Journal of the Egyptian Mathematical Society
Volume (26) - Issue (3) - 2018
ISSN:1110-256X
DOI:10.21608/joems.2018.2691.1052

EFFICIENT GENERATION OF SHORTEST ADDITION-MULTIPLICATION CHAINS

Hatem M. Bahig and A. E. A. Mahran

Department of Mathematics, Faculty of Science, Ain Shams University, Cairo, Egypt
hmbahig@sci.asu.edu.eg, h.m.bahig@gmail.com, a.e.a.mahran@gmail.com

Received 26/3/2018 Revised 30/4/2018 Accepted 23/7/2018

Abstract

The aim of this paper is to generalize some results on *addition chains* to *addition-multiplication chains*. The paper concerns with generating shortest addition-multiplication chains. It first presents two methods for generating short addition-multiplication chains. Second, it presents an algorithm for generating a shortest addition-multiplication chain. Then it proposes three main improvements for generating a shortest addition-multiplication chain. The practical results show that the proposed improvements reduce, on the average, the running time and storage of the algorithm by about 95% and 35% respectively for data range $12-20$ bits. Similar practical results are obtained for generating all shortest addition-multiplication chains. Finally, the paper discusses how to apply the algorithm to obtain some results that have been uncovered previously.

keywords: addition chain, addition-multiplication chain, branch and bound algorithm, bounding sequence
MSC: 68W01, 11Y16

1 Introduction

Given a natural number x, an *addition-multiplication chain* [1–3] of length l for x, denoted by AMC(x,l), is a monotonic increasing sequence of numbers $1=a_0,a_1,\ldots,a_l=x$, where for each $0<i\leq l$, a_i is the sum or product of two (not necessary distinct) preceding elements, i.e.,

$$a_i = a_j \overset{+}{*} a_k,\ 0\leq k\leq j\leq i-1 \tag{1}$$

Let $\ell_{AM}(x)$ denotes the shortest length of all possible AMCs for x. If Eq.(1) is replaced by $a_i=a_j+a_k$, then the chain is called *addition chain*, simply AC, i.e., AMCs are extension of ACs. The shortest length of all possible ACs for x is denoted by $\ell_A(x)$. ACs play an important role for integer evaluation [2, 4, 5]. The length of an AC for n measures the number of multiplications needed for computing powers y^x from y, where y is an element of some group, such as Z_n, or elliptic curves, and the allowed operation, in such group, is the product of two previously-computed powers. For example, computing y^{43} can be done with 7 multiplications $y, y^2=y*y, y^3=y^2*y, y^5=y^3*y^2, y^{10}=y^5*y^5, y^{20}=y^{10}*y^{10}, y^{40}=y^{20}*y^{20}, y^{43}=y^{40}*y^3$, using the AC: $1,2,3,5,10,20,40,43$. While it can be computed with 8 multiplications using the AC: $1,2,4,8,16,32,40,42,43$.

Design an efficient algorithm for computing y^x has important applications in cryptography [6, 7].
The problem of generating an AC with shortest length is NP-complete [2] while it remains open for AMCs [3, 4].

The notation of AMCs is firstly introduced by Dobkin and Lipton [1, 2] as a computational model for polynomials evaluation.

It is important to point out that Eq.(1) may hold for more than one pair (j,k), and one operation. For example, suppose that an AMC$(8,5)$ is $1,2,3,4,6,8$. The number 4 has different representations: $4=3+1, 4=2+2, 4=2*2$. Similarly, the number 6 has different representations: $6=4+2, 6=3+3, 6=2*3$. To fix this problem, there are two approaches:

1. As in [5], let j be as large as possible. This guarantees that a_i can be represented uniquely except when $a_i=a_{i-1}+a_{i-1}$, $a_i=a_{i-1}*a_1$, and $i>1$. In this case, one can consider $a_i=a_{i-1}*a_1$.

2. As in [7], associate to the sequence $a_0, a_1, \ldots, a_l$ a sequence $w_1, \ldots, w_l$ of pairs $w_i = (j_i, k_i), 0 \le j_i, k_i < i$ such that for each $0 < i \le l$, $a_i = a_{j_i} \,{}^{+}_{*}\, a_{k_i}$.

For simplicity and most commonly used in practice, we use the first approach.

Each step i, i.e., $a_i = a_j \,{}^{+}_{*}\, a_k$, can be defined by two notations:

(1) star-nonstar: Let $o \in \{+, *\}$. A step i is called *o-star* (or star for o) if $a_i = a_{i-1}\, o\, a_k$, $0 \le k < i$, i.e., $j = i - 1$. Therefore, a step i is called *o-nonstar* if $a_i = a_j\, o\, a_k, 0 \le k \le j \le i - 2$. A special case of o-star is *o-doubling* (or doubling for o) when $j = k = i - 1$. Similarly, a step is called *o-nondoubling* if $a_i = a_j\, o\, a_k, 0 \le j \le i-1, 0 \le k \le i-2$.

If the set of possible types of step i in a partial AMC $a_0, a_1, \ldots, a_{i-1}$ is $\{*\texttt{-star}, +\texttt{-star}\}$, then we call step i a *star*. Similarly, for others types of steps. An AMC(x, l) is called star if every step is star.

For simplicity, when we say that a_i is o-star, for example, it means that step i is o-star.

(2) small-big: Define the function λ_{AM} as follows:

$$\lambda_{AM}(1) = 0; \quad \texttt{and} \quad \lambda_{AM}(n) = \lfloor \log_2 \log_2 n \rfloor,\ n \ge 2.$$

If $\lambda_{AM}(a_i) = \lambda_{AM}(a_{i-1})$, then step $i > 0$ is called *small;* otherwise, (i.e., $\lambda_{AM}(a_{i-1}) = \lambda_{AM}(a_i) - 1$), it is called *big.*

Suppose that $i \ge 1$. The number of small steps in the partial AMC $a_0, a_1, \ldots, a_i$ is denoted by $SS_{AM}(a_0, a_1, \cdots, a_i)$.

