SIAM J. COMPUT.
Vol. 48, No. 5, pp. 1481–1486
© 2019 Oded Schwartz and Elad Weiss

REVISITING "COMPUTATION OF MATRIX CHAIN PRODUCTS"*

ODED SCHWARTZ† AND ELAD WEISS†

Abstract. The matrix chain ordering problem aims to reduce the number of arithmetic operations required for evaluating the product of N matrices. Using a dynamic programming algorithm this problem can be solved in $O(N^3)$ time. Hu and Shing obtained a sophisticated algorithm that solves the problem in $O(N \log N)$ [*SIAM J. Comput.*, 11 (1982), pp. 362–373]. Unfortunately, as we show here, the correctness proof of their algorithm is wrong. This flaw affects another algorithm for the same problem, by Wang, Zhu, and Tian (2013), and algorithms for many other problems that use chain matrix multiplication as a building block. We present an alternative proof for the correctness of the first two algorithms and show that a third algorithm by Nimbark, Gohel, and Doshi (2011) is beyond repair.

Key words. matrix multiplication, algorithms, matrix chain product

AMS subject classification. 68W40

DOI. 10.1137/18M1195401

1. Introduction. Given a chain multiplication of N matrices with dimensions $p_1, \ldots, p_{N+1}$, the number of scalar multiplications required is determined by the order of multiplication, namely, by the parentheses assignment. The objective of the matrix chain ordering problem *(MCOP)* is to find an optimal such parentheses assignment. The dynamic programming algorithm of Godbole [4] solves this problem in $O(N^3)$. Hu and Shing obtained an $O(N \log N)$ algorithm for the problem [5, 6, 7] by reducing it to finding the optimal triangulation of a convex polygon (see Definition 2.8 in section 2). Unfortunately, the proof of one of the fundamental lemmas in [7] is incorrect: Lemma 1 of [7] whose proof was omitted from the journal version and provided only on page 3 of Part ii in [5]. Wang, Zhu, and Tian obtained a simpler $O(N \log N)$ algorithm [16]. However, they rely on the correctness of Hu and Shing, particularly on the erroneous proof. Other works also build on Hu and Shing's algorithm ([18, 17, 3, 2, 9, 8, 11, 13, 12, 15] is a partial list). We explain the flaw in the proof and present an alternative proof, suggested by Shing [14].

We note that Bradford, Rawlins, and Shannon obtained a parallel polylogarithmic MCOP algorithm with total work $O\left(N \log^{1.5} N\right)$ [1]. This algorithm is independent of Hu and Shing's, hence it is not affected by our findings.

Also, a greedy algorithm for solving MCOP was proposed in [10], which does not depend in any way on Hu and Shing's. We prove that this algorithm is incorrect by providing a counterexample.

*Received by the editors June 20, 2018; accepted for publication (in revised form) July 1, 2019; published electronically September 5, 2019.
https://doi.org/10.1137/18M1195401

Funding: This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement 818252). Research is supported by grants 1878/14, and 1901/14 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities) and grant 3-10891 from the Ministry of Science and Technology, Israel. Research is also supported by the Einstein Foundation and the Minerva Foundation. This work was supported by the PetaCloud industry-academia consortium. This research was supported by a grant from the United States-Israel Bi-national Science Foundation (BSF), Jerusalem, Israel. This work was supported by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

†School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel (odedsc@cs.huji.ac.il, eladweiss@cs.huji.ac.il).

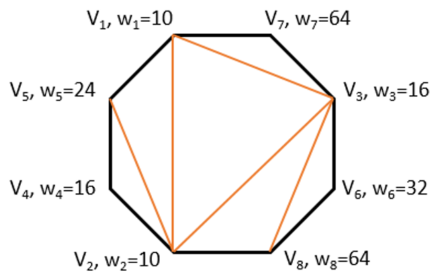

FIG. 1. *Vertex indexes of an* 8*-gon. Vertex weights are* 10, 16, 24, 10, 64, 16, 32, 64*, going clockwise.*

Our contribution. In section 2 we provide preliminaries. In section 3 we bring the lemma and the proof of [5] and explain the flaw in their proof. We then provide a corrected proof by Shing [14]. In section 4 we provide a counterexample to the greedy algorithm of [10].

