LINEAR-TIME APPROXIMATION ALGORITHMS FOR COMPUTING NUMERICAL SUMMATION WITH PROVABLY SMALL ERRORS[*]

MING-YANG KAO[†] AND JIE WANG[‡]

Abstract. Given a multiset $X = \{x_1, \ldots, x_n\}$ of real numbers, the *floating-point set summation* problem asks for $S_n = x_1 + \cdots + x_n$. Let E_n^* denote the minimum worst-case error over all possible orderings of evaluating S_n. We prove that if X has both positive and negative numbers, it is NP-hard to compute S_n with the worst-case error equal to E_n^*. We then give the first known polynomial-time approximation algorithm that has a provably small error for arbitrary X. Our algorithm incurs a worst-case error at most $2(\lceil \log(n-1) \rceil + 1)E_n^*$.[1] After X is sorted, it runs in $O(n)$ time. For the case where X is either all positive or all negative, we give another approximation algorithm with a worst-case error at most $\lceil \log \log n \rceil E_n^*$. Even for unsorted X, this algorithm runs in $O(n)$ time. Previously, the best linear-time approximation algorithm had a worst-case error at most $\lceil \log n \rceil E_n^*$, while E_n^* was known to be attainable in $O(n \log n)$ time using Huffman coding.

Key words. floating-point summation, error analysis, addition trees, combinatorial optimization, NP-hardness, approximation algorithms

AMS subject classifications. 65G05, 65B10, 68Q15, 68Q25, 68R05

1. Introduction.

Summation of floating-point numbers is ubiquitous in numerical analysis and has been extensively studied (for example, see [2, 4, 5, 7, 8, 11, 10, 6, 12]). This paper focuses on the *floating-point set summation* problem which, given a multiset $X = \{x_1, \ldots, x_n\}$ of real numbers, asks for $S_n = x_1 + x_2 + \cdots + x_n$. Without loss of generality, let $x_i \neq 0$ for all i throughout the paper. Here X may contain both positive and negative numbers. For such a general X, previous studies have discussed heuristic methods and obtained statistical or empirical bounds for their errors. We take a new approach by designing efficient algorithms whose worst-case errors are provably small.

Our error analysis uses the standard model of floating-point arithmetic with unit roundoff $\alpha \ll 1$:

$$\mathrm{fl}(x+y) = (x+y)(1+\delta_{xy}), \text{where } |\delta_{xy}| \leq \alpha.$$

Since operator $+$ is applied to two operands at a time, an ordering for adding X corresponds to a binary addition tree of n leaves and $n-1$ internal nodes, where a leaf is an x_i and an internal node is the sum of its two children. Different orderings yield different addition trees, which may produce different computed sums $\hat{S}_n$ in floating-point arithmetic. We aim to find an optimal ordering that minimizes the error $E_n = |\hat{S}_n - S_n|$. Let $I_1, \ldots, I_{n-1}$ be the internal nodes of an addition tree T over X. Since α is very small even on a desktop computer, any product of more than

[*]A preliminary version of this work appeared as part of *Efficient minimization of numerical summation errors*, in Lecture Notes in Computer Science 1443: Proceedings of the 25th International Colloquium on Automata, Languages, and Programming, K. G. Larsen, S. Skyum, and G. Winskel, eds., Springer-Verlag, New York, NY, 1998, pp. 375–386.

[†]Department of Computer Science, Yale University, New Haven, CT 06520. Email: kao-ming-yang@cs.yale.edu. Supported in part by NSF Grant CCR-9531028.

[‡]Department of Mathematical Sciences, University of North Carolina at Greensboro, Greensboro, NC 27412. Email: wang@uncg.edu. Supported in part by NSF Grant CCR-9424164.