For example, Table 1 shows the type and the number of small steps for each step $0 < i \le 6$ in the AMC$(44, 6)$: $1, 2, 3, 9, 11, 33, 44$.

Table 1: Type of each step in the chain $1, 2, 3, 9, 11, 33, 44$.

i	a_i	*star-nonstar step*	*small-big step*	$SS_{AM}(a_0, a_1, \ldots, a_i)$
1	2	+-star, +-double	small	1
2	3	+-star	small	2
3	9	$*$-star, $*$-double	big	2
4	11	+-star	small	3
5	33	$*$-star	big	3
6	44	+-star	small	4

There is a lot of work for studying ACs including determining $\ell_A(x)$ and generating short or shortest ACs, see for examples [5–10], while there are a few papers for studying AMCs [1, 3, 11, 12]. To the best of our knowledge, there is no article studied generation of AMCs. Therefore, our goal in this paper is to study generation of a shortest AMC.

The remainder of this paper is organized as follows. Section 2 mentions some results needed in the paper. Section 3 studies an AMC that includes d $*$-doubling steps. Section 4 describes two methods to generate a short AMCs. Section 5 includes a description of a depth-first branch and bound algorithm to find a shortest AMC. It also includes proposing new bounding sequences to improve the efficiency and the storage of the algorithm. Section 6 presents the implementation of the algorithm and some improvements. Some computational results and remarks that have been uncovered previously are presented in Section 7. Finally, Section 8 presents the conclusion and some open problems.

2 Preliminaries

This section presents some known results needed in this paper.

Proposition 1. *[3]*

1. $\ell_{AM}(x) \geq \log_2 \log_2 x + 1,\ x \geq 2.$
2. $\ell_{AM}(2^{2^x}) = x + 1,\ x \geq 0.$
3. $\ell_{AM}(y^{2^x}) = x + s + 2,\ x \geq 0$ *if* y *is one of the following values:*
   - $2^{2^s} + 1,\ 0 \leq s.$
   - $2^{2^s} + 2^{2^t},\ 0 \leq t \leq s.$
   - $2^{2^s} * 2^{2^t},\ 0 \leq t \leq s.$

Lemma 2. *[3] The last step in a shortest AMC is star.*

Lemma 3. *[3] Suppose that* $a_0, a_1, \ldots, a_i, \ldots, a_l = x$ *is an AMC*(x, l)*. Then* a_i *is star if* $SS_{AM}(a_0, a_1, \cdots, a_{i-3}) = 1$, $a_{i-2} = a_{i-3} * a_j, j < i - 4$ *and* $a_{i-1} = a_{i-3} * a_{i-3}$.

3 $*$-doubling step

This section presents some properties of AMC that includes d $*$-doubling steps.

Theorem 4. *Let* $1 = a_0, a_1, \ldots, a_l = x$ *be an AMC*(x, l) *that includes* d $*$*-doubling, then*

$$x \leq 2^{2^{d-1} fib(l-d+3)} \tag{2}$$

where $fib(k)$ *is Fibonacci sequence defined by* $fib(0) = 0, fib(1) = 1, fib(k) = fib(k-1) + fib(k-2), k > 1.$

Proof. The proof is by induction on l. One can prove the theorem at $l = 1, 2$, and then at $l \geq 3$. Suppose that $l = 1$. Then the AMC is $1, 2$. Clearly, $d = 0$, $fib(4) = 3$, and so Eq.(2) holds. Similarly, if $l = 2$, then Eq.(2) holds since there are two AMCs $1, 2, 4$ and $1, 2, 3$ with $d = 1$, and 0 respectively.
Now suppose that $l \geq 3$.
If step l is a $*$-doubling, then

$$x = a_l = a_{l-1} * a_{l-1} \leq 2^{2(2^{d-2} fib(l-d+3))} = 2^{2^{d-1} fib(l-d+3)}$$

and so Eq.(2) holds.
Otherwise, step l can be a $*$-nondoubling, $+$-nondoubling, or $+$-doubling. The $+$-doubling step can be considered a $*$-nondoubling since $a_l = a_{l-1} + a_{l-1} = 2 * a_{l-1} = a_1 * a_{l-1}$ $(l \geq 3)$. Thus, step l is o-nondoubling, where $o \in \{*, +\}$. Therefore, there exists j satisfies $0 \leq j \leq l - 2$ such that

$$x = a_l \leq a_{l-1} \stackrel{+}{*} a_j \leq a_{l-1} \stackrel{+}{*} a_{l-2} \leq a_{l-1} * a_{l-2} \tag{3}$$

Similar to the step l, there are two cases for step $l - 1$.
Case 1: if step $l - 1$ is a $*$-doubling, then by Eq.(3)

$$x \leq a_{l-2}^3 \leq (2^{2^{d-2} fib(l-d+2)})^3 = 2^{3(2^{d-2} fib(l-d+2))}.$$

Now, since

$$\begin{aligned} 2fib(l-d+3) - 3fib(l-d+2) &= 2fib(l-d+2) + 2fib(l-d+1) - 3fib(l-d+2) \\ &= 2fib(l-d+1) - fib(l-d+2) \\ &= 2fib(l-d+1) - fib(l-d+1) - fib(l-d) \\ &= fib(l-d+1) - fib(l-d) \\ &> 0 \quad \texttt{for all} \quad l - d > 0 \end{aligned}$$

