The complexity of greatest common divisor computations

Bohdan S. Majewski[⋆] and George Havas[⋆⋆]

Key Centre for Software Technology
Department of Computer Science
The University of Queensland
Queensland 4072, Australia

Abstract. We study the complexity of expressing the greatest common divisor of n positive numbers as a linear combination of the numbers. We prove the NP-completeness of finding an optimal set of multipliers with respect to either the L_0 metric or the L_∞ norm. We present and analyze a new method for expressing the gcd of n numbers as their linear combination and give an upper bound on the size of the largest multiplier produced by this method, which is optimal.

1 Introduction

Euclid's algorithm for computing the greatest common divisor of 2 numbers is considered to be the oldest proper algorithm known ([Knu73]). It has been much analyzed and various versions exist that exploit different aspects. The extent of the literature on the gcd of just two numbers indicates that this seemingly simple problem has much more to it than meets the eye.

The nature and complexity of computing the gcd of two numbers is reasonably well understood and analyzed. From a variety of traditional algorithms [Knu73,Wat77,dBZ53], through algorithms that obtain excellent performance by exploiting binary representation of integers [Man88,Nor87], to algorithms intended to speed up a computation when the input numbers are very large [Knu73,Sor94], the two number gcd problem enjoys considerable attention. This shows how valuable it is that most of the questions about it are solved positively.

The gcd problem for more than two numbers is interesting in its own right (witness the research level problem in [Knu73]). Furthermore, it has important applications, for example in computing canonical normal forms of integer matrices ([HM93,HM94,Ili89]). However, although there are a few algorithms for computing the greatest common divisor of $n > 2$ numbers ([Bla63,Bra70,Ili89,Wat77]), there was no good understanding of the nature of such computations.

There are a number of problems for which efficient solutions are not readily available. Given a multiset of n numbers, how many of them do we need to take to obtain the gcd of all of them? How can we efficiently find 'good' multipliers

⋆ Email: bohdan@cs.uq.oz.au
⋆⋆ Email: havas@cs.uq.oz.au; partially supported by the Australian Research Council

in an extended gcd computation? How quickly can we obtain the result? We investigate some of these problems and provide concrete answers.

2 On the complexity of finding small multipliers

One of the more challenging issues in any gcd computation is the expression of the result as an integer linear combination of the input. This is done by an extended gcd computation. For example, when divisions are replaced by logical shift operations, the resulting algorithm for just two numbers fails to produce optimal multipliers. Here by optimal we mean a definitely least solution [Lev56], where $\gcd(a_1, a_2)$ is expressed as $x_1a_1 + x_2a_2$ with $|x_1| \leq a_2/2$ and $|x_2| \leq a_1/2$.

The situation is worse if we consider expressing the gcd of n numbers, for arbitrary n, as a linear combination. It is not even clear how 'a definitely least solution' should be defined in this context. A number of sensible interpretations are possible. The collection of numbers is a multiset, which corresponds to the set A and size $s(a) \in Z^+$ of [GJ79, problem SP12]. By ordering the multiset we can conveniently use linear algebra, so that $\sum_{i=1}^{n} x_ia_i$ is simply the dot product of two vectors $\mathbf{x} = [x_1, \ldots, x_n]$ and $\mathbf{a} = [a_1, \ldots, a_n]$. Thus we define an optimal vector $\mathbf{x}$ with respect to a specific metric. An optimal vector with respect to the L_0 metric may represent a very poor solution with respect to the L_2 or the L_∞ norm. For the L_0 metric we at least know that an optimal solution $\mathbf{x}$ will satisfy the upper bound $|\mathbf{x}|_0 \leq \log_2\bigl(\min\{a_1, \ldots, a_n\}\bigr)$. No such bounds were known for other metrics for general n.

We give proofs about the complexity of obtaining optimum multipliers with respect to the L_0 metric and the L_∞ norm. We present a new method for expressing the gcd of n numbers as their linear combination, and prove that it achieves the best possible upper bound on coordinates of $\mathbf{x}$, that is, with respect to the L_∞ norm.