[1]All logarithms log in this paper are base 2.

one α is negligible in our consideration. Using this approximation,

$$\hat{S}_n \approx S_n + \sum_{i=1}^{n-1} I_i \delta_i.$$

Hence, $E_n \approx |\sum_{i=1}^{n-1} I_i\delta_i| \le \alpha \sum_{i=1}^{n-1} |I_i|$, giving rise to the following definitions:

- The *worst-case error* of T, denoted by $E(T)$, is $\alpha \sum_{i=1}^{n-1} |I_i|$.
- The *cost* of T, denoted by $C(T)$, is $\sum_{i=1}^{n-1} |I_i|$.

Our task is to find a fast algorithm that constructs an addition tree T over X such that $E(T)$ is small. Since $E(T) = \alpha{\cdot}C(T)$, minimizing $E(T)$ is equivalent to minimizing $C(T)$. We further adopt the following notations:

- E_n^* (respectively, C_n^*) is the minimum worst-case error (respectively, minimum cost) over all orderings of evaluating S_n.
- $T_{\min}$ denotes an optimal addition tree over X, i.e., $E(T_{\min}) = E_n^*$ or equivalently $C(T_{\min}) = C_n^*$.

In §2, we prove that if X contains both positive and negative numbers, it is NP-hard to compute a $T_{\min}$. In light of this result, we design an approximation algorithm in §3.1 that computes a tree T with $E(T) \le 2(\lceil \log(n-1) \rceil + 1)E_n^*$. After X is sorted, this algorithm takes only $O(n)$ time. This is the first known polynomial-time approximation algorithm that has a provably small error for arbitrary X. For the case where X is either all positive or all negative, we give another approximation algorithm in §3.2 that computes a tree T with $E(T) \le (1 + \lceil \log\log n \rceil)E_n^*$. This algorithm takes only $O(n)$ time even for unsorted X. Previously [5], the best linear-time approximation algorithm had a worst-case error at most $\lceil \log n \rceil E_n^*$, while E_n^* was known to be attainable in $O(n \log n)$ time using Huffman coding [9].

2. Minimizing the worst-case error is NP-hard.

If X contains both positive and negative numbers, we prove that it is NP-hard to find a $T_{\min}$. We first observe the following properties of $T_{\min}$.

LEMMA 2.1. *Let z be an internal node in $T_{\min}$ with children z_1 and z_2, sibling u, and parent r.*

1. *If $z > 0$, $z_1 \ge 0$, and $z_2 > 0$, then $u \ge 0$ or $r < 0$.*
2. *If $z < 0$, $z_1 \le 0$, and $z_2 < 0$, then $u \le 0$ or $r > 0$.*

Proof. By symmetry. we only prove the first statement. $C(T_r) = |r| + |z| + C_f$, where $C_f = C(T_{z_1}) + C(T_{z_2}) + C(T_u)$. Assume to the contrary that $u < 0$ and $r \ge 0$. Then $z \ge |u|$. We swap T_{z_1} with T_u. Let $z' = u + z_2$. Now r becomes the parent of z' and z_1. This rearrangement of nodes does not affect the value of node r, and the costs of T_{z_1}, T_{z_2}, and T_u remain unchanged. Let T_r' be the new subtree with root r. Let T' be the entire new tree resulted from the swapping. Since u and z_2 have the opposite signs, $|z'| < \max\{|u|, z_2\} \le z$. Hence, $C(T_r') = r + |z'| + C_f < r + z + C_f = C(T_r)$. Thus, $C(T') < C(T_{\min})$, contradicting the optimality of $T_{\min}$. This completes the proof. □

For the purpose of proving that finding a $T_{\min}$ is NP-hard, we restrict all x_i to nonzero integers and consider the following optimization problem.

MINIMUM ADDITION TREE (MAT)

Input: A multiset X of n nonzero integers $x_1, \ldots, x_n$.

Output: Some $T_{\min}$ over X.

The following problem is a decision version of MAT.

ADDITION TREE (AT)

Instance: A multiset X of n nonzero integers $x_1, \ldots, x_n$, and an integer $k \ge 0$.

Question: Does there exist an addition tree T over X with $C(T) \leq k$?

