots & \rightarrow \vdots \\
   a_{n+1} & \rightarrow a_{n+1} & \rightarrow a_{n+1}' \\
   e & \rightarrow e \\
\end{align*}
\]

(3)

where to obtain column \( i+1 \), we drop the last entry of column \( i \), and replace the last but two entry by its product with the last but one entry, plus the entry being dropped.

Let

\[
\begin{align*}
   (a^0) & \cdot a' \mod d_i \equiv a_i \\
   (a^0) & \cdot b' \mod d_i \equiv b_i,
\end{align*}
\]

then

\[
\begin{align*}
   x &= a_{i+1} + x_1 \\
   &= b_{i+1} + x_2
\end{align*}
\]

(4)

(5)

This \( x \) is the least positive solution; other solutions will be \( x + \text{constant} \) (i.e. \{d_i, d_i\}).

Proof: The proof of the theorem is elementary and it is stated in Datta and Singh. We provide the outline of this proof below.

The operations in (2) represent division of \( d_i \) by \( d_{i-1}, d_{i-2} \) by the remainder in the previous step, and so on in sequence. The \( a \)'s are the quotients obtained in this process. Since (1) can be rewritten as \( x = a_i + x_1 = b_i + x_2 \), therefore, the problem is transformed to the solution in terms of \( a \) and \( b \) of

\[
ad_1 - b d_0 = e.
\]

Use of the equations of (2) repeatedly in this equation until the last \( r_n \), which is 1, sets up a sequence of equations where working backwards amounts in a reduction of the column of \( a \)'s and \( c \) into the values \( a' \), \( b' \) which yield \( a, b \).

Comment 1: It should be noted that the *Aryabhatās* algorithm provides the solution to the class of problems defined by \( x_0 - x_1 = e \), which is more general than the Chinese Remainder problem described first in San Tōn San Chōng (Master Sun's Arithmetical
Manual), which, according to Needham, was written between 280 a.d. and 473 a.d. The problem reads:

We have a number of things, but do not know exactly how many. If we count them
by threes we have two left over. If we count them by fives we have three left over.
If we count them by sevens we have two left over. How many things are there?

Sun Tzu determined the "use numbers" 70, 21, and 15, these are multiples of 5 × 7, 3 × 7
and 3 × 5, and have the remainder 1 when divided by 3, 5 and 7, respectively. The sum
2 × 70 + 3 × 21 + 2 × 15 = 233 is the answer, and by setting out a multiple of 3 × 5 × 7
(=105) as many times as possible (in this case, twice) the least answer, 23, is obtained.
In the eighth century a.d. I-Hsing used the method for solving calendar problems,
and in the thirteenth century a.d. Chhin Chiu-Shao gave a full explanation.

Sun Tzu's problem also occurs in identical words as Problem 5 of the 'supple-
mentary problems' printed by Hoche in his edition of the 'Introduction to Arithmec'
of Nikomachos of Gerasa. According to Needham this problem occurs in only two or
three of the nearly fifty extant manuscripts of Nikomachos. Three of the five 'supple-
mentary problems' are ascribed to the monk Isaac Argyros (14th century a.d.), so it
seems reasonable to assume that Problem 3 was also added by Argyros or his contem-
poraries.7

The stanza (A2. 31) immediately preceding the pulverizer presumably represents
the motivation for the method. It deals with an astronomical problem:

2:31. The two distances between two planets moving in opposite directions is
divided by the sum of their daily motions. The two distances between two planets
moving in the same direction is divided by the difference of their daily motions.
The two results (in each case) will give the time of meeting of the two in the past
and in the future.8

This represents a concern different to that of Sun Tzu.

Brahmagupta in his Siddhanta describes a similar problem: "What number, divided
by 6 has a remainder of 5, and by 5 a remainder of 4, and by 4 a remainder of 3, and
by 3 a remainder of 2?" Brahmagupta and Bhaskara II showed how Aryabhata's
general method to solve linear indeterminate equations could be used to solve the
problem. As mentioned before, Aryabhata's algorithm can solve problems more
generally than the Chinese Remainder problem.

We note that Aryabhata example is very different from that of Sun Tzu or
Nikomachos. The main motivation to consider such problems in India was the cyclic
cosmological system related to the Siddhya School (700 a.c.). It appears, therefore,
that the Indian tradition of solving congruence problems was independent of the
Chinese and was perhaps older.
The recent demonstration that the quotients can be combined in the forward direction (in contrast to the backward direction as described above) resulting in a faster procedure raises the question if Āryabhaṭa’s stanzas admit the new interpretation. This question deserves a thorough investigation.

Some examples of the application of the algorithm are now described.