Generally, the numeric variables in this paper are integers. For simplicity we assume that all a_i's are positive, which does not diminish the generality of our considerations. We start with the following problem:

MINIMUM GCD SET

INSTANCE: Multiset A of positive integers, positive integer $K \leq |A|$.
QUESTION: Does A contain a subset R, such that $|R| \leq K$ and $\gcd(r \in R) = \gcd(a \in A)$.

Theorem 1. *The MINIMUM GCD SET problem is NP-complete.*

Proof. Clearly the problem is in NP, as given a set R we can verify in polynomial time that $|R| \leq K$ and that $\gcd(r \in R) = \gcd(a \in A)$ (cf. [Bra70]).

Now we show that there exists a transformation from MINIMUM COVER [GJ79, problem SP5] to MINIMUM GCD SET, such that the answer 'yes' or 'no' for MINIMUM GCD SET is also the right answer for MINIMUM COVER. An instance of MINIMUM COVER is defined by a collection C of m subsets of a finite set S of size n and a positive integer $K \leq |C|$. We show how a polynomial time algorithm for MINIMUM GCD SET solves MINIMUM COVER in polynomial time.

Initially we select the first n prime numbers, say $p_1 = 2$, $p_2 = 3$, …, p_n. Let $C = \{c_1, c_2, \ldots, c_m\}$, where $c_i = \{s_{i_1}, \ldots, s_{i_h}\}$ for $h > 0$. For each $c_i \in C$ construct the number $a_i = \prod_{i=1}^{n} p_i^{\alpha(i,j)}$, $j = 1, \ldots, n$, where $\alpha(i,j) = 0$ if $s_j \in c_i$, and 1 if $s_j \notin c_i$. The size of p_n is $O(\log n \log\log n)$. Therefore the length of each a_i is bounded by $O(n^{1+\epsilon})$ for any $\epsilon > 0$. As there are m such numbers, the size of the new problem is at most $O(mn^{1+\epsilon})$.

The greatest common divisor of the a_i's is, from the definition, equal to

$$p_1^{\min(\alpha(1,1),\ldots,\alpha(1,n))} \cdots p_n^{\min(\alpha(m,1),\ldots,\alpha(m,n))} = \prod_{i=1}^{n} p_i^0 = 1.$$

Now suppose that we have found a subset R with the desired property. As $\gcd(r \in R) = 1$, for each $r \in R$ there is a corresponding $c \in C$, $c = c(r)$ and $\bigcup_{r \in R} c(r) = S$. (To have $\gcd(r \in R) = 1$ we must have the factor p_i^0 in some $r \in R$ for each $i = 1, \ldots, n$, and there is a 1–1 mapping between the p_i's and s_i's.) Thus $C' = C'(R)$ is a solution to MINIMUM COVER. If however $l > K$ numbers are required in order to obtain the correct greatest common divisor there is no subset C' of C with the specified property. □

Next we look at the task of finding small multipliers. First consider the following problem (satisfaction of an L_∞ norm bound):

MINIMUM GCD MULTIPLIERS

INSTANCE: Multiset A of positive integers, positive integer K.
QUESTION: Does there exist a solution to

$$\sum_{i=1}^{n} x_i a_i = \gcd(a_1, \ldots, a_n)$$

such that each $|x_i|$ is bounded by K?

The MINIMUM GCD MULTIPLIERS problem is NP-complete. To prove this we proceed in two steps. First we prove that the restricted case, where each multiplier is either $+1$ or -1, is NP-complete. Then we use the bit-pattern engineering technique of van Emde Boas [vEB81] to build a polynomial time reduction from the restricted problem to the case where each $x_i \in \{-1, 0, +1\}$. A modification of this gives the proof of our claim.

We start by exhibiting a polynomial time transformation from PARTITION [GJ79, problem SP12] to UNEVEN PARTITION (UP). UP asks if, for a given multiset A of n positive integers and positive constant c, there exists a submultiset $A' \subset A$, such that

$$\sum_{a \in A'} a = \sum_{a \in A \setminus A'} a + c.$$

There are various ways to prove the NP-completeness of UP. We use an auxiliary lemma, which is a modification of [vEB81, Lemma 1.2], since we also need it later.