Hence $x \leq 2^{2^{d-1} fib(l-d+3)}$.
Case 2: Otherwise, step $l-1$ is an o-nondoubling, where $o \in \{*, +\}$. Then, by Eq.(3),

$$x \leq 2^{2^{d-1} fib(l-d+2)} * 2^{2^{d-1} fib(l-d+1)} \leq 2^{2^{d-1}(fib(l-d+2)+fib(l-d+1))} = 2^{2^{d-1} fib(l-d+3)}$$

so Eq.(2) holds. □

Corollary 5. *Let $1 = a_0, a_1, \ldots, a_l = x$ be an AMC(x, l) that includes d $*$-doubling and s small steps, then*

$$s < l - d < 3.271s.$$

Proof. Obviously $s \leq l - d$. By Lemma 1, and Theorem 4, it follows that $2^{2^{\lambda_{AM}(x)}} \leq x \leq 2^{2^{d-1} fib(l-d+3)} \leq 2^{2^d \phi^{l-d}}$ since $fib(l-d+3) < 2\phi^{l-d}$ where $\phi = \frac{1+\sqrt{5}}{2}$ when $l - d \geq 0$. Since $\lambda_{AM}(x) + s = l$, it follows that $2^{2^{\lambda_{AM}(x)}} \leq 2^{2^{\lambda_{AM}(x)+s(\frac{\phi}{2})^{l-d}}}$ and so $2 \leq 2^{2^s(\frac{\phi}{2})^{l-d}}$. By applying $\lg\lg$, $0 \leq s + (l-d)(\lg\phi - 1)$. Hence, $s < l - d < 3.271s$ follows from the fact that $\frac{1}{1-\lg\phi} \approx 3.2706$. □

4 Generating Short AMCs

This section presents two methods to generate a short AMC. Similar methods on ACs can be found in [5].

4.1 The $m-$ary Method

The $m-$ary method depends on expressing x as $x = \alpha_0 m^t + \alpha_1 m^{t-1} + \ldots + \alpha_{t-1} m + \alpha_t$, where $0 \leq \alpha_i < m$ for $0 \leq i \leq t = \lfloor \log_m(x) \rfloor$.
The AMC$(x, m - 1 + 2t)$ generated by the $m-$ary method is as follows:
$1, 2, 3, \cdots, m, m\alpha_0, m\alpha_0 + \alpha_1, m(m\alpha_0 + \alpha_1), m(m\alpha_0 + \alpha_1) + \alpha_2, \cdots, m(\cdots(m(m\alpha_0 + \alpha_1) + \alpha_2) + \cdots) + \alpha_t$
Hence,

$$\ell_{AM}(x) \leq m - 1 + 2\lfloor \log_m(x) \rfloor \leq m - 1 + 2\log_m(x) \tag{4}$$

For a fixed number x, the right hand side of Eq.(4) is minimized if m satisfies the condition

$$m(\ln(m))^2 = 2\ln(x) \tag{5}$$

which is obtained by differentiation. Table 2 shows different integer values for x and m for Eq.(5).

Table 2: Different values for x and m in Eq.(5)

range of x	m	*range of x*	m
2	2	$2958 \sim 88154$	6
$3 \sim 15$	3	$88155 \sim 4091319$	7
$16 \sim 162$	4	$4091320 \sim 284005321$	8
$163 \sim 2957$	5	$284005322 \sim 28537793165$	9

For examples,

1. Let $x = 14$. Using Table 2, $m = 3$. Therefore, $t = 2$. It follows that x can be expressed as $x = 14 = 1 * 3^2 + 1 * 3^1 + 2$ where $\alpha_0 = \alpha_1 = 1$, and $\alpha_2 = 2$. By using the $m-$ary method, one can construct the following AMC

$$1, 2, 3 = m, 3 = m\alpha_0, 4 = m\alpha_0 + \alpha_1, 12 = m(m\alpha_0 + \alpha_1), 14 = m(m\alpha_0 + \alpha_1) + \alpha_2.$$

After removing the repeated numbers (since AMCs are monotonic increasing sequences), the AMC$(14, 5)$ is

$$1, 2, 3, 4, 12, 14.$$

2. Let $x = 167$. Then $m = 5$, $t = 3$, and
$$167 = 1 * 5^3 + 1 * 5^2 + 3 * 5^1 + 2.$$
Using the $m-$ary method (and after removing the repeated number $5 = m * \alpha_0$), the AMC$(167, 9)$ is
$$1, 2, 3, 4, 5, 6 = 5 + 1, 30 = 6 * 5, 33 = 30 + 3, 165 = 33 * 5, 167 = 165 + 2.$$

4.2 The Factor Method

Let $1 = a_0, a_1, \ldots, a_r = x$ be a shortest AMC(x, r), and $1 = b_0, b_1, \ldots, b_t = y$ be a shortest AMC(y, t). One can construct AMC$(xy, r + t)$ as follows:
$$a_0, a_1, \ldots, a_r, b_2, b_3, \ldots, b_t, a_r b_t = xy.$$
Sometimes there is a need to remove the repeated numbers and reorder the numbers. This proves the following proposition

Proposition 6.
$$\ell_{AM}(xy) \leq \ell_{AM}(x) + \ell_{AM}(y).$$

For examples:

1. The chain $1, 2, 3, 9, 81, 90$ is AMC$(90, 5)$, while the chain $1, 2, 4, 16, 17$ is AMC$(17, 4)$. Using the factor method (and after reordering the numbers), the chain
$$1, 2, 3, 4, 9, 16, 17, 81, 90, 1530 = 90 * 17$$
is AMC$(1530, 9)$.
2. The chain $1, 2, 3, 9, 81, 90$ is AMC$(90, 5)$, while the chain $1, 2, 3, 9, 11$ is AMC$(11, 4)$. Using the factor method (and after removing the repeated numbers and reordering the numbers), the chain
$$1, 2, 3, 9, 11, 81, 90, 990 = 90 * 11$$
is AMC$(1530, 7)$.