Example 1: Solve for \( x \) when:
\[
\begin{align*}
x & \equiv x_1 \pmod{63}, \\
x & \equiv x_2 \pmod{100}, \\
\text{and } x_3 - x_4 & = 70.
\end{align*}
\]

Solution:
\[
\begin{array}{c}
63 | 100 \\
63 \\
37 | 3 \\
37 \\
26 | 1 \\
26 \\
11 | 2 \\
22 \\
4 | 2 \\
8 \\
3 | 1 \\
1
\end{array}
\]

The sequence of quotients is 1, 1, 2, 2, 1. We can now apply the Āryabhaṭa algorithm.

\[
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1890 \\
1 & 1 & 1 & 1 & 1 & 1190 & 1190 \\
1 & 1 & 1 & 1 & 700 & 700 \\
2 & 2 & 2 & 490 & 490 \\
2 & 2 & 1 & 210 & 210 \\
1 & 70 & 70 \\
70 & 70 & 0
\end{array}
\]

Thus \( x' = 1890, b' = 1190 \).
Now using (4):

\[ x = 1890 \mod 100 = 90 \]
\[ \delta = 1190 \mod 61 = 56. \]

Therefore,

\[ x = 30.65 + x_1 \]
\[ = 56100 + x_6. \]

Let \( x_1 = 2, x_6 = 72, \) then

\[ x = 5672. \]

**Example 2:** Solve for \( x \) when:

\[ x \mod 26 = 18 \]
\[ x \mod 57 = 11. \]

**Solutions:** The sequence of quotients is 1, 2, 2, 1. The value of \( x \) is 11 - 18 = -7. We form the table (5):

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
2 & 2 & 2 & -49 \\
& 2 & -21 & -21 \\
1 & -7 & -7 & -7 \\
0 & & & \\
\end{array}
\]

Therefore,

\[ x = -70 \mod 57 = 4 \]
\[ x = -49 \mod 26 = 3. \]

and

\[ x = 4.26 + 18 \]
\[ = 3.37 + 11 = 122. \]

There is no need to introduce negative numbers in this example if one changes the order of the congruences; we have done so to show that the algorithm works irrespective of the sign of numbers.

Āryabhaṭa's algorithm was generally used to solve problems in astronomy. The following problem by Brahmagupta (born 598 A.D.) which appears in his Brahmasphatsi-padi-bahdhatu (628 A.D.) (The Revised System of Brahmas) is an example:
Suppose that viewed from the earth the sun, moon, etc. have travelled for the following number of days after completing full revolutions since the beginning of the Kalpa (when the sun and the planets were collinear):

<table>
<thead>
<tr>
<th></th>
<th>Sun</th>
<th>Moon</th>
<th>Mars</th>
<th>Mercury</th>
<th>Jupiter</th>
<th>Saturn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>1000</td>
<td>41</td>
<td>315</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>

Given that the sun completes 3 revolutions in 1096 days, the moon 5 revolutions in 137 days, Mars 1 in 685 days, Mercury 13 in 1096 days, Jupiter 3 in 10960 days, Saturn 1 in 10960 days, find the number of days elapsed since the beginning of the Kalpa.

The solution to this problem can be easily seen to be 11960.

**Comment 2:** It can be easily established that in algorithm (3), the multiplication by \( c \) can be made at the very end in the last column. This reduces the computational effort considerably. We describe the modified Āryabhata’s algorithm:

Step 1 : Replace \( x \) by \( i \) in column 1 in (3).
Step 2 : Compute \( a' \) and \( b' \) by algorithm (3).
Step 3 : Replace \( a' \) and \( b' \) by \( ca' \) and \( cb' \) in (4). Compute \( x \) as in (5). \[ x = \frac{b - a}\frac{b}{a} \] \[ (6) \]

**Comment 3:** Āryabhata’s algorithm can be used repeatedly to solve for more than two congruences. Thus if one has three congruences \( A, B, C \) where the moduli are pairwise prime, solve \( A \) and \( B \) first and then use this solution with \( C \) to get the final answer.

**Moduli not relatively prime**

Āryabhata’s algorithm can solve simultaneous congruences with nonrelatively prime moduli (where a solution exists) if one can reduce the congruences to a linear indeterminate equation, where the common factors of the moduli can be divided out.

**Example:** Consider Brahmaγupta’s problem:

\[ x = 5 \mod 6 \]
\[ x = 4 \mod 5 \]
\[ x = 3 \mod 4 \]
\[ x = 2 \mod 3, \]
where we want to solve for smallest \( x \).