Lemma 1. *Given a multiset of positive integers $A = \{a_1, \ldots, a_n\}$ and a positive constant c, if each a_i can be expressed as $a_i = r_i + Mq_i$, with $M > K \sum_{i=1}^{n} |r_i| + c$, then the equation $\sum_{i=1}^{n} x_i a_i = c$, with each $|x_i|$ bounded by K, is equivalent to the system of two equations*

$$\begin{array}{l} \sum_{i=1}^{n} x_i r_i = c \\ \sum_{i=1}^{n} x_i q_i = 0 \end{array} \qquad \text{subject to } |x_i| \leq K.$$

Proof. Consider the equation $\sum_{i=1}^{n} x_i r_i + M \sum_{i=1}^{n} x_i q_i = c$ and suppose that $\sum_{i=1}^{n} x_i q_i = \alpha$. Then the left hand side of the equation is of the form $\sum_{i=1}^{n} x_i r_i + \alpha M$. As $M > K \sum_{i=1}^{n} |r_i| + c$, the expression $\sum_{i=1}^{n} x_i r_i + \alpha M$ cannot be equal to c unless $\alpha = 0$, since each $|x_i|$ is bounded by K. (The contribution to the expression made by $\alpha K \sum_{i=1}^{n} |r_i|$, for $\alpha \neq 0$, can at best be cancelled by the $\sum_{i=1}^{n} x_i r_i$, leaving a residue exceeding αc.) Hence the only possible solution is expressed by the system of two equations presented above. □

Theorem 2. *UNEVEN PARTITION is NP-complete.*

Proof. Verifying a solution for UP can be done in polynomial time, hence UP $\in$ NP. For a multiset $A = \{a_1, \ldots, a_n\}$, solving UP for the multiset $B = \{b_1, \ldots, b_{n+1}\}$, where $b_i = 0 + Ma_i$, for $1 \leq i \leq n$, and $b_{n+1} = c + M \times 0$, with $M > 2c$ (which is of the form of the problem in Lemma 2) is equivalent to finding a partition of A. This follows from substituting the appropriate r_i and q_i in the equivalent system of equations in Lemma 2. Moreover, it is easy to see that a bounded version of UP, where each x_i must be in the range $[-K, +K]$, for some positive constant K, is also NP-complete. □

Corollary 1. *The instance of extended gcd computation where we ask for all multipliers to be units is NP-complete.*

Proof. This is UNEVEN PARTITION with $c = \gcd(a_1, \ldots, a_n)$.

Next we look at the complexity of WEAK UNEVEN PARTITION (WUP). For a given multiset of integers $A = \{a_1, \ldots, a_n\}$ and a positive constant c, WUP asks for a solution to the following 0-1 integer linear programming problem:

$$\begin{array}{rl} \text{find} & \displaystyle\sum_{i=1}^{n} x_i a_i - \sum_{i=1}^{n} y_i a_i = c \\ \text{subject to} & x_i + y_i \leq 1, \quad \text{for } i = 1, \ldots, n. \end{array}$$

WUP is a variation of WEAK PARTITION, which was proved to be NP-complete by van Emde Boas [vEB81]. Notice however that, unlike WEAK PARTITION, the parasite solution with $x_i = y_i = 0$ is impossible. Our proof closely follows the proof in [vEB81]. Due to the relative inaccessibility of that proof, and a few minor changes specific to our problem, we include the proof here.

Theorem 3. *WEAK UNEVEN PARTITION is NP-complete.*

Proof. Again, verifying a solution for WUP can be done in polynomial time, hence WUP $\in$ NP. To transform a given instance of UP for a set A into WUP we introduce five new numbers for each a_i, as defined below. We choose $M = 2\sum_{i=1}^{n} a_i + c + 1$, and a constant $d > 4$.