LEMMA 2.2. *If* MAT *is solvable in time polynomial in* n*, then* AT *is also solvable in time polynomial in* n.

Proof. Straightforward. ☐ In light of Lemma 2.2, to prove that MAT is NP-hard, it suffices to reduce the following NP-complete problem [3] to AT.

3-PARTITION (3PAR)

Instance: A multiset B of $3m$ positive integers $b_1, \ldots, b_{3m}$, and a positive integer K such that $K/4 < b_i < K/2$ and $b_1 + \cdots + b_{3m} = mK$.

Question: Can B be partitioned into m disjoint sets $B_1, \ldots, B_m$ such that for each B_i, $\sum_{b \in B_i} b = K$? (B_i must therefore contain exactly three elements from B.)

Given an instance (B, K) of 3PAR, let

$$W = 100(5m)^2 K;\ a_i = b_i + W;\ A = \{a_1, \ldots, a_{3m}\};\ L = 3W + K.$$

LEMMA 2.3. (A, L) *is an instance of* 3PAR*. Furthermore, it is a positive instance if and only if* (B, K) *also is.*

Proof. Since $K/4 < b_i < K/2$, $K/4 + W < a_i < K/2 + W$ and thus $L/4 < a_i < L/2$. Next, $a_1 + a_2 + \cdots + a_{3m} = 3mW + mK = mL$. This complete the proof of the first statement. The second statement follows from the fact that $b_i + b_j + b_k = K$ if and only if $a_i + a_j + a_k = L$. ☐

Write

$$\epsilon = \frac{1}{400(5m)^2};\ h = \lfloor 4\epsilon L \rfloor;\ H = L + h;$$

$$h = \beta_0 H;\ a_i = \left(\frac{1}{3} + \beta_i\right)H;\ a_i = \left(\frac{1}{3} + \epsilon_i\right)L;\ a_M = \max\{a_i : i = 1, \ldots, 3m\}.$$

LEMMA 2.4.

1. $|\epsilon_i| < \epsilon$ *for* $i = 1, \ldots, 3m$.
2. $0 < \beta_0 < 4\epsilon$*, and* $|\beta_i| < 4\epsilon$ *for* $i = 1, \ldots, 3m$.
3. $3a_M < H$.

Proof.

Statement 1. Note that $b_i + W = (1/3 + \epsilon_i)(3W + K)$. Thus, $b_i = K/3 + \epsilon_i(300(5m)^2 + 1)K$. Since $K/4 < b_i < K/2$, $-1/12 < \epsilon_i(300(5m)^2 + 1) < 1/6$. Hence, $4(5m)^2|\epsilon_i| < 10^{-2}$, i.e., $|\epsilon_i| < \epsilon$.

Statement 2. Since $4\epsilon L > 1$, we have $\beta_0 > 0$. Also, since $H > L$ and $\beta_0 H = \lfloor 4\epsilon L \rfloor$, we have $\beta_0 < 4\epsilon$. Next, for each a_i, we have $\beta_i = (\epsilon_i L - h/3)/(L + h)$. Then by the triangular inequality and Statement 1, $|\beta_i| < 7\epsilon/3 < 4\epsilon$.

Statement 3. By Statement 1, $a_i < (1/3 + \epsilon)L$. Thus $3a_M < L + 3\epsilon L$. Then, since $3\epsilon L < 3K \leq h$, $3a_M < L + h = H$. ☐

To reduce (A, L) to an instance of AT, we consider a particular multiset

$$X = A \cup \{-H, \ldots, -H\} \cup \{h, \ldots, h\}$$

with m copies of $-H$ and h each. Given a node s in $T_{\min}$, let T_s denote the subtree rooted at s. For convenience, also let s denote the value of node s. Let $v(T_{\min})$ denote the value of the root of $T_{\min}$, which is always 0. For brevity, we use λ with or without scripts to denote the sum of at most $5m$ numbers in the form of $\pm\beta_i$. Then all nodes are in the form of $(N/3 + \lambda)H$ for some integer N and some λ. Since by Lemma 2.4, $|\lambda| \leq (5m)(4\epsilon) = (500m)^{-1}$, the terms N and λ of each node are uniquely determined.