5 Generating Shortest Addition-Multiplication Chains

This section contains two subsections. Section 5.1 proposes a depth-first branch and bound algorithm to search for an AMC with the shortest length. Section 5.2 proposes a new lower bound for each element in any AMC(x, l). The set of lower bounds is called *bounding sequence*. The proposed bounding sequence is used to cut off some elements (and so branches) in the search tree that cannot lead to AMC with shortest length.

5.1 The Algorithm

The proposed algorithm is a depth-first branch and bound algorithm that is similar to algorithms in [13–16] for finding shortest ACs with three differences in:

1. the lower and upper bounds of the shortest length.
2. the set of possible children for any node in the search tree.
3. the bounding sequence, see Section 5.2.

The algorithm starts by computing the lower bound $lb = \lceil \log_2 \log_2(x) \rceil + 1$ of ℓ_{AMC} using Proposition 1-(1), and then generates a short AMC using the m-ary method (see Section 4.1) with length equals $ub = m - 1 + 2\lfloor \log_m(x) \rfloor$. The algorithm extends the partial chain $a_0 = 1, a_1 = 2$ to find a shortest AMC(x, lb), where $lb < ub$. If no AMC(x, lb) is found with $lb < ub$, then the generated m-ary AMC is shortest. Suppose that the current level is cl, $1 \leq cl < lb$ and so the current path is $a_0, a_1, \cdots, a_{cl}$. To extend the search tree, the algorithm first should ensure that the current level $cl < lb$, and then adds the children of a_{cl} and their levels $cl + 1$ onto the stack ST. Then the element and its level at the top of the stack ST are popped and assigned to (cl, a_{cl}). If $a_{cl} = n$, then the algorithm has found a shortest AMC and so the algorithm terminates. Otherwise, the algorithm continues to find an AMC(x, lb). If $i < 2$ after pop the top of the stack, this means that the algorithm exhaustive all search tree and so there is no an AMC(x, lb). Therefore, the algorithm should increase the depth search by 1, i.e., $lb = lb + 1$, and repeats the previous steps. The following algorithm is for finding a shortest AMCs.

*Algorithm **GSAMC**: generating a shortest AMC*
Input: $x > 2$.
Output: AMC$(x, \ell_{AM}(x))$
Begin
  set the lower bound lb of $\ell_{AM}(x)$ to $\lceil \log_2 \log_2 x \rceil + 1$
  set the upper bound ub of $\ell_{AM}(x)$ to the length of the short AMC generated by
      the m-ary method.
  while $(lb < ub)$ **loop**
    set a_0 and a_1 to 1 and 2 respectively
    add the pair $(0, a_0)$ onto the stack ST
    add the pair $(1, a_1)$ onto the stack ST
    set the current level cl to 1
    do
      if $(cl < lb)$ **then**
        add the possible children a_{cl+1} of a_{cl} and their levels $cl + 1$ onto the stack ST
      end if
      Pop and assign to (cl, a_{cl}) the top of the stack ST
      if a_{cl} is equal to x **then**
        output the shortest AMC$(x, cl) : a_0, a_1, \cdots, a_{cl} = x$
      end if
    while $cl > 1$
    increment lb by 1
  end while
  output the generated AMC of length ub by the m-ary method.
End.

One of the most important steps to speed up the algorithm is reducing the set of possible children of a node a_i

$$\{a_i < a_j \stackrel{+}{*} a_k \leq x; 0 \leq j, k \leq i\}.$$

Generation of the children of a node a_i takes $O(i^2)$ steps, where every step includes addition and multiplication of two numbers. The next subsection proposes a bounding sequence to minimize the number of possible children.

5.2 Bounding Sequences

Suppose that we want to generate an AMC(x, l). A set of positive numbers $\{b_i\}_{i=0}^{l}$ is called a bounding sequence of length l for x, denoted by **BSeq**(x, l), if $b_i \leq a_i$ for every AMC(x, l) $a_0, a_1, \ldots, a_l = x$. This subsection proposes a new bounding sequence as follows.

Since $a_i \leq a_{i-1}^2$, define a **BSeq**(x, l) as follows:

$$b_i = \lceil x^{2^{-(l-i)}} \rceil,\ i = l, l-1, \cdots, 0 \tag{6}$$

It is easy to see that

$$b_l = x;\ \ b_i = \lceil \sqrt{b_{i+1}} \rceil,\ i = l-1, \cdots, 0,$$

and so each b_i can be easily computed from b_{i+1}.

Now, Theorems 7 and 9 show that the proposed bounding sequence Eq.(6) can be used to cut off all paths generated from the path $a_0, a_1, \ldots, a_i$ if $a_i < b_i$ or $a_i * a_{i-1} < b_{i+1}$.

Theorem 7. *Suppose that $\{b_i\}_{i=0}^{l}$ is a **BSeq**(x, l) defined by Eq.(6). Then the partial AMC $a_0, a_1, \ldots, a_i$ cannot be extended to an AMC(x, l) if $b_i > a_i$.*

Proof. Suppose that there is a partial AMC $a_0, a_1, \ldots, a_i$ such that $a_i < b_i = \lceil x^{2^{-(l-i)}} \rceil$. Whether $x^{2^{-(l-i)}}$ is an integer or not, $a_i < x^{2^{-(l-i)}}$. Thus,

$$x = a_l \le a_i^{2^{l-i}} < (x^{2^{-(l-i)}})^{2^{l-i}} = x.$$

This is a contradiction, i.e., it is impossible to obtain x from a_i in $l-i$ steps. Therefore, $a_0, a_1, \ldots, a_i$ cannot be extended to AMC(x, l). □

For example, let $x = 16$. Using Eq.(6), **BSeq**$(x, 3)$ is $\{b_i\}_{i=0}^{3} = \{1, 2, 4, 16\}$. The algorithm **GSAMC** starts with the partial chain $1, 2$. Then **GSAMC** generates the set of possible children of $a_1 = 2$ which is $\{3, 4\}$. Using the bounding sequence $\{1, 2, 4, 16\}$, the set of possible children of a_1 is reduced to $\{4\}$ since $3 < b_2 = 4$. Thus, **BSeq**$(x, 3)$ cuts off the path $1, 2, 3$ from the search space.