$$\begin{array}{llllllll}
b_{i1} = a_i & + M\big(d^{4i-4} & + d^{4i-3} & + 0 & + d^{4i-1} & + 0 & \big) \\
b_{i2} = 0 & + M\big(0 & + d^{4i-3} & + 0 & + 0 & + d^{4i} & \big) \\
b_{i3} = 0 & + M\big(d^{4i-4} & + 0 & + d^{4i-2} & + 0 & + 0 & \big) \\
b_{i4} = a_i & + M\big(0 & + 0 & + d^{4i-2} & + d^{4i-1} & + d^{4i} & \big) \\
b_{i5} = 0 & + M\big(0 & + 0 & + 0 & + d^{4i-1} & + 0 & \big)
\end{array}$$

In b_{n2} and b_{n4} the term Md^{4n} is replaced by Md^0, which wraps around the $4n$-th bit in the d-ary representation of the integral part of these numbers divided by M.

The definition of M allows us to use Lemma 1, while the condition that $d > 4$ avoids overflows from one d-ary digit to another (since each equation involves at most four x_{ij}'s). Repeated application of Lemma 1 replaces a single equation $\sum_{i,j} x_{ij} b_{ij} = c$ by a system of $4n + 1$ equations

$$\begin{array}{lll}
(0) & \sum_i (x_{i1} + x_{i4}) a_i = c & \\
(i1) & x_{i1} + x_{i3} + x_{i-1,2} + x_{i-1,4} = 0, & i = 2, 3, \ldots, n \\
 & x_{11} + x_{13} + x_{n2} + x_{n4} = 0 & \\
(i2) & x_{i1} + x_{i2} = 0, & i = 1, 2, \ldots, n \\
(i3) & x_{i3} + x_{i4} = 0, & i = 1, 2, \ldots, n \\
(i4) & x_{i1} + x_{i4} + x_{i5} = 0, & i = 1, 2, \ldots, n
\end{array}$$

Combining $(i1)$ with $(i2)$ and $(i3)$ gives $x_{i1} + x_{i3} = x_{i-1,1} + x_{i-1,3}$, hence the value of $x_{i1} + x_{i3}$ does not depend on the index i. Furthermore, by discarding number b_{n2}, thus obtaining $x_{11} + x_{13} + x_{n4} = 0$, we force $x_{i1} + x_{i3}$ to assume only one of the three values $\{-1, 0, +1\}$.

We now show that the only possible values for $x_{i1} + x_{i3}$ are -1 or $+1$. Suppose that $x_{i1} + x_{i3} = 0$. This case may happen for only three pairs: $\langle +1, -1 \rangle$, $\langle -1, +1 \rangle$ and $\langle 0, 0 \rangle$. The first two pairs are impossible, as equation $(i4)$ would necessitate $|x_{i5}|$ to be as large as 2. For the last pair we have $x_{i1} + x_{i4} = x_{i1} - x_{i3} = 0$, which cannot possibly be a solution to (0). Thus $x_{i1} + x_{i3}$ must be either -1 or $+1$. It follows that $x_{i1} + x_{i4}$ is also either -1 or $+1$, so equation (0) provides a solution to UP. Consequently, the ability to solve WUP in polynomial time would mean the ability to solve UP in polynomial time. □

Corollary 2. *Given a multiset of n positive integers $A = \{a_1, \ldots, a_n\}$, the task of expressing $g = \gcd(a_1, \ldots, a_n)$ in the form $\sum_{i=1}^{n} x_i a_i = g$ with $x_i \in \{-1, 0, +1\}$, is NP-complete.*

Corollary 3. *Given a multiset of n positive integers $A = \{a_1, \ldots, a_n\}$, the task of expressing $g = \gcd(a_1, \ldots, a_n)$ in the form $\sum_{i=1}^{n} x_i a_i = g$ with $|x_i| \leq K$, is NP-complete.*

