AN EFFICIENT ALGORITHM FOR COMPUTING SHORTEST ADDITION CHAINS

DANIEL BLEICHENBACHER† AND ACHIM FLAMMENKAMP‡

Abstract. An addition chain for an integer n is an ascending list of integers $1{=}a_0,\ a_1,\ \ldots,\ a_r{=}n$ such that any element of this list except the first one can be represented as the sum of two preceding elements of the list. This paper describes a new efficient algorithm to search for shortest addition chains and summarizes the results that have been found by implementing this algorithm.

Key words. addition chain, reduced directed graph, branch and bound algorithm, $\ell(n)$

AMS subject classifications. 11Y70 , 11B75

1. Introduction. Given a monoid $(\mathcal{X}, \cdot)$ written multiplicatively. For any $x \in \mathcal{X}$ and $n \in \mathbb{N}$ we can define the power x^n as usual by $x^n = \underbrace{x \cdot \ldots \cdot x}_{n}$. In this paper, we will study the problem of computing x^n with a minimal number of multiplications. Minimizing the number of multiplications leads to an efficient way to compute powers. This is especially the case when the cost for one multiplication is close to constant. Such an assumption is often reasonable for example when the powers are computed in a finite field or in the cyclic ring $\mathbb{Z}/m\mathbb{Z}$. These exponentiations are widely used in cryptography. Moreover, since cryptographic applications often use very large fields or rings, the cost for one multiplication is rather high and reducing the number of multiplications needed in exponentiations can significantly improve the speed of a cryptosystem.

One possible way is to compute x^n for $n \in \mathbb{N}$ by the *binary method.* Equation (1.1) is a recursive description of this method.

$$x^n = \begin{cases} x & \text{if } n = 1, \\ x^{n/2} \cdot x^{n/2} & \text{if } n \text{ is even,} \\ x^{n-1} \cdot x & \text{otherwise.} \end{cases} \tag{1.1}$$

This method is reasonably fast. For example, only 7 multiplication are necessary to compute x^{23}. The powers $x^2, x^4, x^5, x^{10}, x^{11}, x^{22}$ and x^{23} are each computed from previous ones using one multiplication. However, the shortest way to compute x^{23} requires only 6 multiplications:

$$x^2 = x \cdot x, \quad x^3 = x^2 \cdot x, \quad x^5 = x^3 \cdot x^2,$$
$$x^{10} = x^5 \cdot x^5, \quad x^{20} = x^{10} \cdot x^{10}, \quad x^{23} = x^{20} \cdot x^3.$$

DEFINITION 1. Let $n, r \in \mathbb{N}$. An *addition chain* for n of length r is an ascending list of integers $a_0, \ldots, a_r$ such that $a_0 = 1$, $a_r = n$ and such that for all $1 \le k \le r$ there exists $i, j < k$ with $a_k = a_i + a_j$. The minimal length of an addition chain for n is denoted by $\ell(n)$. An addition chain for a nonempty, finite set $S \subset \mathbb{N}$ is an addition chain for $\max S$ containing every element of S. The length of a shortest addition chain for a set S is denoted by $\ell(S)$.

†Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, USA (bleichen@research.bell-labs.com)

‡Fakultät für Mathematik, Universität Bielefeld, 33501 Bielefeld, Germany (achim@mathematik.uni-bielefeld.de)

This definition is motivated by the following fact. If $a_0, \ldots, a_r$ is an addition chain for $n = a_r$, we can compute the set$\{x^{a_0}, \ldots, x^{a_r}\}$ from x with r multiplications. This follows from the fact that $x^{a_0} = x$ does not need any computation and for all $1 \le k \le r$ only one multiplication is necessary to compute x^{a_k} given the set $\{x^{a_0}, \ldots, x^{a_{k-1}}\}$, since this set must contain two elements x^{a_i} and x^{a_j} such that $x^{a_i}x^{a_j} = x^{a_i+a_j} = x^{a_k}$.

For $n \in \mathbb{N}$, let $B(n)$ be the addition chain corresponding to the binary method defined by (1.1), i.e. $B(n)$ is the list of integers a_i where x^{a_i} is an intermediary result while computing x^n. Therefore, we can describe $B(n)$ as follows

$$B(n) = \begin{cases} 1 & \text{if } n = 1, \\ B(n/2), n & \text{if } n \text{ is even,} \\ B(n-1), n & \text{otherwise.} \end{cases} \tag{1.2}$$

It is easy to show that the length of $B(n)$ is $\lfloor \log_2(n) \rfloor - 1$ plus the number of bits equal to 1 in the binary representation of n. If we denote this last value by $\nu(n)$, we get the immediate consequence

$$\ell(n) \le \lfloor \log_2(n) \rfloor + \nu(n) - 1 \le 2\lfloor \log_2(n) \rfloor. \tag{1.3}$$

Addition chains have been studied extensively. A good overview is given in [6, p.441–466] and in [12, p.555–574]. Downey, Leong and Sethi show that the problem of finding a minimal addition chain for a set S is NP-complete [3]. Although we are not aware of a proof that the problem of computing $\ell(n)$ given n is NP-complete, there is no known polynomial time algorithm for this problem. Different heuristics for finding short, but not necessarily minimal, addition chains for n large have been proposed. Bos and Coster [2] use a window method. In addition, they give a heuristic method for finding short chains for sets of integers. Yacobi [17] uses a method similar to the Lempel-Ziv data compression algorithm. His method can also be seen as an extension of Bos and Coster's method by allowing windows of different size. A method for computing addition chains under memory restrictions is described in [7]. A method for describing the integers which minimal addition chains have few small steps, has been developed by Flammenkamp [4]. This work extends Theorem B and C of [6, section 4.6.3] and [16]. Problems related to the number of minimal addition chain as well as other related problems are discussed in [15]. The goal of this paper is to describe a new algorithm for computing minimal addition chains efficiently, and to present some results that have been achieved using this algorithms.