Remark 8. *The **BSeq** defined by Eq.(6) may contain $b_{i-1} = b_i$ for some $2 \le i$. For example, **BSeq**$(921, 7)$ is $\{b_i\}_{i=0}^{7} = \{1, 2, 2, 2, 3, 6, 31, 921\}$. Clearly, $b_1 = b_2 = b_3$. Therefore, **BSeq** can be updated to $1, 2, 3, 4, 5, 6, 31, 921$. In general, **BSeq** can be updated to be monotonic increasing as follows:*

set j to 2
while $(b_{j-1} \ge b_j)$ **loop**
    set b_j to $b_{j-1} + 1$
    set j to $j + 1$
end while

Theorem 9. *Let $x \ne y^{2^\alpha}, \alpha \ge 0, y \ge 2$ be a natural number and $\{b_i\}_{i=0}^{l}$ be **BSeq**(x, l) defined by Eq.(6). If there exist a step $2 \le i \le l-1$ in AMC(x, l), such that $b_{i+1} > a_i * a_{i-1}$, then the partial AMC $a_0, a_1, \ldots, a_i$ cannot be extended to an AMC(x, l).*

Proof. Clearly, all steps after step i cannot be $*$-doubling since $x \ne y^{2^\alpha}, \alpha \ge 0, y \ge 2$. Suppose that there exists a step $s > i$ that is an $o-$nondoubling, where $o \in \{+, *\}$. Note that the $+$-doubling step can be considered $*$-nondoubling.

Therefore there exist j, and k satisfy $0 \le j \le s-1$ and $0 \le k \le s-2$ such that

$$\begin{aligned} a_s &= a_j \overset{+}{*} a_k \le a_{s-1} \overset{+}{*} a_{s-2} \le (a_i)^{2^{s-i-1}} \overset{+}{*} (a_{i-1})^{2^{s-i-1}} \\ &\le (a_i)^{2^{s-i-1}} * (a_{i-1})^{2^{s-i-1}} \quad \texttt{since } i \ge 2 \\ &\le (a_i * a_{i-1})^{2^{s-i-1}} \end{aligned}$$

If $a_{i-1} * a_i < b_{i+1} = \lceil x^{2^{-(l-(i+1))}} \rceil$, then

$$x = a_l \le (a_s)^{2^{l-s}} \le ((a_i * a_{i-1})^{2^{s-i-1}})^{2^{l-s}} = (a_i * a_{i-1})^{2^{l-i-1}} < (x^{2^{-l+i+1}})^{2^{l-i-1}} = x$$

a contradiction. Therefore, the partial AMC $a_0, a_1, \ldots, a_i$ cannot be extended to an AMC(x, l). □

The following is an example on Theorem 9. Using Eq.(6), **BSeq**$(63, 4)$ is $\{b_i\}_{i=0}^{4} = \{1, 2, 3, 8, 63\}$. Using Theorem 7, the set of possible children of $a_1 = 2$ in the partial chain $1, 2$ is $\{3, 4\}$. But, by using Theorem 9, the set of possible children of $a_1 = 2$ is $\{4\}$ since $a_1 * a_2 = 2 * 3 < b_3 = 8$. Thus, Theorem 9 cuts off the branch $1, 2, 3$ from the search tree. Similarly, the set of possible children of $a_2 = 4$ in the partial AMC $1, 2, 4$ is $\{8, 16\}$ by using Theorem 7, while it is $\{16\}$ by using Theorem 9.

Remark 10. *The condition "$b_{i+1} > a_i * a_{i-1}$" in Theorem 9 cannot be replaced by $b_{i+1} > a_i + a_{i-1}$. For example, using Eq.(6),* ***BSeq****$(91, 6)$ is $\{1, 2, 3, 4, 5, 10, 91\}$. Each element a_i in the partial AMC $1, 2, 3, 6, 7, 13$ satisfies $b_i \leq a_i\, (i \geq 0)$ and $b_{i+1} \leq a_i * a_{i-1}\, (i \geq 2)$. Thus, this partial AMC may lead to an AMC$(91, 6)$. The partial AMC leads to the AMC$(91, 6)$: $1, 2, 3, 6, 7, 13, 91 = 13 * 7$, while it cannot lead to an AMC$(91, 6)$ if someone uses $b_{i+1} < a_i + a_{i-1}$ since $91 > 13 + 7$.*

6 Experimental Results

This section reports the experimental study we have performed to generate one (or all) shortest AMC(s) using **GSAMC**. The section reports the impact of using the proposed bounding sequence, and Lemma 2. For generating one shortest AMC, this section also reports the impact of using the so called "Star-NonStar strategy" [16] which is to find an AMC(x, l), first try to find a star AMC(x, l). If no star AMC(x, l) is found, then try to find an AMC(x, l), where each step may be star or nonstar. All implementations were made with the C language and were compiled by gcc compiler. Experiments were conducted on a Pentium IV with 3.2 GHz, and Linux operating system was used to run **GSAMC** and obtain the performance and storage results. We have tested five data sets. The data set is chosen randomly with 200 numbers each of fixed n-bits, where $n = 12, 14, 16, 18$, and 20.