The nodes in the form of λH are called the *type*-0 nodes. Note that $T_{\min}$ has m type-0 leaves, i.e., the m copies of h in X.

LEMMA 2.5. *In $T_{\min}$, type-0 nodes can only be added to type-0 nodes.*

Proof. Assume to the contrary that a type-0 node z_1 is added to a node z_2 in the form of $(\pm N/3+\lambda)H$ with $N \geq 1$. Then $|z_1+z_2| \geq (1/3+\lambda')H$ for some λ'. Let z be the parent of z_1 and z_2. Since $v(T_{\min})=0$, z cannot be the root of $T_{\min}$. Let u be the sibling of z. Let r be the parent of z and u. Let t be the root of $T_{\min}$. Let P_r be the path from t to r in $T_{\min}$. Let m_r be the number of nodes on P_r. Since $T_{\min}$ has $5m-1$ internal nodes, $m_r < 5m-1$.

We rearrange $T_{\min}$ to obtain a new tree T' as follows. First, we replace T_z with T_{z_2}; i.e., r now has subtrees T_{z_2} and T_u. Let T'' be the remaining tree; i.e., T'' is $T_{\min}$ after removing T_{z_1}. Next, we create T' such that its root has subtrees T_{z_1} and T''. This tree rearrangement eliminates the cost $|z_1+z_2|$ from T_r but may result in a new cost in the form of λH on each node of P_r. The total of these extra costs, denoted by C_λ, is at most $m_r(5m)(4\epsilon)H < (5m-1)(5m)(4\epsilon)H$. Then, $C(T') = C(T_{\min}) - |z_1+z_2| + C_\lambda \leq C(T_{\min}) - (1/3+\lambda')H + C_\lambda < C(T_{\min}) + (-1/3 + (5m)^2(4\epsilon))H = C(T_{\min}) + (-1/3+10^{-2})H < C(T_{\min})$, contradicting the optimality of $T_{\min}$. This completes the proof. $\square$

LEMMA 2.6. *Let z be a node in $T_{\min}$.*

1. *If $z < 0$, then $|z| \leq H$.*
2. *If $z > 0$, then $z < H$.*

Proof.

Statement 1. Assume that the statement is untrue. Then, since all negative leaves have values $-H$, some negative internal node z has an absolute value greater than H and two negative children z_1 and z_2. Since $v(T_{\min})=0$, some z has a positive sibling u. We pick such a z at the lowest possible level of $T_{\min}$. Let r be the parent of z and u. By Lemma 2.1(2), $r > 0$. Then $u > |z| > H$. Since all positive leaves have values less than H, u is an internal node with two children u_1 and u_2. Since $u > 0$, $z < 0$, and $r > 0$, by Lemma 2.1(1), u must have a positive child and a negative child. Without loss of generality, let u_1 be positive and u_2 be negative. Then $u = u_1 - |u_2|$. Since z is at the lowest possible level, $|u_2| \leq H$, for otherwise we could find a z at a lower level under u_2. We swap T_z with T_{u_2}. Let T'_r be the new subtree rooted r. Let $u' = u_1 + z$. Since $u_2 + u' = r > 0$ and $u_2 < 0$, we have $u' > 0$. Since $|u_2| \leq H < |z|$, we have $u' = u_1 - |z| < u_1 - |u_2| = u$. Let $C_f = C(T_z) + C(T_{u_1}) + C(T_{u_2})$. Then, $C(T'_r) = r + u' + C_f < r + u + C_f = C(T_r)$, which contradicts the optimality of $T_{\min}$ because the costs of the internal nodes not mentioned above remain unchanged.