This paper is organized as follows. Section 2 describes the graphical representation of addition chains. This representation is a valuable tool that simplifies the description of relations between equivalent chains and is also very useful in proofs. Section 3 describes the outline of the new algorithm. The runtime of this algorithm strongly depends on the quality of heuristics for computing upper and lower bounds on the length of an addition chain. Methods for finding upper bounds are described in Section 4. Such upper bounds are generally computed by constructing a short addition chain using heuristics. It was our experience that known methods for finding short addition chains were sufficient for computing good upper bounds. Section 5 describes some new methods for finding lower bounds. Section 6 describes still another source that can be used to speed up the algorithm. It is often sufficient to search only for addition chains of some special form, since any other addition chain can be transformed into such a chain. Section 7 reports on techniques to get certain values of $\ell(n)$ above the search limit of the program. It also includes a slight relaxation of the presumptions of our algorithm. Finally, section 8 presents new numerical results.

2. Directed acyclic graphs and addition chains. The following representation of addition chains has been introduced by Knuth [6, p.460ff]. Many transformations of addition chains are easier to understand when this representation is used.

A graph G will be described by a quadruple $G = (V, E, \alpha, \tau)$ where V is the set of vertices, E is the set of edges and $\alpha, \tau : E \to V$ are two functions denoting the initial vertex $\alpha(e)$ and the terminal vertex $\tau(e)$ of an edge e. This notation allows directed graphs with parallel edges (i.e. edges with the same initial and terminal nodes). The *in-degree* of a vertex v denotes the number of edges to v and the *out-degree* of a vertex v denotes the number of edges from v. A *source* is a vertex with in-degree 0. If a graph G is acyclic, the number of distinct paths between two vertices is finite. Let the number of paths from a vertex u to a vertex v be $\mathrm{p}(u, v)$. It can be defined recursively as

$$\mathrm{p}(u,v) = \begin{cases} 1 & \text{if} \quad u = v \\ \displaystyle\sum_{\substack{e\in E \\ \tau(e)=v}} \mathrm{p}(u,\alpha(e)) & \text{if} \quad u \neq v \end{cases} \tag{2.1}$$

because the number of paths from u to v is equal to the sum of the number of paths from u to all vertices of G with a directed edge to v.

DEFINITION 2. A directed acyclic graph $G = (V, E, \alpha, \tau)$ with exactly one source w is called *associated* to the addition chain $A = a_0, \ldots, a_r$, iff $\#E - \#V + 1 = r$ and for all $v \in V$ there exists $a_k \in A$ with $\mathrm{p}(w, v) = a_k$.

A directed graph $G = (V, E)$ with unique source $w \in V$ associated to an addition chain for $S \subset \mathbb{N}$ is called *reduced*, iff

(i) for all distinct $v, v' \in V$ we have $\mathrm{p}(w, v) \neq \mathrm{p}(w, v')$. This implies that for all $v \neq w$ holds in-degree$(v) \geq 2$.

(ii) for every vertex $v \in V$ with out-degree$(v) \leq 1$ there exists $s \in S$ with $s = \mathrm{p}(w, v)$.

THEOREM 1. *Let $A = a_0, \ldots, a_r$ be an addition chain for $S \subset \mathbb{N}$. Then there exists a reduced graph $G = (V, E, \alpha, \tau)$ with $\#E - \#V + 1 \leq r$ that is associated to A.*

Proof. We will first show that there exists a directed acyclic graph $\hat{G} = (\hat{V}, \hat{E}, \hat{\alpha}, \hat{\tau})$ with $r+1$ vertices $\hat{V} = \{v_0, \ldots, v_r\}$ and $2r$ edges such that v_0 is the only source of $\hat{G}$ and $\mathrm{p}(v_0, v_i) = a_i$ for $0 \leq i \leq r$.

We can find indices i_k and j_k such that $a_k = a_{i_k} + a_{j_k}$ for all $1 \leq k \leq r$. The graph $\hat{G}$ can then be constructed by introducing one vertex v_k for every a_k. For each $1 \leq k \leq r$ two edges, one from v_{i_k} to v_k and one from v_{j_k} to v_k are created. In particular, $\hat{G} = (\hat{V}, \hat{E}, \hat{\alpha}, \hat{\tau})$ where $\hat{V} = \{v_0, \ldots, v_r\}$, $\hat{E} = \{e_1, \ldots, e_r, \tilde{e}_1, \ldots, \tilde{e}_r\}$, $\alpha(e_k) = v_{i_k}$, $\alpha(\tilde{e}_k) = v_{j_k}$ and $\tau(e_k) = \tau(\tilde{e}_k) = v_k$. The relation $\mathrm{p}(v_0, v_k) = a_k$ follows by induction over k.

Next, we will show that $\hat{G}$ can be transformed into a reduced graph G. This can be done by iteratively deleting the vertices of $\hat{G}$ that do not satisfy the definition of a reduced graph:

Any vertex v not corresponding to an element in S (i.e. $\mathrm{p}(w, v) \notin S$)

(i) having out-degree 0 is deleted including its edges.

(ii) having out-degree 1 is deleted together with the outgoing edge and every edge to v is redirected to the successor of v.

This process deletes at least as many edges as vertices. Thus we have $\#E-\#V+1 \leq r$ for the resulting graph $G=(V,E,\alpha,\tau)$. $\square$

Remarks. Hence, iff every element in the addition chain is necessary for S or a later addition step, we have $\#E-\#V+1=r$. Furthermore, it should be noted that the graph constructed in theorem 1 is not unique when the choice of i_k and j_k is not unique.

Two graphs representing the addition chains for $1,2,3,5,10,20,23$ and $1,2,3,5,10,13,23$ are shown in figure 2.1.