Proof.([vEB81]) A variant of the previous reduction is used. The constant d is replaced by Kd, and all terms Md^{4i-j} are replaced by KMd^{4i-j}, except for the numbers b_{i5}. These changes modify equation $(i4)$, which now becomes:

$$(i4')\ K(x_{i1} + x_{i4}) + x_{i5} = 0.$$

Since x_{i5} is bounded by K this enforces $|x_{i1} + x_{i4}|$ to be bounded by 1. By again discarding number b_{n2}, we force x_{n1} to 0, and consequently x_{n4} must take one of the three values: -1, 0, $+1$. If $x_{n4} = 0$ then $x_{11} + x_{13} = 0 = x_{i1} + x_{i3}$, for $i = 1, \ldots, n$. Combining this with the inequality $|x_{i1} - x_{i3}| \leq 1$ proves that no solution is possible for this choice. If x_{n4} is equal to ± 1 so is $x_{i1} + x_{i3}$, and hence $x_{i1} + x_{i4} = x_{i1} - x_{i3} \in \{-1, +1\}$, as required. For our specific corollary, we set $c = \gcd(a_1, \ldots, a_n)$. □

3 Some bounds

We prove that the size of the multipliers for an arbitrary number of numbers can be kept within the same bounds as for just two numbers. This improves significantly on the bound in [Ili89], which grows with the number of numbers. While it would be nice to have a bound which decreases as the number of numbers increases, this is not possible. The multiset $\{2, 2k+1, 2k+1, \ldots, 2k+1\}$, with $k \geq 1$, serves as a simple counterexample. Our bound is constructive, and we start by giving the lemma that forms the basis of our approach.

Lemma 2. *For a pair of positive integers, $\{a_1, a_2\}$, with $\gcd(a_1, a_2) = 1$, there exists a solution to the equation*

$$x_1 a_1 + x_2 a_2 = c \tag{1}$$

with $|x_1| \leq \max(a_2/2, c)$ and $|x_2| \leq a_1/2$.

Proof. The lemma is true if $a_1 = 1$, thus consider the case where $a_1 \geq 2$. Equation (1) may be rewritten as

$$(x_1 + a_2 i)a_1 + (x_2 - a_1 i)a_2 = c$$

By setting $i = \lfloor x_2/a_1 \rceil$, we ensure that the second multiplier, $x_2(i)$ say, is equal to $x_2 - a_1 \lfloor x_2/a_1 \rceil = x_2 \operatorname{rem} a_1$, where rem denotes the least remainder operation. Thus $|x_2(i)| \leq a_1/2$. For the first multiplier, $x_1(i)$, we have

$$\begin{aligned} x_1(i) &= x_1 + a_2 \left\lfloor \frac{x_2}{a_1} \right\rceil \\ &= x_1 + \frac{a_2}{a_1}\left(x_2 - x_2 \operatorname{rem} a_1\right) \\ &= \frac{c}{a_1} - \frac{a_2}{a_1} x_2(i) \end{aligned}$$

If $|x_2(i)| = \frac{1}{2}a_1$ we can select i such that $x_2(i) = +\frac{1}{2}a_1$ and then $x_1(i) = c/a_1 - a_2/2$. If $c \leq a_1 a_2$ then $|x_1(i)| \leq a_2/2$, otherwise $|x_1(i)| \leq c$. Now assume that $|x_2(i)| < a_1/2$. This time we have no choice of sign for $x_2(i)$ and thus a bound on $|x_1(i)|$ is $c/a_1 + |x_2(i)|a_2/a_1$. First we take $c \leq a_2/2$. Then we have, for some positive integer α,

$$\begin{aligned}
|x_1(i)| &\leq \frac{c}{a_1} + \frac{a_2(a_1 - \alpha)}{2a_1} \\
&\leq \frac{a_2}{2a_1} + \frac{a_2}{2} - \frac{\alpha a_2}{2a_1} \\
&= \frac{a_2}{2} - \frac{a_2}{2a_1}(\alpha - 1)
\end{aligned}$$

As $\alpha \geq 1$ we have $|x_1(i)| \leq a_2/2$. If $c > a_2/2$ a similar argument shows that the bound $|x_1(i)| \leq c$ holds. $\square$

Lemma 2 can be obtained as a corollary of [Lev56, Theorem 3]. However, our proof of Lemma 2 also gives us a constant time method for computing small multipliers whenever we have any solution to (1) available. This property is used in the method we describe next.