Statement 2. Assume that this statement is false. Then, sinc

some $a_j \in A$ or to some $z_1 = (-1/3 + \lambda_1)H$. In turn, z_1 can only be the sum of $-H$ and some $z_2 = (2/3 + \lambda_2)H$. In turn, z_2 is the sum of some a_k and $a_\ell \in A$. Hence, in $T_{\min}$, $2m$ leaves in A are added in pairs. The sum of each pair is then added to a leaf node $-H$. This sum is then added to a leaf node in A. This sum is a type-0 node with value $-|\lambda'|H$, which can only be added to another type-0 node. Let $a_{p,1}, a_{p,2}, a_{p,3}$ be the three leaves in A associated with each $-H$ and added together as $((a_{p,1}+a_{p,2})+(-H))+a_{p,3}$ in $T_{\min}$. The cost of such a subtree is $2H-(a_{p,1}+a_{p,2}+a_{p,3})$. There are m such subtrees R_p. Their total cost is $2mH-\sum_{i=1}^{3m} a_i = mH+mh$. Hence, $C(T_{\min}) \geq mH + mh$.

If (A, L) is not a positive instance of 3PAR, then for any $T_{\min}$, there is some subtree R_p with $a_{p,1} + a_{p,2} + a_{p,3} \neq L$. Then, the value of the root r_i of R_p is $a_{p,1} + a_{p,2} + a_{p,3} - H \neq -h$. Since r_i is a type-0 node, it can only be added to a type-0 node. No matter how the m root values r_k and the m leaves h are added, some node resulting from adding these $2m$ numbers is nonzero. Hence, $C(T_{\min}) > mH + mh$.

If (A, L) is a positive instance of 3PAR, let $\{a_{p,1}, a_{p,2}, a_{p,3}\}$ with $1 \leq p \leq m$ form a 3-set partition of A; i.e., A is the union of these m 3-sets and for each p, $a_{p,1} + a_{p,2} + a_{p,3} = L$. Then each 3-set can be added to one $-H$ and one h as $(((a_{p,1} + a_{p,2}) + (-H)) + a_{p,3}) + h$, resulting in a node of value zero and contributing no extra cost. Hence, $C(T_{\min}) = mH + mh$. This completes the proof. ▯

THEOREM 2.11. *It is NP-hard to compute an optimal addition tree over a multiset that contains both positive and negative numbers.*

Proof. By Lemma 2.2, it suffices to construct a reduction f from 3PAR to AT. Let $f(B, K) = (X, mH + mh)$, which is polynomial-time computable. By Lemma 2.10, $(X, mH + mh)$ is a positive instance of AT if and only if (A, L) is a positive instance of 3PAR. Then, by Lemma 2.3, f is a desired reduction. ▯

3. Approximation algorithms.

In light of Theorem 2.11, for X with both positive and negative numbers, no polynomial-time algorithm can find a $T_{\min}$ unless P = NP [3]. This motivates the consideration of approximation algorithms.

3.1. Linear-time approximation for general X.

This section assumes that X contains at least one positive number and one negative number. We give an approximation algorithm whose worst-case error is at most $2(\lceil \log(n-1) \rceil + 1)E_n^*$. If X is sorted, this algorithm takes only $O(n)$ time.

In an addition tree, a leaf is *critical* if its sibling is a leaf with the opposite sign. Note that if two leaves are siblings, then one is critical if and only if the other is critical. Hence, an addition tree has an even number of critical leaves.

LEMMA 3.1. *Let T be an addition tree over X. Let $y_1, \ldots, y_{2k}$ be its critical leaves, where y_{2i-1} and y_{2i} are siblings. Let $z_1, \ldots, z_{n-2k}$ be the noncritical leaves. Let $\Pi = \sum_{i=1}^{k} |y_{2i-1} + y_{2i}|$, and $\Delta = \sum_{j=1}^{n-2k} |z_j|$. Then $C(T) \geq (\Pi + \Delta)/2$.*

Proof. Let x be a leaf in T. There are two cases.