THEOREM 2. *For every directed acyclic graph $G=(V,E,\alpha,\tau)$ with source w there exists an addition chain of length at most $\#E-\#V+1$ for the set $\{\mathrm{p}(w,v) : v \in V\}$.*

Proof. First, we will construct a set S containing the set $\{\mathrm{p}(w,v) : v \in V\}$ such that for all $x \in S \setminus \{1\}$ there exist $y,z \in S$ with $x=y+z$. For all $v \in V$ we construct a set A_v as follows. For the source w we set $A_w=\{1\}$. For a vertex v with in-degree 1 we set $A_v=\emptyset$. Finally, if v is a vertex with in-degree $t \geq 2$ and predecessor[1] $v_1,\dots,v_t$ we construct the set A_v from the following elements

$$\begin{aligned}
&\mathrm{p}(w,v_1)+\mathrm{p}(w,v_2),\\
&\mathrm{p}(w,v_1)+\mathrm{p}(w,v_2)+\mathrm{p}(w,v_3),\\
&\vdots\\
&\mathrm{p}(w,v_1)+\mathrm{p}(w,v_2)+\cdots+\mathrm{p}(w,v_t)=\mathrm{p}(w,v).
\end{aligned}$$

Let $S=\bigcup_{v\in V} A_v$. Then $\mathrm{p}(w,v) \in S$ for all $v \in V$ because we have $\mathrm{p}(w,v)=\mathrm{p}(w,v')$ for all vertices v with in-degree 1 and predecessor v'. All elements of $S \setminus \{1\}$ are therefore the sum of two elements of S. Moreover, we have $\#A_v = \text{in-degree}(v)-1$ for all $v \neq w$. Thus $\#S \leq \#E - \#(V\setminus\{w\})+1$, where we have equality when the sets A_v are disjunct. Hence by sorting S, we get an addition chain of length $\#S-1 \leq \#E-\#V+1$. $\square$

Backed up by Theorem 1 and Theorem 2, we can now introduce equivalence classes on the set of addition chains. The motivation behind these equivalence classes is that we would like to examine only one addition chain out of each equivalence class instead of all members of a class. Depending on the size of the equivalence classes, this will enormously increase the speed of our search algorithms.

At first, we need an unique mapping g from the set of the addition chains $\mathcal{A}$ into the set of reduced graphs $\mathcal{G}$, such that A is associated to $g(A)$. Let $\mathcal{G}^*=g(\mathcal{A})$ be the image of g. It is important to note that $\mathcal{G}^* \neq \mathcal{G}$, since there exist reduced graphs which are not in the image of g, since some addition chains contain an element a_k which can be computed in different ways.

However, the ambiguity above would vanish if the definition of addition chains would include the way each element of a chain is computed (e.g. the two chains $1,2,3,4_{(=1+3)},7$ and $1,2,3,4_{(=2+2)},7$ would then be different addition chains). Then we could also find a surjective mapping from this (enlarged) set of addition chains to the set of reduced graphs.

Whatever definition we choose, we get $g : \mathcal{A} \longmapsto \mathcal{G}^*$. Now we call two addition chains *equivalent*, iff they are mapped to the same reduced graph, i.e. $A \sim B \quad :\Leftrightarrow g(A)=g(B)$.

[1] A vertex may be a multiple predecessor of v if there are parallel edges from this vertex to v. Such a vertex would appear more than once in the list of predecessor.

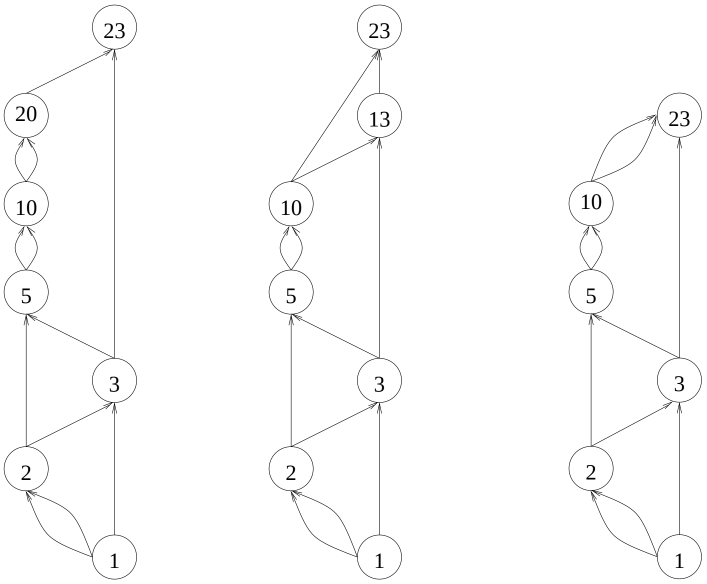

Fig. 2.1. *Graphical representation of addition chains for* 23*. Each label of a vertex* v *corresponds to* $\mathrm{p}(w, v)$ *where* w *is the source of the graph. The two graphs on the left are associated to the addition chains* $1, 2, 3, 5, 10, 20, 23$ *and* $1, 2, 3, 5, 10, 13, 23$ *respectively. The rightmost graph is a reduced graph of the other two graphs. This shows that the two addition chains are equivalent.*

Next, we define a mapping $f : \mathcal{G}^* \longmapsto \mathcal{A}^* \subset \mathcal{A}$, which has the property that $f(G)$ is an addition chain of minimal length among all chains with $g(A) = G$. The set $\mathcal{A}^*$ will be a set of addition chains, which can be interpreted as a set of representatives of the equivalence classes of $\mathcal{A}$. The function f selects an unique A among all $A \in \mathcal{A}$ with $g(A) = G$ for any fixed $G \in \mathcal{G}^*$. This implies that f is a bijection from $\mathcal{G}^*$ to $\mathcal{A}^*$.

If g is chosen appropriately, it is possible to give the mapping f explicitly throughout construction of an addition chain A for a given reduced graph $G = (V, E)$ with only source w, i.e. we simply have to describe a deterministic version of the mapping described in Theorem 2. One canonical way is to order the edges $e \in E$ according to $\mathrm{p}(w, \tau(e))$ and those which have the same terminal node, ascending according to $\mathrm{p}(w, \alpha(e))$. In this order we build up the elements a_k of the addition chain A to be generated and finally sort them into ascending order. Note that this construction assures that the length of A is $\#E - \#V + 1$ and A contains no superfluous elements, because G is reduced. Therefore Theorem 1 assures that a corresponding mapping g exists. Further note that the precise definition of g in the case of a definite choice of indices i_k, j_k for an element a_k must be such that $f(g(A)) = A$ for all $A \in \mathcal{A}^*$.

The elements of $\mathcal{A}^*$ are called *normal forms* of addition chains with respect to f.

3. Outline of the search algorithm. In this section, we describe the outline of an algorithm for computing shortest addition chains. Since this algorithm takes advantage of previously computed information (e.g. the computation of $\ell(n)$ uses the values $\ell(m)$ with $m < n$) it is most efficient when it is used to compute $\ell(n)$ for all n up to a given bound.