Theorem 4. *There is an optimal time, optimal space algorithm for computing the extended gcd of n integers $\{a_1, \ldots, a_n\}$, which guarantees that no multiplier is larger than the largest of the numbers divided by 2.*

Proof. We start by describing a binary tree method for computing the gcd of n numbers and then prove that the multipliers produced by the method obey the bound specified in the theorem.

Consider the following algorithm. For n numbers (we may restrict our attention to positive numbers) $\{a_1, \ldots, a_n\}$, the algorithm builds a binary tree on $2n - 1$ nodes, called a gcd tree. Label the nodes in the top to bottom and left to right fashion, so that the children of node i have labels $2i$ and $2i + 1$ and the parent of child j has the label $\lfloor j/2 \rfloor$, as in Fig. 1. We name each node by its label and the letter g, and take the liberty of associating the value in a node with the node name. Nodes g_n to g_{2n-1} (the leaves of a complete binary tree) are set to a_1 to a_n. At the end of the algorithm, node g_1 will hold $\gcd(a_1, \ldots, a_n)$, while nodes g_n to g_{2n-1} will store multipliers x_1 to x_n such that $\sum_{i=1}^{n} x_i a_i = g_1$.

The computation proceeds in two phases, bottom-top and top-bottom. In the bottom-top phase, the algorithm starts at the bottom of the tree. It takes two leaves with the same parent at a time. Let the selected nodes be g_{4i} and g_{4i+1} (Fig. 1). Using the standard extended Euclidean method, the algorithm computes $g_{2i} = y_{4i}g_{4i} + y_{4i+1}g_{4i+1}$ and stores g_{2i}, y_{4i} and y_{4i+1} at the nodes $2i$, $4i$ and $4i + 1$, respectively. The process works up through the levels (and computations at each level can in fact be done in parallel) until the value of g_1 is computed.

In the top-down phase, the algorithm starts at node 4 (here we assume $n \geq 3$) and proceeds down through the levels, taking two nodes with a common parent at a time. Let the selected nodes be g_{4i} and g_{4i+1}. For these two nodes the algorithm computes two multipliers z_{4i} and z_{4i+1}, such that $z_{4i}g_{4i} + z_{4i+1}g_{4i+1} = g_{2i}z_{2i}$.

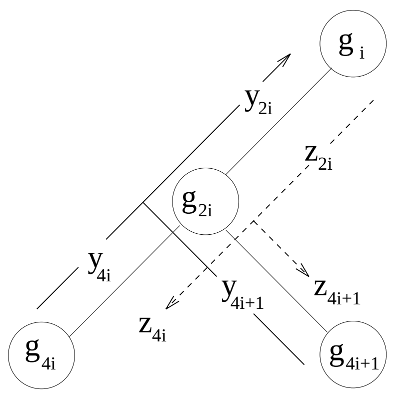

Fig. 1. Part of a gcd tree. Solid lines indicate the actions of the bottom-top phase, while dashed lines follow the top-bottom phase.

The value of z_{2i} is either directly the value of y_{2i} or the value computed in a previous step of the top to bottom phase. To compute z_{4i} and z_{4i+1}, the algorithm uses the fact that $y_{4i}g_{4i} + y_{4i+1}g_{4i+1} = g_{2i}$. The algorithm premultiplies each side by z_{2i} and then uses the technique of Lemma 2 to compute two 'small' multipliers. The phase ends when the multipliers for all leaves of the tree are computed. Finally x_i, for $i = 1$ to n, are set to z_{n+i-1}. From the description of the algorithm it is clear that both the gcd and the multipliers are computed in optimal time and space.