Case 1: x is some critical leaf y_{2i-1} or y_{2i}. Let r_i be the parent of y_{2i-1} and y_{2i} in T for $1 \leq i \leq k$. Then $|r_i| = |y_{2i-1} + y_{2i}|$.

Case 2: x is some noncritical leaf z_j. Let w_j be the sibling of z_j in T. Let q_j be the parent of z_j and w_j. There are three subcases.

Case 2A: w_j is also a leaf. Since z_j is noncritical, w_j has the same sign as z_j and is also a noncritical leaf. Thus, $|q_j| = |z_j| + |w_j|$.

Case 2B: w_j is an internal node with the same sign as z_j. Then $|q_j| \geq |z_j|$.

Case 2C: w_j is an internal node with the opposite sign to z_j. If $|w_j| \geq |z_j|$, then $|q_j| + |w_j| \geq |z_j|$; if $|w_j| < |z_j|$, then $|q_j| + |w_j| = |z_j|$. So, we always have

$|q_j| + |w_j| \geq |z_j|$.

Observe that

$$C(T) \geq \sum_{i=1}^{k} |r_i| + \frac{1}{2}\left(\sum_{z_j \text{ in Case 2A}} |q_j|\right) + \sum_{z_j \text{ in Case 2B}} |q_j| + \sum_{z_j \text{ in Case 2C}} |q_j|;$$

$$C(T) \geq \sum_{z_j \text{ in Case 2C}} |w_j|.$$

Simplifying the sum of these two inequalities based on the case analysis, we have $2C(T) \geq \Pi + \Delta$ as desired. □

In view of Lemma 3.1, we desire to minimize $\Pi + \Delta$ over all possible T. Given $x_t, x_{t'} \in X$ with $t \neq t'$, $(x_t, x_{t'})$ is a *critical pair* if x_t and $x_{t'}$ have the opposite signs. A *critical matching* R of X is a set $\{(x_{t_{2i-1}}, x_{t_{2i}}) : i = 1, \dots, k\}$ of critical pairs where the indices t_j are all distinct. For simplicity, let $y_j = x_{t_j}$. Let $\Pi = \sum_{i=1}^{k} |y_{2i-1} + y_{2i}|$ and $\Delta = \sum_{z \in X - \{y_1, \dots, y_{2k}\}} |z|$. If $\Pi + \Delta$ is the minimum over all critical matchings of X, then R is called a *minimum critical matching* of X. Such an R can be computed as follows. Assume that X consists of ℓ positive numbers $a_1 \leq \cdots \leq a_\ell$ and m negative numbers $-b_1 \geq \cdots \geq -b_m$.

Algorithm 1.

1. If $\ell = m$, let $R = \{(a_i, -b_i) : i = 1, \dots, \ell\}$.
2. If $\ell < m$, let $R = \{(a_i, -b_{i+m-\ell}) : i = 1, \dots, \ell\}$.
3. If $\ell > m$, let $R = \{(a_{i+\ell-m}, -b_i) : i = 1, \dots, m\}$.

Lemma 3.2. *If X is sorted, then Algorithm 1 computes a minimum critical matching R of X in $O(n)$ time.*

Proof. By case analysis, if $a_i \leq a_j$ and $b_{i'} \leq b_{j'}$, then $|a_i - b_{i'}| + |a_j - b_{j'}| \leq |a_i - b_{j'}| + |a_j - b_{i'}|$. Thus, if $\ell = m$, then pairing a_i with $-b_i$ returns the minimum $\Pi + \Delta$. For the case $\ell < m$, let ϵ be an infinitesimally small positive number. Let X' be X with additional $m - \ell$ copies of ϵ. Then, $\sum_{i=1}^{\ell} |a_i - b_{i+m-\ell}| + \sum_{i=1}^{m-\ell} |\epsilon - b_i| = (\ell - m)\epsilon + \Pi + \Delta$ is the minimum over all possible critical matchings of X'. Thus, $\Pi + \Delta$ is the minimum over all possible critical matching of X. The case $\ell > m$ is symmetric to the case $\ell < m$. Since X is sorted, the running time of Algorithm 1 is $O(n)$. □

We now present an approximation algorithm to compute the summation over X.