One method to find $\ell(S)$ is to compute it recursively from $\ell(S')$ where $\max S' < \max S$. In particular, let S be a nonempty set not equal to $\{1\}$ and let m the maximal element of S. Assume that $a_0, a_1 \ldots a_k$ is a minimal addition chain for S. Then $a_k = m$

and $a_k = a_i + a_j$ for some $i, j < k$. It follows that the chain $a_0, a_1 \ldots a_{k-1}$ is a minimal addition chain for the set $S \cup \{a_i, a_j\} \setminus \{a_k\}$. Thus $\ell(S) = k$ implies that there exists $1 \le x < m$ such that $\ell(S \cup \{x, m-x\} \setminus \{m\}) = k - 1$. Conversely, given an addition chain A for the set $\ell(S \cup \{x, m-x\} \setminus \{m\})$ of length $k - 1$, we can easily construct the addition chain A, m of length k for S.

Using a back-track search, we can thus find an addition chain for a given set S of length k or prove that no such chain exists. To make the search efficient we have to recognize as soon as possible paths that do not lead to a sufficiently small chain. Therefore, we try to prove good lower bounds on the length of minimal addition chains for all sets S' that occur during the search. Altogether, we have the following algorithm:

Search an addition chain of length k for a nonempty set $S \subset \mathbb{N}$.

Given a set S and an upper bound k.

If $S = \{1\}$ and $k = 0$ then return the chain 1.

Compute a lower bound r on $\ell(S)$.

If $r > k$ then return 'no such chain exists'.

Let $m \leftarrow \max S$ be the largest element of S.

Let $R \leftarrow S \setminus \{m\}$.

Check for special cases

Set $x \leftarrow m$ and repeat:

> *Set $x \leftarrow \max\{x' : x' < x, \quad \ell(x') < k\}$.*
> *If $x < m/2$ then return 'no such chain exists'.*
> *Let $S' \leftarrow R \cup \{x, m - x\}$*
> *Search an addition chain of length $k - 1$ for S'*
> *(i.e. executing this algorithm with $k \leftarrow k - 1$ and $S \leftarrow S'$)*
> *If an addition chain A' for S' is found then*
> > *set $A \leftarrow A', m$ and return A.*

Check for special cases is an optional refinement, which will be described in section 6. The selection of the next value of x as the largest x' with $x' < x$ and $\ell(x') < k$ is not necessary and could be replaced by $x \leftarrow x - 1$, but it can be implemented efficiently and results in a large speedup, since the function computing lower bounds is much less frequently performed.

A simple lower bound $lb(S)$ for a set S can be defined by

$$lb(S) := \max_{x \in S} l_0(x) \tag{3.1}$$

where we set $l_0(x) = \ell(x)$ if $\ell(x)$ is known and $l_0(x) = 0$ otherwise. Figure 3.1 shows the search tree for proving $\ell(19) > 5$ when this function lb for computing lower bounds is used. This tree is remarkably small, although lb can still be improved a lot. Using the more precise lower bounds of Section 5 would result in a tree with only 4 nodes. An algorithm that finds the shortest addition chain can now be implemented as follows.

Computation of the length of the shortest addition chain of n.

Use heuristic methods to find short addition chains for n.
Let r be the length of the smallest addition chain found.

Set k to a lower bound for $\ell(n)$.

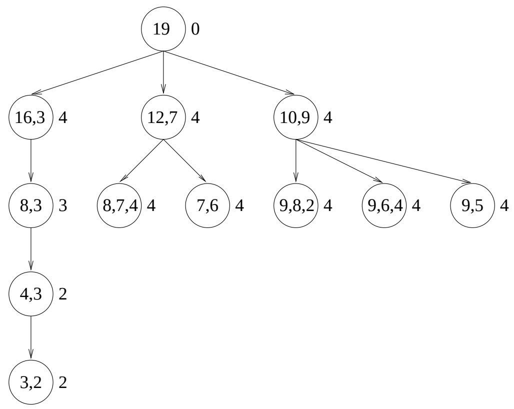

FIG. 3.1. *Search tree for the proof of $\ell(19) > 5$. The nodes of the search tree are labeled with the sets S and the lower bound $lb(S)$.*

While $k < r$ repeat:
    Use a recursive depth-first search which
        either finds an addition chain A for n of length exactly k
        or proves that no such addition chain exists.
    If an addition chain for n was found
        then return k
    Increase k by 1.
Return r.

Assume that the lower bound k for $\ell(n)$ that is computed in this algorithm is smaller than $\ell(n) - 1$. Then some nodes are computed more than once, since the algorithm first proves that there is no addition chain of length k and that there is no addition chain of length $k+1, \ldots$ in the following steps. However, it turns out that proving the nonexistence of an addition chain of length k is generally much less time consuming than proving the nonexistence of an addition chain of length $k+1$ and thus not a big loss of time.

Contrary, it may be a great waste of time trying to prove $\ell(n) > k$ when an addition chain of length k (or even a shorter one) exists. Therefore, it is also important to have good heuristics for finding upper bounds. Such heuristics are discussed in detail in Section 4.

4. Methods for finding upper bounds. The basic strategy for finding upper bounds on $\ell(n)$ is to build addition chains using different heuristic methods and to chose the shortest one as an upper bound on $\ell(n)$. As stated in Section 3, heuristics that often find an addition chain of minimal length may decrease the search time substantially since the search algorithm only has to prove the nonexistence of a shorter addition chain, but will not have to search for an addition chain of length $\ell(n)$ when such a chain is known.

Since we have used different heuristics in our programs and since bad heuristics would result in bad upper bounds, which would only affect the performance of the programs but not their correctness we describe only some of the heuristics used:

- Given an addition chain $a_0, \ldots, a_r$ of length r for $x = a_r$. We can construct a new chains $a_0, \ldots, a_r, a_r + a_i$ where $0 \le i \le r$. This results in a chain for $a_r + a_i$ of length $r + 1$.