The proposed improvements are

I1: using Theorem 7, i.e., every child a_{i+1} of a_i $(1 \leq i \leq l-1)$ should satisfy $a_{i+1} \geq b_{i+1}$. If $a_{i+1} < b_{i+1}$ for some i, then Theorem 7 cuts off at least $l!/(i+1)!$ paths from the search tree.

I2: using Theorem 9, i.e., every child a_{i+1} of a_i $(1 \leq i \leq l-2)$ should satisfy $a_{i+1} * a_i \geq b_{i+2}$. If $a_{i+1} * a_i < b_{i+2}$ for some i, then Theorem 9 cuts off at least $l!/(i+1)!$ paths from the search tree.

I3: using Lemma 2, i.e., the last step $i = l$ is star. **GSAMC** restricts the generation of children at step l to star elements that are equal to x. This improvement speeds up the generation of the last element a_l in the chain by $(l-1)(l-2)/2$ steps, where each step contains addition and multiplication of two numbers in the current path $a_0, a_1, \ldots, a_{l-1}$. Clearly, this improvement doesn't change the size of the stack.

Table 3 presents the average of the execution time in seconds and the obtained improvements to find a shortest AMC by using **GSAMC** and its improvements. While Table 4 presents the average of the maximum number of elements in the stack and the obtained improvements to find a shortest AMC for the same data set used in Table 3.

The data in Tables 3 and 4 show that following results:

1. the improvements **I1** and **I3** have a good impact to improve the execution time of **GSAMC**, while the improvement **I2** has a small impact. The performance of **I2** over **I1** is about $2 \sim 3\%$, while the performance of **I3** over **I1** and **I2** is about $17 \sim 20\%$.
2. the improvements **I1**, **I2**, and **I3** together reduce the execution time of **GSAMC** by about 95%.
3. the improvement **I1** has a good impact to reduce the memory storage of the stack. It reduces the storage by about 35%. On the average, the improvement **I2** has almost no effect in reducing the storage of the stack. The improvement **I3** doesn't change the storage of the stack.

The data in Tables 5 and 6 show the impact of using Star-NonStar strategy to find a shortest AMC. One can conclude the following:

1. the strategy improves the execution time of **GSAMC** by about 50%. In other side, it has a very small impact (about $1 \sim 2.5\%$) when it applied with the improvements **I1**, **I2**, and **I3**.

2. the strategy reduces the memory storage of the stack by about 20%. It also reduces the memory storage of the stack by about 15% when it applied with the improvements **I1**, **I2**, and **I3**.

3. although uses of the strategy improves the memory storage of the stack and the execution time of **GSAMC**, the uses of the strategy with **GSAMC** has less impact compared to uses of the improvements **I1**, **I2**, and **I3** with **GSAMC**.

Finally, Tables 7 and 8 show the impact of using the improvements **I1**, **I2**, and **I3** to find all shortest AMCs. For generating one or all shortest AMC, the data in Tables 3, 4, 7, and 8 show that the percentages of the improvements are almost the same.

Table 3: The average of execution times (in seconds) to find a shortest AMC by using **GSAMC** and its improvements

GSAMC	n–bits				
with	12	14	16	18	20
	0.15	1.04	12.53	73.42	695.25
I1	0.05	0.29	3.25	17.58	150.12
	66.6%	72.1%	74%	76%	78.4%
I1+I2	0.04	0.26	2.86	15.75	132.41
	73.3%	75%	77.1%	78.5%	80.9%
I1+I2+I3	0.01	0.05	0.58	3.01	24.22
	93.3%	95.1%	95.3%	95.9%	96.5%

Table 4: The average of the memory storage (maximum length of the stack) needed for the stack to find a shortest AMC using **GSAMC** and its improvements

GSAMC	n–bits				
with	12	14	16	18	20
	79.66	98.27	126.86	148.65	180.08
I1	51.32	62.91	82.63	96.67	116.32
	35.5%	35.9%	34.8%	34.9%	35.4%
I1+I2	51.26	62.71	82.54	96.51	116.21
	35.6%	36.1%	34.9%	35.0%	35.4%

Table 5: Comparison (in execution time in seconds) between different strategies to find a shortest AMC.

GSAMC	n–bits				
with	12	14	16	18	20
	0.15	1.04	12.53	73.39	695.25
Star-NonStar strategy	0.07	0.6	5.42	37.39	334.7
	53.3%	42.3%	56.7%	49%	51.8%
I1+I2+I3	0.01	0.05	0.58	3.01	24.22
	93.3%	95.1%	95.3%	95.9%	96.5%
I1+I2+I3 and	0.006	0.04	0.29	1.59	12.64
Star-NonStar strategy	96%	96.1%	97.6%	97.8%	98.1%

Table 6: Comparison (in the maximum length of the stack) between different strategies to find a shortest AMC.

GSAMC	n–bits				
with	12	14	16	18	20
	79.66	98.27	126.86	148.65	180.08
Star-NonStar strategy	60.72	76.8	100.03	119.19	144.56
	23.7%	21.8%	21.4%	19.8%	19.7%
I1+I2+I3	51.26	62.71	84.74	96.51	116.21
	35.6%	36.1%	34.9%	35.0%	35.4%
I1+I2+I3 and	40.04	48.12	62.72	73.84	94.15
Star-NonStar strategy	49.7%	51%	50.5 %	50.3%	47.7%

Table 7: The average of execution times (in seconds) to find all shortest AMCs by using **GSAMC** and its improvements

GSAMC	n–bits				
with	12	14	16	18	20
	1.22	11.54	99.54	990.26	8549.3*
I1+I2+I3	0.08	0.62	4.59	38.45	251.35
	93.4%	94.6%	95.3%	96.1%	97%

$*$ The average of 50 numbers.