To complete the proof we show that each multiplier z_j computed in the top-bottom phase is no greater than $\max\{a_i\}/2$. The inductive proof uses Lemma 2. Firstly we observe that $g_1 \leq \min(a_1, \ldots, a_n) \leq \max(a_1, \ldots, a_n)$ and both g_2 and g_3, by the properties of the extended Euclidean method for two numbers, have their absolute values less than the maximum number in the input set divided by $2g_1$. Hence the theorem is true for the nodes on levels 1 and 2. Assume that this holds for levels 1 through 2i. Now consider nodes $4i$ and $4i + 1$. The multipliers z_{4i} and z_{4i+1} must solve $z_{4i}(g_{4i}/g_{2i}) + z_{4i+1}(g_{4i+1}/g_{2i}) = z_{2i}$. By the inductive hypothesis, we have $z_{2i} \leq \max(a_1, \ldots, a_n)/2$. By Lemma 2, we can find a pair, z_{4i} and z_{4i+1}, such that either $|z_{4i+1}| \leq g_{4i}/(2g_{2i})$ and $|z_{4i}| \leq g_{4i+1}/(2g_{2i})$ or $|z_{4i+1}| \leq g_{4i}/(2g_{2i})$ and $|z_{4i}| \leq |z_{2i}| \leq \max(a_1, \ldots, a_n)/2$. Thus in either case the hypothesis holds for the new level. □

The fact that our upper bound is optimal relies on an example with repeated numbers. It is reasonable to hope that a better upper bound may apply if all numbers are required to be distinct. However the following theorem indicates problems which need to be addressed in developing an algorithm to achieve a better bound for sets of numbers (as against multisets).

Theorem 5. *The bound achieved by the method of Theorem 4 is the best possible for any method that computes the gcd of a set of numbers by computing gcds pairwise and reducing the resulting multipliers in the natural way using the associated linear equations.*

Proof. The existence an algorithm that could improve the previous bounds while manipulating at most two numbers at a time contradicts [Lev56, Theorem 3]. □

4 Conclusions

In this paper we concentrated our attention on the complexity of expressing the gcd of $n > 2$ numbers as a linear combination of them. We showed that for $n > 2$ the definition of minimum multipliers depends on the selected metric. We proved that the complexity of computing optimal multipliers with respect to the L_0 metric and the L_∞ norm is NP-complete.

On the other hand, we described and analyzed a new method for computing the gcd of $n \geq 2$ numbers, and proved that no multiplier generated by this method will exceed the largest number in the input set divided by 2. This bound is the best possible without additional restrictions on the input. It is also the tightest bound for any method that operates on two numbers at a time.

References

[Bla63] W.A. Blankinship. A new version of the Euclidean algorithm. *Amer. Math. Monthly*, 70:742–745, 1963.

[Bra70] G.H. Bradley. Algorithm and bound for the greatest common divisor of n integers. *Communications of the ACM*, 13:433–436, 1970.

[dBZ53] N.G. de Bruijn and W.M. Zaring. On invariants of g.c.d. algorithms. *Nieuw Archief voor Wiskunde*, I(3):105–112, 1953.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-completness.* W.H. Freeman, San Francisco, 1979.

[HM93] G. Havas and B.S. Majewski. Integer matrix diagonalization. Technical Report TR0277, The University of Queensland, Brisbane, 1993.

[HM94] G. Havas and B.S. Majewski. Hermite normal form computation for integer matrices. Technical Report TR0295, The University of Queensland, Brisbane, 1994.

[Ili89] C.S. Iliopoulos. Worst case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. *SIAM J. Computing*, 18:658–669, 1989.

[Knu73] D.E. Knuth. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms.* Addison-Wesley, Reading, Mass., 2nd edition, 1973.

[Lev56] R.J. Levit. A minimum solution to a Diophantine equation. *Amer. Math. Monthly*, 63:647–651, 1956.

[Man88] D. M. Mandelbaum. New binary Euclidean algorithms. *Electron. Lett.*, 24:857–858, 1988.

[Nor87] G. Norton. A shift-remainder gcd algorithm. In *Proc. 5th International Conference on Applied Algebra, Algebraic Algorithms and Error-correcting Codes – AAECC'87*, Lecture Notes in Computer Science 356, pages 350–356, Menorca, Spain, 1987.

[Sor94] J. Sorenson. Two fast GCD algorithms. *Journal of Algorithms*, 16:110–144, 1994.

[vEB81] P. van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report MI/UVA 81–04, The University of Amsterdam, Amsterdam, 1981.

[Wat77] M.S. Waterman. Multidimensional greatest common divisor and Lehmer algorithms. *BIT*, 17:465–478, 1977.