- Given addition chains $a_0, \ldots, a_r = x$ and $b_0, \ldots, b_k = y$. Then

$$a_0, \ldots, a_r, a_r b_1, \ldots a_r b_k$$

is an addition chain of length $r + k$ for xy.
- An addition chain for $xy + z$ with $x \geq z$ can be constructed as follows. First, we find an addition chain $a_0, \ldots, a_i = z, \ldots, a_r = x$ for the set $\{x, z\}$ and an addition chain $b_0, \ldots, b_k$ for y. Then

$$a_0, \ldots, a_r, b_1 a_r, \ldots, b_k a_r, b_k a_r + a_i = xy + z.$$

is an addition chain of length $r + k + 1$.
Given n we also have to decide for which x, y, z satisfying $n = xy + z$ we should try this construction. It is reasonable to chose y such that $\ell(y)$ is small, i.e. to try the values $y = 2^i$ and $y = 2^i + 1$ for $1 \leq i < log_2(n)$. Small chains for the set $\{x, z\}$ can for example be found by using the heuristics proposed by Bos and Coster in [2]. Alternatively, optimal chains for a some selected sets can be precomputed and used.

Since we are basically interested in the length of the generated chains it is not necessary to actually construct the chains. The knowledge of the length of the chains used in the construction is sufficient to determine the length of the new addition chain. Let $x, y, z \in \mathbb{N}$. Then we can derive the following upper bounds from the constructions shown so far.

$$\ell(x + y) \leq \ell(\{x, y\}) + 1 \tag{4.1}$$

$$\ell(xy) \leq \ell(x) + \ell(y) \tag{4.2}$$

$$\ell(xy + z) \leq \ell(y) + \ell(\{x, z\}) + 1 \tag{4.3}$$

5. Methods for finding lower bounds. One of the most time consuming part of the search is the proof that there exists no addition chain for a given set and a given length. The search tree can be pruned a lot when good methods to prove lower bounds on $\ell(S)$ are available. We often found that even small looking improvements to the lower bound had significant effects on the running time of the program.

THEOREM 3. *Let S be a set, $\ell(S)$ the length of an minimal addition chain for S and $m, t \in \mathbb{N}$.*

(i) If $m > 2^t \max S$ then $\ell(S \cup \{m\}) \geq \ell(S) + t + 1$.

(ii) If m is odd and $m > 3 \cdot 2^t \max S$ then $\ell(S \cup \{m\}) \geq \ell(S) + t + 3$.

Proof. Let $A = a_0, \ldots, a_r$ be an addition chain for the set $S \cup \{m\}$ and $y = \max S$. y must be an element of the addition chain, say $y = a_j$. It follows $j \geq \ell(S)$ since otherwise we would have found an addition chain for S that is shorter than $\ell(S)$. Since every element in an addition chain is at most twice as large as its predecessor it follows that $a_{j+t} \leq 2^t \max S$. Thus, if $m > 2^t \max S$ then the length of every addition chain for $S \cup \{m\}$ must be at least $j + t + 1$. This shows (i). Moreover, if $t > 0$ and a_{j+t+2} is odd then a_{j+t+2} must be the sum of two different elements and it follows by the preceding argument that $a_{j+t+2} \leq a_{j+t+1} + a_{j+t} \leq 3 \cdot 2^t \max S$. Thus if m is odd and $m > 3 \cdot 2^t \max S$ then the length every addition chain for $S \cup \{m\}$ is at least $j + t + 3$. This shows (ii). ∎

Theorem 3 can be used to compute a lower bound on the length of an addition chain for a set S.

Compute a lower bound on the length of an addition chain for S.

Given a nonempty set $S = \{s_0, \ldots, s_r\} \subset \mathbb{N}$ *where the numbers* s_i *are ascending ordered i.e.* $s_i < s_{i+1}$ *for* $0 \le i < r$.

Set $k_0 \leftarrow \max\{\lceil \log_2(s_0) \rceil\ ,\ \ell(s_0)\}$

Set $i \leftarrow 0$

While $i < r$ *repeat:*

  Increase i *by* 1

  Set $t \leftarrow \lceil \log_2(\frac{s_i}{s_{i-1}}) \rceil$

  If s_i *is odd and* $s_i \ge 3s_{i-1}$ *and* $\log_2(\frac{2s_i}{3s_{i-1}}) > \lfloor \log_2(\frac{s_i}{s_{i-1}}) \rfloor$

    then increase t *by* 1

  Set $k_i \leftarrow \max\{k_{i-1} + t\ ,\ \ell(s_i)\}$

Return k_r

Remarks. Here, we have assumed that $\ell(s_i)$ for all $s_i \in S$ is known, otherwise we could replace $\ell(s_i)$ by any lower bound on $\ell(s_i)$. An attractive lower bound was conjectured by Stolarsky[11] and others [6, p. 451]:

$$\ell(n) \ge \lfloor \log_2(n) \rfloor + \lceil \log_2(\nu(n)) \rceil \quad . \tag{5.1}$$

Thurber has proved this conjecture for $\nu(n) \le 16$ [13] and Schönhage has shown another very close result [9].

6. Transformations into equivalent chains. In many situations, it is possible to show that minimal addition chains of a special form must exist. This can often be done by showing that any addition chain can be transformed into an equivalent chain. In such a case it is sufficient to search only for addition chains of this special form or to show that no chain of this special form exists. Often this results in a much smaller search tree than without this optimization. In other words, we will search only for one addition chain in a class of equivalent chains.

THEOREM 4. *Let* $x \in \mathbb{N}$, S *be a nonempty set of integers,* $\ell(S \cup \{x\}) = k$ *and* $x > \max S$. *Then at least one of the following conditions is true*

- x *is even and* $\ell(S \cup \{x/2\}) = k - 1$.
- *There exists* y *with* $\max S < y < x/2$ *such that* $\ell(S \cup \{y, x - 2y\}) = k - 2$ *and* $(k + 1 - \ell(S \cup \{y\}))y \ge x$.
- $(k + 1 - \ell(S)) \max S \ge x$.