7 Computational Results

Implementing **GSAMC** with some modifications enables one to reveal some properties of AMCs had not previously been observed. For examples,

1. besides pushing $i+1$ and a_{i+1} in the stack ST, **GSAMC** also pushes the type of the element ($*$-star, $*$-doulbling, $\cdots$). This enables one to study whether Lemma 3 can be generalized or not.
2. allowing **GSAMC** to find all shortest AMCs enables one to answer some questions such as:
   - Does there exist a shortest AMC that starts with $1, 2, 4$ for any number n?
   - Does there exist a shortest AMC for any number n such that every step is star?

This section reports some observations on AMCs by running **GSAMC** with some modifications.

7.1 Remarks on Lemma 3

The following are remarks on Lemma 3.

1. The condition "$SS_{AM}(a_0, a_1, \cdots, a_{i-2}) = 1$" is necessary since the chain

$$1, 2, 3, 5, 15, 25, 45, 1125, 1170, 1195$$

Table 8: The average of the maximum length of the stack to find all shortest AMCs by using **GSAMC** and its improvements

GSAMC	n–bits				
with	12	14	16	18	20
	88.56	107.47	134.53	157.76	189.94*
I1+I2+I3	56.61	66.81	88.16	104.75	127.05
	36%	37.8%	34.4%	33.6%	33.1%

$*$ The average of 50 numbers.

is AMC(1195, 9), also it is shortest by executing **GSAMC**, where $a_{i-2} = 5$ with $SS_{AM}(a_0, a_1, \cdots, a_{i-2}) \neq 1$ but a_{i+1} is nonstar.

2. The condition "$a_i = a_{i-2}^2$" cannot be replaced by "$a_i = a_j + a_k, k, j \leq i-1$" since

$$1, 2, 4, 16, 64, 65, 80, 5200, 5201, 5266, 10467$$

is an AMC(10467, 10), also it is shortest by executing **GSAMC**, where $SS_{AM}(a_0, a_1, \cdots, a_{i-2}) = SS_{AM}(a_0, a_1, \cdots, a_3) = 1$, a_{i-1} is $*$-star but $a_i = a_5 = 65 = a_4 + a_0$ and $a_{i+1} = 80$ is $+$-nonstar.

3. The condition "$a_{i-1} = a_{i-2} * a_k, k \leq i-3$" cannot be replaced by "$a_{i-1} = a_{i-2} + a_k, k \leq i-3$" since

$$1, 2, 4, 5, 16, 25, 41$$

is an AMC(41, 6), also it is shortest by executing **GSAMC**, where $a_{i-1} = 5 = a_2 + a_0$ is $+$-star but $a_{i+1} = 25 = a_3 * a_3$ is $*$-nonstar.

7.2 $\ell_{AM}(xy)$ and $\ell_{AM}(x)$

Obviously, $\ell_{AM}(2x) \leq \ell_{AM}(x) + 1$. But, does there exist a number x such that $\ell_{AM}(xy) \leq \ell_{AM}(x)$ for some number y? The answer is yes. For examples,

- $\ell_{AM}(8) = \ell_{AM}(16) = 3$ since $1, 2, 4, 8$ is a shortest AMC(8, 3) and $1, 2, 4, 16$ is a shortest AMC(16, 3).
  Let $n = 375$. By running **GSAMC**, $\ell_{AM}(x) = \ell_{AM}(2x) = 6$, where the AMCs $1, 2, 3, 5, 15, 75, 375$ and $1, 2, 3, 5, 25, 30, 750$ are shortest for x and $2x$ respectively.

- For $x = 128$,
$$\ell_{AM}(2x) = \ell_{AM}(256) = 4 < 5 = \ell_{AM}(128) = \ell_{AM}(x)$$
since, by using Propositions 1-(3) and (2), the AMCs $1, 2, 4, 16, 256$ and $1, 2, 4, 16, 64, 128$ are shortest for 256 and 128 respectively.

In fact, $\ell_{AM}(2x) = \ell_{AM}(x)$ for many numbers x by putting $\beta = 0$ in the following proposition.

Proposition 11. *Let* $x = 2^{2^\alpha + 2^\beta}, \alpha > \beta + 1 \geq 1$. *Then* $\ell_{AM}(2^{2^\beta} x) = \ell_{AM}(x)$

Proof. By Proposition 1-(3), $\ell_{AM}(2^{2^\alpha + 2^\beta}) = \alpha + 2$.
Since $2^{2^\beta} x = 2^{2^\alpha + 2^{\beta+1}}$, it follows that $\ell_{AM}(2^{2^\beta} x) = \alpha + 2 = \ell_{AM}(x)$ by Proposition 1-(3). □

7.3 Star-AMCs

Let $\ell^*_{AM}(x)$ denotes the length of shortest star AMCs for x. Thus, $\ell_{AM}(x) \leq \ell^*_{AM}(x)$.

By running **GSAMC**, the first number with $\ell_{AM}(x) < \ell^*_{AM}(x)$ is $n = 281$, where $\ell_{AM}(281) = 7$ and $\ell^*_{AM}(281) = 8$. By running **GSAMC**, there is only one shortest AMC(281), see Table 9. The next five numbers with $\ell_{AM}(x) < \ell^*_{AM}(x)$ are $913, 941, 996, 997$ and 998 with shortest lengths are $7, 8, 7, 8$, and 8 respectively (see Table 9).