2. Preliminaries. We next provide the required definitions from [5].

DEFINITION 2.1. *n-gon: An n-gon is a convex polygon with n vertexes. In a weighted n-gon, each vertex v is assigned a positive integer weight, $w(v)$. We adapt the notation of* [5] *and denote the edge connecting v and u by $v - u$.*

DEFINITION 2.2. *Vertex ordering: Given an n-gon, we define an order on the n vertexes as follows. The smallest vertex is chosen to be the vertex with the smallest weight. If there are more than one such vertex, then it is arbitrarily chosen from that set. Then, we define $u < v$ if $w(u) < w(v)$, or if $w(u) = w(v)$ and u is closer to the smallest vertex traversing clockwise.*

DEFINITION 2.3. *Vertex indexing (see Figure* 1*): We assign each vertex an index from the set $[n]$ to reflect the vertexes order. The smallest vertex is v_1, and the largest is v_n. From now on, we also use these indexes to denote the weights: w_i is the weight of vertex v_i.*

DEFINITION 2.4. *Edge ordering: We say that an edge $v_i - v_j$ is smaller than an edge $v_p - v_q$ if $\min\{i, j\} < \min\{p, q\}$, or if $\min\{i, j\} = \min\{p, q\}$ and $\max\{i, j\} < \max\{p, q\}$.*

DEFINITION 2.5. *Polygon triangulation (see Figure* 1*): A triangulation of a convex n-gon is a set of $n - 2$ nonintersecting arcs. The cost of a triangulation is the sum of the costs of all the triangles in the triangulation. The cost of a triangle is the product of the weights of its vertexes.*

DEFINITION 2.6. *The l-optimal partition: A partition of an n-gon is a set of edges forming a triangulation of the n-gon. An optimal partition is a partition of minimum cost. If more than one partition minimizes the cost, among all optimal partitions, we define the lexicographically optimal (l-optimal) partition to be the optimal partition of smallest lexicographic order of edges.*

DEFINITION 2.7. *A fan: A fan is a partition containing all arcs between v_1 and the other vertexes of the n-gon.*

DEFINITION 2.8. *The reduction: Given a matrix chain multiplication of length N, it can be reduced to a convex polygon with $N+1$ vertices, with the $N+1$ matrix dimensions as the vertices weights. A triangulation of this polygon corresponds to a parentheses assignment on the matrix chain. Each of the $N-2$ triangles corresponds to a matrix multiplication instance. The cost of a triangle is exactly the cost of multiplying two matrices with dimensions equal to the vertices weights.*

DEFINITION 2.9. *Horizontal and vertical arcs: Consider a 4-gon v_x, v_w, v_z, v_y (vertices ordered clockwise). An arc $v_x - v_z$ is a* vertical arc *(with respect to the 4-gon) if* $\min\{w_x, w_z\} < \min\{w_y, w_w\}$, *or* $\min\{w_x, w_z\} = \min\{w_y, w_w\}$ *and* $\max\{w_x, w_z\} \leq \max\{w_y, w_w\}$. *An arc $v_x - v_z$ is called a* horizontal arc *(w.r.t the 4-gon) if* $\min\{w_x, w_z\} > \min\{w_y, w_w\}$ *and* $\max\{w_x, w_z\} < \max\{w_y, w_w\}$. *For brevity, we use h-arcs and v-arcs to denote horizontal arcs and vertical arcs, respectively.*

DEFINITION 2.10. *Potential horizontal arcs: Consider an arc $v_x - v_z$ in an n-gon. Let v_w be the vertex with smallest weight among all vertices between v_x and v_z, traversing clockwise. Let v_y be the vertex with smallest weight among all vertices between v_z and v_x, traversing clockwise. The arc $v_x - v_z$ is a potential horizontal arc if $v_y < v_x < v_z < v_w$.*

LEMMA 2.11 (see Corollary 1 in [5, Part 1, page 12]). *For every way of choosing $v_1, v_2, \ldots,$ (as prescribed), the l-optimum partition always contains $v_1 - v_2$ and $v_1 - v_3$.*

LEMMA 2.12 (see Corollary 3 in [5, Part 1, page 17]). *All arcs in an optimum partition are either v-arcs or h-arcs.*

LEMMA 2.13 (see Corollary 4 in [5, Part 1, page 19]). *If $v_x - v_z$ is an h-arc of an optimal partition then $v_x - v_z$ is a potential horizontal arc.*