Proof. Let A be an addition chain for $\{x\} \cup S$, let $G = (V, E, \alpha, \tau)$ be the reduced graph for A, let v be the vertex representing x (i.e. $\mathrm{p}(w, v) = x$). Denote by $T = \{e_1, \ldots, e_r\} = \{e \in E : \tau(e) = v\}$ the set of edges whose terminal node is v. Now select the vertex $v' \in \{\alpha(e) : e \in T\}$ that maximizes $y = \mathrm{p}(w, v')$. Since y is maximal we have $ry \ge x$. Moreover, we must have $\ell(S \cup \{y\}) \le k - (r - 1)$ and hence $(k + 1 - \ell(S \cup \{y\})) \max(S \cup \{y\}) \ge x$.

If $y \le \max S$ then it follows that $(k+1-\ell(S)) \max S \ge (k+1-\ell(S \cup \{y\}) \max(S \cup \{y\})$, which validates the third case. On the other hand, if $y > \max S$ then it follows that T contains two edges $e_i \ne e_j$ with $\alpha(e_i) = \alpha(e_j) = v'$ because G is reduced and therefore every vertex not contained in $S \cup \{x\}$ must have out-degree ≥ 2. Thus x can be computed by

$$x = 2y + \sum_{\substack{1 \le h \le r \\ h \ne i, h \ne j}} \mathrm{p}(w, \alpha(e_h)).$$

Now either $T = \{e_i, e_j\}$ in which case we have $x = 2y$ and $\ell(\{y\} \cup S) = k - 1$ or $T \setminus \{e_i, e_j\} \ne \emptyset$. In the later case it follows that $\ell(S \cup \{x, y, x - 2y\}) = \ell(S \cup \{x\}) = k$

and therefore that we can find an addition chain for $S \cup \{x-2y, y\}$ of length $k-2$. □

THEOREM 5. *Let $x \in \mathbb{N}$ and S be a set of integers with $2y \geq x > y > \max S$. If $\ell(x) = \ell(\{x,y\} \cup S) = k$ then $\ell(\{x-y, y\} \cup S) = k-1$.*

Proof. Let A be an addition chain for the set $\{x,y\} \cup S$ of length k. It follows from $\ell(x) = k$ that y is necessary to compute x since otherwise we would get an addition chain for x of length $k-1$ by deleting y in A. From $y > x/2$ follows that there exist integers $y = z_0, z_1, \ldots, z_r = x$ in A such that the differences $b_i = z_{i+1} - z_i$ with $0 \leq i < r$ are contained in A. Since $z_1, \ldots, z_{r-1}$ are all greater than y they are not contained in S. Therefore, we may replace $z_1, \ldots, z_{r-1}$ in A by $b_0 + b_1, \ldots, b_0 + b_1 + \cdots + b_{r-1} = x - y$. After sorting the resulting list we have an addition chain A' for the set $\{x, y, x-y\} \cup S$ of length k since have added as many elements to A as we have deleted. Hence, A' without x is an addition chain for $\{x-y, y\} \cup S$ of length $k-1$ at most. □

The following algorithm represents the improvements to the search algorithm on page 6. Case 1 is an immediate consequence of $S \subset R \Rightarrow \ell(S) \leq \ell(R)$. Case 2 is the implementation of Theorem 5 and case 3 corresponds to Theorem 4.

Check for special cases

case 1: *If* *there exist elements* x, y *in* R *such that* $x + y = m$
    then search an addition chain for R *of length* $k-1$.
    If such a chain A *is found*
        then return A, m
        otherwise return 'no such chain exists'.

case 2: *If* $2 \max R > m$ *and* $\ell(m) = k$
    then let $y \leftarrow \max R$
    search an addition chain for $R \cup \{m-y\}$ *of length* $k-1$.
    If such a chain A *is found*
        then return A, m
        otherwise return 'no such chain exists'.

case 3: *If* $(k + 1 - lb(R)) \max R < m$ *then*
    If m *is even then*
        search an addition chain for $R \cup \{m/2\}$ *of length* $k-1$.
        If such a chain A *is found then return the chain* A, m.
    For all $\max R < y < m/2$ *repeat:*
        If $(k + 1 - lb(R \cup \{y\})y \geq m$ *then*
            search an addition chain for $R \cup \{y, m-2y\}$ *of length* $k-2$.
            If such a chain A *is found then return* $A, m-y, m$.
    Return 'no such chain exists'.

7. Special values of $\ell(n)$. So far, we have described an algorithm that finds all values for $\ell(n)$ up to a given limit x. However, some numbers and their addition chains are more attractive than others. In this section, we describe the methods that have been used to find results on chains for numbers above the limit that has been searched completely.

A first problem which is better solved by searching only for special values rather than computing $\ell(n)$ for all $n < x$ is the computation of the number $d(r)$ of integers n with $\ell(n) = r$. The general approach for this problem is as follows: First, we calculate all values up to 2^t for some given $t \in \mathbb{N}$. In an addition chain $a_0, \ldots, a_k = n$ we have $a_i \leq 2a_{i-1}$ for all $1 \leq i \leq k$ and therefore $\ell(n) \leq t$ implies $n \leq 2^t$. Thus we find $d(r)$

for $0 \le r \le t$ by counting the integers $1 \le n \le 2^t$ for which $\ell(n) = r$. However, we can squeeze some more values of $d(r)$ out of these numbers.

Let us define $s(n)$ by

$$s(n) := \ell(n) - \lfloor \log_2(n) \rfloor$$

and call $s(n)$ the number of *small steps* of n. In order to computed $d(r)$ efficiently, we divide the integers into different sets according to their value $s(n)$. From the definition of $s(n)$ follows

$$d(r) = \sum_{i=0}^{r} \#\{2^i \le n < 2^{i+1} : \; s(n) = r - i\} \quad .$$

In particular, we can compute $d(r)$ for $r \le s + t$ if we know $\ell(n)$ for all $n \le 2^t$ and all $n > 2^t$ which have at most s small steps. The numbers n with $s(n) = 0$ are the powers of 2. Those with $s(n) = 1$ are the sums of two different powers of 2. Knuth describes the numbers with $s(n) = 2$ in [6, Theorem B and C, p. 449] and Flammenkamp extends this result to all numbers n with $s(n) = 3$ in [4, p. 69ff].