Table 9: All shortest AMCs for $281, 913, 941, 996, 997$ and 998. The underlined numbers are $*$-nonstars

x	*List of shortest AMCs*	x	*List of shortest AMCs*
281	1, 2, 4, 5, 16, 25, 256, 281	913	1, 2, 3, 9, 11, 81, 83, 913
941	1, 2, 4, 5, 25, 29, 100, 841, 941	996	1, 2, 3, 9, 12, 81, 83, 996
997	1, 2, 3, 9, 12, 81, 83, 996, 997	998	1, 2, 3, 9, 12, 81, 83, 996, 998

7.4 The branches $1, 2, 4$ and $1, 2, 3$

Shortest AMCs may have the partial chains $1, 2, 4$ or $1, 2, 3$. If shortest AMCs for a number x contain both partial chains, then one can improve generation of a shortest AMC by restricting the search tree to one of the branches $1, 2, 4$ or $1, 2, 3$. Unfortunately, this is not true for all numbers. For examples, there is no shortest AMC starting with the branch $1, 2, 3$ for the number $x = 2^{2^\alpha}, \alpha \geq 0$. While the number $x = 3 \cdot 5^\alpha, \alpha \geq 0$, doesn't have a shortest AMC starting with the branch $1, 2, 4$. See also Table 9 for more examples.

8 Conclusions and Open Problems

This paper has addressed the AMC problem. The contributions, in this paper, consist in (1) extending some theoretical results on ACs to AMCs; (2) adapting two methods for generating short ACs to AMCs; (3) a branch and bound algorithm for generating a shortest ACs to a shortest AMC; (4) proposing a new (and the first) bounding sequence to cut off some branches in the search tree; (5) using of Lemma 2 to speed up the algorithm for generating a shortest AMC; (6) studying the impact of using the Star-NonStar strategy; (7) proposing a simple modification in the stack to discover some new properties, and counter examples on some relations on AMCs.

There are still some interesting open problems related to shortest/short AMCs such as

(1) The number of shortest AMCs, denoted by $\mathbf{NSAMC}(x)$. Determining $\mathbf{NSAMC}(x)$ is an open problem. Does there exist $x \neq 2^{2^\alpha}, \alpha \geq 0$ such that $\mathbf{NSAMC}(x) = 1$? In ACs, the numbers that have only one shortest ACs are 3 and $2^n, n \geq 0$ [17, 18]. In AMCs, there are many numbers that have only one shortest AMC. For examples, see Table 9.

(2) The number of numbers which have a shortest AMC of length r, i.e., $NN_{AM}(r) = |\{x : \ell_{AM}(x) = r\}|$. For examples, $NN_{AM}(0) = 1, NN_{AM}(1) = 1, NN_{AM}(2) = 2, NN_{AM}(3) = 5$, and $NN_{AM}(4) = 16$.

(3) The minimum number which has a shortest AMC of length r, i.e, $MN_{AM}(r) = min\{x : \ell_{AM}(x) = r\}$. For examples, $MN_{AM}(0) = 1, MN_{AM}(1) = 2, MN_{AM}(2) = 3, MN_{AM}(3) = 5$, and $MN_{AM}(4) = 7$.

(4) Proposing another bounding sequence.

(5) There are many methods for generating (short) ACs [6, 7] need to be adapted to AMCs. Examples of such methods are binary, constant length nonzero windows and variable length nonzero windows [6, 7].

Acknowledgement

We would like to thank the referees for their valuable comments.

References

[1] R. Lipton and D. Dobkin, Complexity measures and hierarchies for the evaluation of integers and polynomials. Theoretical Computer Science 3, 349–357 (1976)

[2] P. Downey, B. Leong and R. Sethi, Computing sequences with addition chains, SIAM J. Computing Vol.10, No.3, 638–646 (1981)

[3] H. Bahig, On a generalization of addition chains: Addition-multiplication chains. Discrete Mathematics 308(4): 611-616 (2008)

[4] D. Dobkin and R. Lipton, Addition chain methods for the evaluation of specific polynomials. SIAM J. Computing Vol.9, No.1, 121–125 (1980)

[5] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, Vol.2, 3rd ed., Addison-Wesley, Reading MA, 461–485 (1997)

[6] D. M Gordon A survey of fast exponentiation methods. J. Algorithms 122: 129–146 (1998)

[7] S. Vanstone, P. van Oorschot, A. Menezes, Handbook of applied cryptography, Ch.14, CRC Press, 1st edition (1996)

[8] M. Subbarao, Addition chains–some results and problems, in *Number Theory and Applications,* R.A. Mollin, ed., Kluwer Academic Publishers, Dordrecht, 555–574 (1989).

[9] K. Fathy, Hazem Bahig, A. Ragab, A Fast Parallel Modular Exponentiation Algorithm. Arab J Sci Eng 43, 903-911 (2018)

[10] Hazem Bahig, A fast optimal parallel algorithm for a short addition chain. The Journal of Supercomputing 74(1): 324–333 (2018)

[11] H. Altman, Integer Complexity, Addition Chains, and Well-Ordering. Ph.D. dissertation, University of Michigan, (2014)

[12] T. Saranurak, G. Jindal, Subtraction makes computing integers faster. CoRR abs/1212.2549 (2012)

[13] H. Bahig, Improved Generation of Minimal Addition Chains. Computing 78(2): 161-172 (2006)

[14] H. Bahig, Star reduction among minimal length addition chains. Computing 91(4): 335-352 (2011)

[15] H. Bahig and Hazem Bahig, A new strategy for generating shortest addition sequences. Computing 91(3): 285-306 (2011)

[16] E. Thurber, Efficient generation of minimal length addition chains. SIAM J Comput 28: 1247-1263(1999)

[17] A. Flammenkamp, Integers with a small number of minimal addition chains, Discrete Math. 205, 221–227 (1999)

[18] E. Thurber, Addition chains–an erratic sequence. Discrete Math. **122**, 287–305 (1993)