Algorithm 2.

1. Use Algorithm 1 to find a minimum critical matching R of X. The numbers x_i in the pairs of R are the critical leaves in our addition tree over X and those not in the critical pairs are the noncritical leaves.
2. Add each critical pair of R separately.
3. Construct a balanced addition tree over the resulting sums of Step 2 and the noncritical leaves.

Theorem 3.3. *Let T be the addition tree over X constructed by Algorithm 2. If X is sorted, then T can be obtained in $O(n)$ time and $E(T) \leq 2(\lceil \log(n-1) \rceil + 1)E(T_{\min})$.*

Proof. Steps 2 and 3 of Algorithm 2 both take $O(n)$ time. By Lemma 3.2, Step 1 also takes $O(n)$ time and thus Algorithm 2 takes $O(n)$ time. As for the error analysis, let T' be the addition tree constructed at Step 3. Then $C(T) = C(T') + \Pi$. Let h be the number of levels of T'. Since T' is a balanced tree, $C(T') \leq (h-1)(\Pi + \Delta)$ and thus $C(T) \leq h(\Pi + \Delta)$. By assumption, X has at least two numbers with the opposite signs. So there are at most $n-1$ numbers to be added pairwise at Step 3. Thus, $h \le \lceil \log(n-1) \rceil + 1$. Next, by Lemma 3.1, since R is a minimum critical matching of X, we have $C(T_{\min}) \ge (\Pi + \Delta)/2$. In summary, $E(T) \le 2(\lceil \log(n-1) \rceil + 1)E(T_{\min})$. ∎

3.2. Improved approximation for single-sign X. This section assumes that all x_i are positive; the symmetric case where all x_i are negative can be handled similarly.

Let T be an addition tree over X. Observe that $C(T) = \sum_{i=1}^{n} x_i d_i$, where d_i is the number of edges on the path from the root to the leaf x_i in T. Hence, finding an optimal addition tree over X is equivalent to constructing a Huffman tree to encode n characters with frequencies $x_1, \ldots, x_n$ into binary strings [9].

FACT 3.1. *If X is unsorted (respectively, sorted), then a $T_{\min}$ over X can be constructed in $O(n \log n)$ (respectively, $O(n)$) time.*

Proof. If X is unsorted (respectively, sorted), then a Huffman tree over X can be constructed in $O(n \log n)$ [1] (respectively, $O(n)$ [9]) time. ∎

For the case where X is unsorted, many applications require faster running time than $O(n \log n)$. Previously, the best $O(n)$-time approximation algorithm used a balanced addition tree and thus had a worst-case error at most $\lceil \log n \rceil E_n^*$. Here we provide an $O(n)$-time approximation algorithm to compute the sum over X with a worst-case error at most $\lceil \log \log n \rceil E_n^*$. More generally, given an integer parameter $t > 0$, we wish to find an addition tree T over X such that $C(T) \le C(T_{\min}) + t \cdot |S_n|$.

ALGORITHM 3.

1. Let $m = \lceil n/2^t \rceil$. Partition X into m disjoint sets $Z_1, \ldots, Z_m$ such that each Z_i has exactly 2^t numbers, except possibly Z_m, which may have less than 2^t numbers.
2. For each Z_i, let $z_i = \max\{x : x \in Z_i\}$. Let $M = \{z_i : 1 \le i \le m\}$.
3. For each Z_i, construct a balanced addition tree T_i over Z_i.
4. Construct a Huffman tree H over M.
5. Construct the final addition tree T over X from H by replacing z_i with T_i.