Using these results the numbers $2^t < n \le 2^{t+3}$ with $s(n) \le 3$ can be computed very fast. Thus we can find all n with $\ell(n) \le t+3$ and therefore also $d(r)$ for $r \le t+3$. If we knew the structure of all addition chains with four small steps we could go even one step further, but unfortunately, their number grows exponentially and the time to generate them algorithmically seems to increase even stronger. We have calculated $\ell(n)$ for all $n \le 2^{20}$. The sieved numbers up to 2^{23} together with the calculated numbers determine the values of $d(r)$ up to $r = 23$, which are given in table 8.1.

As mentioned at the beginning of section 3 the branch and bound algorithm uses the values $\ell(m)$ for all $m < n$ to calculate $\ell(n)$. But these values are not necessary at all costs, since they are only used to compute lower bounds or if available for upper bounds. If bounds instead of exact values are used, our algorithm still works correctly, but the size of the search tree grows and thus increases the time necessary to calculate $\ell(n)$. On the other hand, we are now more flexible, since we no longer have to compute $\ell(n)$ in sequential order. This idea can, for example, be used to parallelize our program. We also have used this idea to verify that $\ell(2^n - 1) = n - 1 + \ell(n)$ holds for all $1 \le n \le 28$ to extend Thurber's result [14]; however for $n \ge 25$ we had to assume conjecture (5.1). This equation is a stronger form of the Scholz-Brauer [8] conjecture $\ell(2^n - 1) \le n - 1 + \ell(n)$.

The same idea can also be used to find the smallest integer n with $\ell(n) = r$ more efficiently, if a good heuristic method for constructing short addition chains is available. The heuristic method can be used to exclude most candidates for this number efficiently so that only a small numbers of candidates has to be tested.

8. Numerical results. Both authors have independently implemented the presented algorithm on computers and used it to find the shortest addition chains for all $n \le 2^{20}$. We have not tried to exchange any code so that the we could get reliable results by comparing the output of our programs.

The smallest n for which $\ell(n) = r$ is denoted by $c(r)$. Table 8.1 shows the values of $c(r)$. This table extends previously published results. Knuth [6] lists c up to $c(18) = 11\,231$ and Flammenkamp [4] has computed c up to $c(23) = 196\,591$ and Bleichenbacher [1] up to $c(24) = 357\,887$. It was stated in [4, p. 24]: "For $r \to \infty$ it seems that $c(r) = \Theta(\frac{2^r}{r^2})$". At least for $7 \le r \le 27$ holds $11 < \frac{c(r)}{2^r r^{-2}} < 14$. A much better approximation is $\frac{r}{r - \log_2 c(r)} = \Theta(\log_2 r)$, where the upper and lower constants of the Θ-bound for the range $9 \le r \le 27$ can be chosen < 1 and > 0.9. The value $c(r)$ seems to variate randomly around the later approximation in opposite to a more run-away behavior in the first extrapolation.

This table also shows the number $d(r)$ of values n which have shortest addition chains of length r. Knuth [6] lists d up to $d(15)$ and Flammenkamp [4] has computed d up to $d(21)$.

Table 8.2 shows the smallest n for which m with $\ell(\frac{n}{m}) = \ell(n)$. If m is not a power of two then the difference $\ell(\frac{n}{m}) - \ell(n)$ can become arbitrary large [5]. Otherwise, only for $m = 2$ are infinitely many values known with difference zero. Among those the family $n(i) = 2(129 \cdot 2^i + 13)$ with $i \ge 6$ is the only one known with $\nu(n) = 5$ — fewer 1-bits are impossible because all even n with at most three small steps satisfy $\ell(n/2) = \ell(n) - 1$.

None of our results is a counterexample to conjecture (5.1). Table 8.3 gives the smallest values $n = n(\delta)$ for which $\ell(n) - (\lfloor \log_2(n) \rfloor + \lceil \log_2(\nu(n)) \rceil) = \delta$. Using this estimation, we can store $\delta(n)$ instead of $\ell(n)$ to get compressed tables of $\ell(n)$.

It is conjectured in [10] that for all $n \ge 4$ there exists a minimal addition chain for n not containing the element 3. This conjecture is false and the smallest counterexamples are $14759, 15449, 26089$.

r	$c(r)$	$d(r)$	r	$c(r)$	$d(r)$	r	$c(r)$	$d(r)$
0	* 1	1	10	* 127	136	20	34303	49544
1	2	1	11	191	246	21	* 65131	90371
2	* 3	2	12	379	432	22	110591	165432
3	5	3	13	607	772	23	196591	303475
4	* 7	5	14	1087	1382	24	357887	558275
5	11	9	15	* 1903	2481	25	685951	1028508
6	19	15	16	3583	4490	26	1176431	
7	* 29	26	17	6271	8170	27	2211837	
8	47	44	18	11231	14866	28		
9	71	78	19	18287	27128	29		

TABLE 8.1

Values $c(r) = \min\{n : \ell(n) = r\}$ and $d(r) = \#\{n : \ell(n) = r\}$. A $*$ preceding a value $c(r)$ indicates that this is also the smallest $n \in \mathbb{N}$ with $s(n) > s(m)$ for all $m < n$.

m	n	$\nu(n)$	$\ell(n)$	$\frac{n}{m}$
2	382	7	11	191
3	513	2	10	171
5	16385	2	15	3277
6	$*$ 16386	2	15	2731
7	196609	3	19	28087
9	2097153	2	22	233017
10	$*$ 4325410	4	24	432541
11	10485761	3	25	953251
12	$*$ 8388612	2	24	699051
14	$*$ 25165826	3	26	1797559

Table 8.2
Smallest values n with $\ell(\frac{n}{m}) - \ell(n) = 0$. A $*$ preceding a number indicates that this value is also the smallest for $\ell(\frac{n/2}{m/2}) - \ell(n/2) = 1$.

δ	n
1	29
2	3691
3	919627

Table 8.3
Smallest n with $\ell(n) - (\lfloor \log_2(n) \rfloor + \lceil \log_2(\nu(n)) \rceil) = \delta$

At last, figure 8.1 shows the effort to calculate the values $\ell(n)$. For the range $2^{11} \leq n \leq 2^{19}$ we have drawn the number of examined nodes of the search tree of our algorithm. We can observe a characteristic periodicity which is formed by dark stripes, where the points n typically have a fixed value $\ell(n)$. So the slightly ascending period length equals the average increment of n to enlarge the value of $\ell(n)$ by one.