**Solution:** The moduli of the first and the second pair are relatively prime. Applying the Āryabhata algorithm:

\[ x = 29 \mod 30 \]
\[ x = 11 \mod 12. \]
We convert this into a linear indeterminate equation by:

\[ x = 30a + 29 = 12b + 11 \]

or

\[ 12b - 30a = 16. \]

which reduces to:

\[ 28 - 5a = 3. \]

Using Aryabhata's algorithm, the least positive solution is:

\[ x = 59. \]

**Complexity of the Algorithm**

We count the number of multiplication, division and addition operations in the Aryabhata algorithm and compare that with those in the standard Chinese Remainder (CR) algorithm.

**Aryabhata Algorithm**

Let \( N \) be the order of the moduli \( d_1 \) and \( d_2 \). Step (2) requires roughly \( \log \_2 N \) divisions. Step (3) requires about \( \log \_2 N \) multiplications and the same number of additions. Step (4) requires 2 divisions, while step (5) requires 1 division and 1 multiplication. This adds up to approximately

\[ 3 + \log \_2 N \text{ divisions} \]

\[ 1 + \log \_2 N \text{ multiplications, and} \]

\[ 1 + \log \_2 N \text{ additions.} \]

(7)

**CR Algorithm**

In this algorithm we first compute \( y_1 \), such that \( d_2 y_1 \mod d_1 = 1 \), and \( d_1 y_2 \mod d_2 = 1 \). Then

\[ x = d_1 y_2 y_1 \cdot d_2 y_1 y_2 \mod d_2 d_1 = n. \]

(8)

If \( \phi (d_1) \) and \( \phi (d_2) \) are known then \( y_1 \) and \( y_2 \) can be obtained in about \( \log \_2 \phi (d_1) \)

\[ \phi (d_1) = \log \_2 N \text{ multiplications and divisions (where the size } N \text{ is taken to be the same as } n), \]

by using Euler's generalization of Fermat's theorem. This complexity is of the same order as in Aryabhata's algorithm.

If \( \phi (d_2) \) are not known, one can use an extension of Euclid's algorithm for computing the greatest common divisor. The number of operations performed in this algorithm is roughly \( 2 \log \_2 d_1 \) multiplications, \( \log \_2 d_1 \) divisions and \( \log \_2 d_2 \) additions (see Knuth or Denning for details). The total number of operations, considering that we must obtain \( y_1 \) and \( y_2 \) and compute (8), is therefore

\[ 1 + \log \_2 N \text{ divisions} \]

\[ 4 + 2 \log \_2 N \text{ multiplications, and} \]

\[ 1 + \log \_2 N \text{ additions.} \]

(9)
We conclude that the Aryabhata and the CR algorithms have about the same complexity.

It appears that algorithmic ideas have pervaded Indian mathematics since the earliest times. The Sidhantasya rules on altar construction amount to arithmetic and algebraic procedures. The logic behind these procedures must have been well understood which would explain why irrational numbers resulting from the use of these procedures were readily accepted. Some of the constructions require solution to simultaneous equations. The Men Pudransa is a procedure to find combinations that was described by Pingala in 200 B.C. Algebra that appears in Aryabhata can be seen to be an extension of the algebra of the Sidhantasya.

CONCLUDING REMARKS

In the late nineteenth century considerable attention was given to the contributions of the ancient Indian mathematicians. That was the age when classical mathematics itself was being formalized, and historians found the Indian sources, in contrast to the greatest concern of the mathematics of the day, lacking in formalization and proof. Ancient Indian mathematics emphasized algorithms and computational techniques, which are constructive procedures. The nineteenth century historians did not consider computational issues and, therefore, many results derived using novel procedures were forgotten as mathematical curiosities.

The motivation for the development of clever algorithms by the ancient Indians was presumably the urge to algorithmize knowledge in the spirit of Plato. A corroboration of this hypothesis is provided by the recent work of C.-O. Seelens who has shown that the chakravala method of Jayadeva and Bhakara II for solving the indeterminate equation of the multiplied square

\[ x^2 - Dy^2 = 1 \]  

(10)

led to a minimum number of steps; in other words, the chakravala method is an optimum algorithm. Seelens notes that "the method represents a best approximation algorithm of minimal length that, owing to several minimization properties, with minimal effort ("economization") and avoiding large numbers, always automatically (without trial processes) produces the least solutions to the equation, and thereby the whole set of solutions... It is accepted that the chakravala method here explained anticipated the European methods by more than a thousand years. But, as we have seen, no European performances in the whole field of algebra at a time much later than Bhakara's, say, nearly up to our times, equalled the marvellous complexity and ingenuity of chakravala".

This shows that the efficiency of the mixed is no accident, and there must have been a deliberate search for powerful computing methods.
COMPUTATIONAL ASPECTS OF ÁRYABHATA ALGORITHM

REFERENCES AND NOTES

[Further text follows, likely discussing references and notes related to the computational aspects of the Áryabhata algorithm, possibly including historical context or specific mathematical details.]