THEOREM 3.4. *Assume that $x_1, \ldots, x_n$ are all positive. For any integer $t > 0$, Algorithm 3 computes an addition tree T over X in $O(n + m \log m)$ time with $C(T) \le C(T_{\min}) + t|S_n|$, where $m = \lceil n/2^t \rceil$. Since $|S_n| \le C(T_{\min})$, $E(T) \le (1+t)E(T_{\min})$.*

Proof. For an addition tree L and a node y in L, the *depth* of y in L, denoted by $d_L(y)$, is the number of edges on the path from the root of L to y. Since H is a Huffman tree over $M \subseteq X$ and every $T_{\min}$ is a Huffman tree over X, there exists some $T_{\min}$ such that for each z_j, its depth in $T_{\min}$ is at least its depth in H. Furthermore, in $T_{\min}$, the depth of each $y \in Z_i$ is at least that of z_i. Therefore,

$$\sum_{i=1}^{m} \sum_{x_j \in Z_i} x_j \cdot d_H(z_i) \le C(T_{\min}).$$

Also note that for $x_j \in Z_i$, $d_T(x_j) - d_H(z_i) \le \log 2^t = t$. Hence,

$$\begin{aligned} C(T) &= \sum_{x_i \in X} x_i \cdot d_T(x_i) \\ &= \sum_{i=1}^{m} \sum_{x_j \in Z_i} x_j \cdot d_H(z_i) + \sum_{i=1}^{m} \sum_{x_j \in Z_i} x_j \cdot (d_T(x_j) - d_H(z_i)) \\ &\le C(T_{\min}) + t \sum_{x_i \in X} x_i \end{aligned}$$

In summary, $C(T) \leq C(T_{\min}) + tS_n$. Since Step 4 takes $O(m \log m)$ time and the others take $O(n)$ time, the total running time of Algorithm 3 is as stated. □

COROLLARY 3.5. *Assume that $n \geq 4$ and all $x_1, \ldots, x_n$ are positive. Then, setting $t = \lfloor \log((\log\ n) - 1) \rfloor$, Algorithm 3 finds an addition tree T over X in $O(n)$ time with $E(T) \leq \lceil \log \log n \rceil E(T_{\min})$.*

Proof. Follows from Theorem 3.4. □

Acknowledgments. The authors thank Don Rose, Hai Shao, Xiaobai Sun, and Steve Tate for helpful discussions and thank the anonymous referees for their detailed comments.

REFERENCES

[1] T. H. Cormen, C. L. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[2] J. W. Demmel, *Underflow and the reliability of numerical software*, SIAM Journal on Scientific and Statistical Computing, 5 (1984), pp. 887–919.

[3] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, NY, 1979.

[4] D. Goldberg, *What every computer scientist should know about floating-point arithmetic*, ACM Computing Surveys, 23 (1990), pp. 5–48.

[5] N. J. Higham, *The accuracy of floating point summation*, SIAM Journal on Scientific Computing, 14 (1993), pp. 783–799.

[6] N. J. Higham, *Accuracy and stability of numerical algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996.

[7] D. E. Knuth, *The Art of Computer Programming II: Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 3rd ed., 1997.

[8] U. W. Kulisch and W. L. Miranker, *The arithmetic of the digital computer: a new approach*, SIAM Review, 28 (1986), pp. 1–40.

[9] J. V. Leeuwen, *On the construction of Huffman trees*, in Proceedings of the 3rd International Colloquium on Automata, Languages, and Programming, 1976, pp. 382–410.

[10] A. I. Mikov, *Large-scale addition of machine real numbers: Accuracy estimates*, Theoretical Computer Science, 162 (1996), pp. 151–170.

[11] T. G. Robertazzi and S. C. Schwartz, *Best "ordering" for floating-point addition*, ACM Transactions on Mathematical Software, 14 (1988), pp. 101–110.

[12] V. J. Torczon, *On the convergence of the multidirectional search algorithm*, SIAM Journal on Optimization, 1 (1991), pp. 123–145.