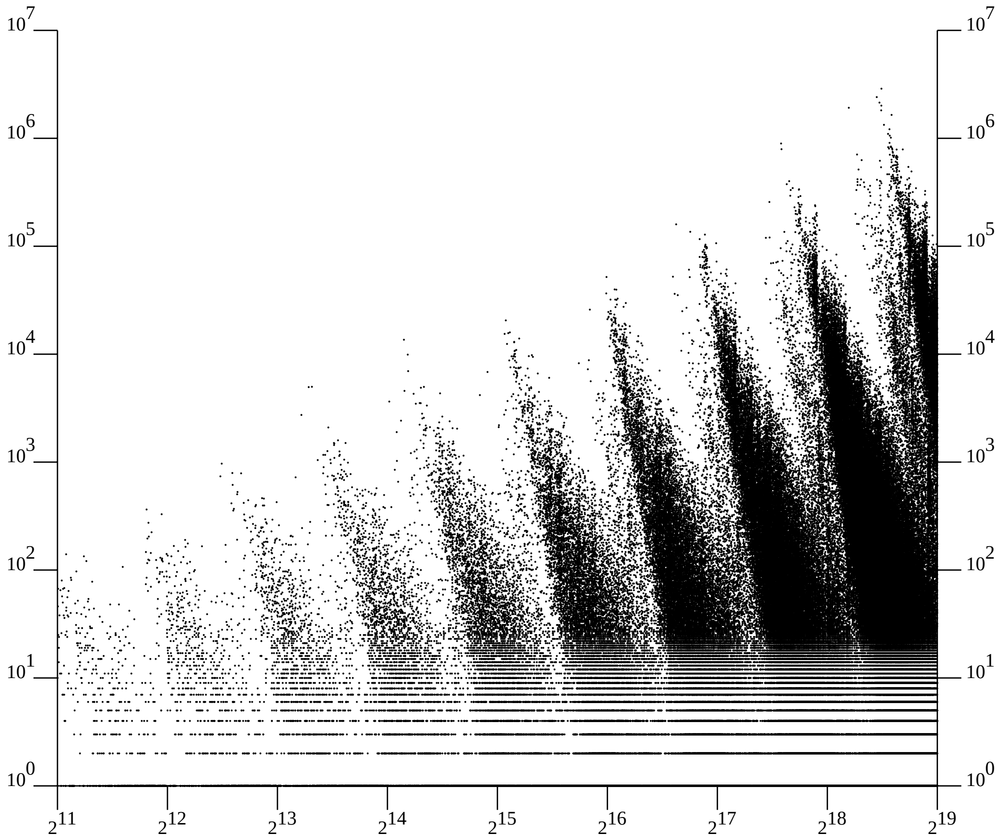

FIG. 8.1. *Examined number of nodes of the search tree to calculate $\ell(n)$ with our branch and bound algorithm.*

REFERENCES

[1] D. Bleichenbacher, *Efficiency and Security of Cryptosystems based on Number Theory*, PhD thesis, Swiss Federal Institute of Technology Zürich (ETHZ), 1996.

[2] J. Bos and M. Coster, *Addition chain heuristics*, in Advances in Cryptology – CRYPTO '89, G. Brassard, ed., vol. 435 of Lecture Notes in Computer Science, Springer Verlag, 1990, pp. 400–407.

[3] P. Downey, B. Leong, and R. Sethi, *Computing sequences with addition chains*, SIAM Journal on Computing, 10 (1981), pp. 638–646.

[4] A. Flammenkamp, *Drei Beiträge zur diskreten Mathematik: Additionsketten, no-three-in-line-problem, sociable numbers.* Diploma thesis, Universität Bielefeld, 1991.

[5] K. R. Hebb, *Some results on addition chains*, Notices American Mathematical Society, 21 (1974), pp. A–294.

[6] D. E. Knuth, *The art of computer programming, Seminumerical Algorithms*, vol. 2, Addison Wesley, 2nd ed., 1981.

[7] J. Sauerbrey and A. Dietel, *Resource requirements for the application of addition chains in modulo exponentiation*, in Advances in Cryptology — EUROCRYPT '92, R. A. Rueppel, ed., vol. 658 of Lecture Notes in Computer Science, Springer Verlag, 1993, pp. 174–182.

[8] A. Scholz, *Aufgabe 253*, in Jahresbericht der deutschen Mathematiker Vereinigung, E. Sperner, ed., vol. 47 2nd division, B. G. Teubner, 1937, pp. 41–42.

[9] A. Schönhage, *A lower bound for the length of addition chains*, Theoretical Computer Science, 1 (1975), pp. 229–242.

[10] R. Sonntag, *Theorie der Additonsketten*, PhD thesis, Technische Universität Hannover, 1975.

[11] K. B. Stolarsky, *A lower bound for the Scholz-Brauer problem*, Canadion Journal of Mathematics, 21 (1969), pp. 675–683.
[12] M. V. Subbarao, *Addition chains — some results and problems*, in Number theory and applications, R. A. Mollin, ed., vol. 265 of NATO Advanced Science Institutes Series C: Mathematical and Pysical Sciences, Kluwer Academic Publishers Group, 1989, pp. 555–574.
[13] E. G. Thurber, *The Scholz-Brauer problem on addition-chains*, Pacific Journal of Mathematics, 49 (1973), pp. 229–242.
[14] ———, *Addition chains and solutions of* $l(2n) = l(n)$ *and* $l(2^n - 1) = n + l(n) - 1$, Discrete Mathematics, 16 (1976), pp. 279–289.
[15] ———, *Addition chains – an erratic sequence*, Discrete Mathematics, 122 (1993), pp. 287–305.
[16] Y. Tsai and Y. Chin, *A study of some addition chain problems*, International Journal of Computer Mathematics, 22 (1987), pp. 117–134.
[17] Y. Yacobi, *Exponentiating faster with addition chains*, in Advances in Cryptology: – EUROCRYPT '90, vol. 473 of Lecture Notes in Computer Science, Springer Verlag, 1990, pp. 222–229.