3. Lemma's proof correction. We first cite Lemma 1 in [5] (see Part ii, page 3) and its proof:

> LEMMA 3.1. *If none of the potential h-arcs appears in the l-optimum partition of the n-gon, the l-optimum partition must be the fan of the n-gon.*
>
> *Proof text as given in* [5]. From Theorem 3 of Part I, we know that any arc which exists as an h-arc in the l-optimum partition must be a potential h-arc. Hence, if the l-optimum partition does not contain any potential h-arc, the l-optimum partition must be made up of v-arcs only. Hence, we have to show that among all partitions which are made up of v-arcs only, the fan is (i) the lexicographically smallest and (ii) one of the cheapest partitions in the n-gon.
>
> (i) Since the fan consists of only v-arcs joining v_1 to all other vertices in the n-gon, it is by definition the lexicographically smallest partition.
> (ii) Suppose the l-optimum partition contains v-arcs only but is not the fan. There must exist three vertices v_i, v_j, v_k such that the triangles $(v_1, v_i, v_j), (v_i, v_j, v_k)$ are present in the l-optimum partition. Since $v_i - v_j$ is a v-arc (by assumption) and v_1 is the

smallest vertex in the n-gon, we have $w_1 = \min\{w_i, w_j\}$ and $\max\{w_i, w_j\} \leq w_k$. *If we replace the arc $v_i - v_j$ with the arc $v_1 - v_k$ we can get a partition whose cost is less than or equal to that of the l-optimum partition but is lexicographically smaller than the l-optimum partition, and results in a contradiction.* □

The last transition in the proof, marked in italic, is erroneous because it implicitly assumes that

$$w_1 \cdot w_i \cdot w_j + w_i \cdot w_j \cdot w_k > w_1 \cdot w_i \cdot w_k + w_1 \cdot w_j \cdot w_k.$$

However, since $w_i = w_1$ and $w_j \leq w_k$

$$w_1 \cdot w_i \cdot w_j + w_i \cdot w_j \cdot w_k \leq w_1 \cdot w_i \cdot w_k + w_1 \cdot w_j \cdot w_k,$$

making the last transition in their proof incorrect. This lemma is crucial for the correctness of their algorithm (both the version in [5] and in [7]).

Furthermore, the algorithm of Wang, Zhu, and Tian relies on this lemma as well (see Theorem 1, page 704 in [16]).

3.1. Corrected proof. We present a corrected proof by Shing [14].

Proof of Lemma 3.1. Suppose the l-optimum partition contains v-arcs only but is not the fan. There must exist three vertices v_i, v_j, v_k such that the triangles $(v_1, v_i, v_j), (v_i, v_j, v_k)$ are present in the l-optimum partition. Since the original n-gon is l-optimally partitioned, the subpolygon $v_1, v_i, \ldots, v_k, \ldots, v_j$ which is formed by the arcs $v_1 - v_i, v_1 - v_j$ and the sides of the n-gon from v_i to v_j in the clockwise direction must also be l-optimally partitioned. Since v_i and v_j are the only vertices adjacent to v_1 in the l-optimum partition of the subpolygon, it follows from Lemma 2.11 that one of $\{v_i, v_j\}$ must be the second smallest vertex and the other must be the third smallest vertex in the subpolygon. Hence, for each vertex v_m between v_i and v_j, we have either $v_1 \leq v_i \leq v_j \leq v_m$ if v_i is the second smallest vertex or $v_1 \leq v_j \leq v_i \leq v_m$ if v_j is the second smallest vertex, which implies that $v_i - v_j$ is a potential h-arc in the original n-gon, a contradiction. □

4. Another algorithm based on Hu and Shing's. Wang, Zhu, and Tian [16] build on the algorithm of Hu and Shing [5] and some of their theorems, to obtain a simplified version of the $O(NlogN)$ algorithm. However, Lemma 1 of [5] is at the heart of their construction (see Theorem 1, page 704 in [16]). Hence the corrected proofs are vital to the correctness of their algorithm as well.

5. A greedy $O(N)$ algorithm. Nimbark, Gohel, and Doshi [10] presented an $O(N)$ greedy algorithm for finding the optimal parentheses assignment (see Algorithm 1). Unfortunately, their algorithm is incorrect, as the following counterexample demonstrates.

Consider the matrix product $A_1A_2A_3$, where the dimensions are $2, 1000, 999, 10$. The greedy algorithm results with the parentheses assignment $(A_1(A_2A_3))$. This costs $999 \cdot 10 \cdot 999 + 2 \cdot 1000 \cdot 10 = 10,000,010$. However, the optimal assignment is $((A_1A_2)A_3)$ with cost $2 \cdot 1000 \cdot 999 + 2 \cdot 999 \cdot 10 = 2,017,980$.

Algorithm 1 Greedy Algorithm.

Let $P = A_1 \cdot A_2 \cdot \ldots \cdot A_n$ be a chain multiplication instance of n matrices.
Let a_{i-1}, a_i be the dimensions of matrix A_i.
Let $L_{colOrder}$ be a list of the matrices $A_{1,\ldots,n}$ sorted by their column dimension in descending order.
Let $L_{rowOrder}$ be a list of the matrices $A_{1,\ldots,n}$ sorted by their row dimension in descending order.
while $L_{colOrder}$ contains more than one matrix **do**
  Let A_k be the last element in $L_{colOrder}$.
  Let A_j be the last element in $L_{rowOrder}$ for which $A_j \cdot A_k$ is a valid matrix multiplication (e.g., A_j's columns dimension is the same as A_k's rows dimension).
  Put a new pair of parentheses: to the left of A_j, and to the right of A_k.
  Create new intermediate result matrix A_{kj}, having dimensions a_j, a_{k-1}, and insert it at the end of $L_{colOrder}$.
  Delete A_k and A_j from both $L_{colOrder}$ and $L_{rowOrder}$.
end while

Acknowledgments. We are most grateful to Mantak Shing for reading an earlier version of this manuscript and for offering the proof of Lemma 3.1.

REFERENCES

[1] P. G. Bradford, G. J. Rawlins, and G. E. Shannon, *Efficient matrix chain ordering in polylog time*, SIAM J. Comput., 27 (1998), pp. 466–490.
[2] A. Czumaj, *Parallel algorithm for the matrix chain product and the optimal triangulation problems*, in Annual Symposium on Theoretical Aspects of Computer Science, Springer, Berlin, 1993, pp. 294–305.
[3] Z. Galil and K. Park, *Dynamic programming with convexity, concavity and sparsity*, Theoret. Comput. Sci., 92 (1992), pp. 49–76.
[4] S. S. Godbole, *On efficient computation of matrix chain products*, IEEE Trans. Comput., 100 (1973), pp. 864–866.
[5] T. Hu and M. Shing, *Computation of Matrix Chain Products, Part* i*, Part* ii, Technical report, STAN-CS-81-875, Stanford University CA Department of Computer Science, Palo Alto, CA, 1981.
[6] T. C. Hu and M. T. Shing, *Computation of matrix chain products, Part* I, SIAM J. Comput., 11 (1982), pp. 362–373.
[7] T. C. Hu and M. T. Shing, *Computation of matrix chain products, Part* II, SIAM J. Comput., 13 (1984), pp. 228–251.
[8] H. Lee, J. Kim, S. J. Hong, and S. Lee, *Processor allocation and task scheduling of matrix chain products on parallel systems*, IEEE Trans. Parallel Distrib. Syst., 14 (2003), pp. 394–407.
[9] Y. A. Liu and S. D. Stoller, *Dynamic programming via static incrementalization*, Higher-Order Symb. Comput., 16 (2003), pp. 37–62.
[10] H. Nimbark, S. Gohel, and N. Doshi, *A novel approach for matrix chain multiplication using greedy technique for packet processing*, in Computer Networks and Information Technologies, Springer, Berlin, 2011, pp. 318–321.
[11] R. N. Pfeifer, J. Haegeman, and F. Verstraete, *Faster identification of optimal contraction sequences for tensor networks*, Phys. Rev. E (3), 90 (2014), 033315.
[12] P. Ramanan, *A new lower bound technique and its application: Tight lower bound for a polygon triangulation problem*, SIAM J. Comput., 23 (1994), pp. 834–851.
[13] P. Ramanan, *An efficient parallel algorithm for the matrix-chain-product problem*, SIAM J. Comput., 25 (1996), pp. 874–893.
[14] M. Shing. *private communication*, 2018.
[15] D. Tang and G. Gupta, *An efficient parallel dynamic programming algorithm*, Comput. Math. Appl., 30 (1995), pp. 65–74.

[16] X. Wang, D. Zhu, and J. Tian, *Efficient computation of matrix chain*, in Proceedings of the 2013 8th International Conference on Computer Science & Education (ICCSE), IEEE, Piscataway, NJ, 2013, pp. 703–707.

[17] F. F. Yao, *Speed-up in dynamic programming*, SIAM J. Algebr. Discrete Methods, 3 (1982), pp. 532–540.

[18] R. Yuster and U. Zwick, *Fast sparse matrix multiplication*, ACM Trans. Algorithms, 1 (2005), pp. 2–13.