On Lower Bounds for the Matrix Chain Ordering Problem[*]

Phillip G. Bradford[†] Venkatesh Choppella[‡] Gregory J. E. Rawlins[§]

November 17, 1993

Abstract

This paper shows that the radix model is not reasonable for solving the Matrix Chain Ordering Problem. In particular, to have an n-matrix instance of this problem with an optimal parenthesization of depth $\Theta(n)$ in the worst case requires the matrix dimensions to be exponential in n. Considering bit complexity, a worst case lower bound of $\Omega(n^2)$ is given. This worst case lower bound is parameterized and, depending on the optimal product tree depth, it goes from $\Omega(n^2)$ down to $\Omega(n \lg n)$. Also, this paper gives an $\Omega(n \lg n)$ work lower bound for the Matrix Chain Ordering Problem for a class of algorithms on the atomic comparison model with unit cost comparisons. This lower bound, to the authors' knowledge, captures all known algorithms for solving the Matrix Chain Ordering Problem, but does not consider bit operations. Finally, a trade-off is given between the bit complexity lower bound and the atomic comparison based lower bound. This trade-off basically shows that hard instances for the comparison based model are easy instances for the bit complexity model and vice versa.

1 Introduction

The *matrix chain ordering problem* (MCOP) is to find the cheapest way to multiply a chain of n matrices, where the matrices are pairwise compatible but of varying dimensions [11]. There has been significant research on the MCOP, take for example [6, 7, 8, 10, 12, 13, 15, 16, 18, 20, 21, 24, 25, 26, 27, 28, 29, 31]. The MCOP is also the focus of much pedagogy because of its amenability to an elementary dynamic programming solution. The dynamic programming paradigm is based on the *principle of optimality*. This principle is: for a structure to be optimal all of its well-formed substructures must also be optimal.

At first glance, it seems possible that an algorithm on the fixed radix based model might be faster than the present algorithms for the MCOP. This paper dashes hopes of using algorithms on a fixed or logarithmic radix model to tackle the MCOP in full generality. But, considering

[*]This is an update of Technical Report # 391, Oct. 1993, Indiana University. Hardcopy is available or a postscript file is obtainable via internet: cs.indiana.edu:/usr/ftp/pub/techreports/tr391.ps.Z. This technical report includes some Scheme programs available from the authors. Please address correspondence to Phil Bradford.

[†]Department of Computer Science, Indiana University, 215 Lindley Hall, Bloomington, Indiana 47405. email: bradford@cs.indiana.edu.

[‡]Department of Computer Science, Indiana University, 215 Lindley Hall, Bloomington, Indiana 47405. email: choppell@cs.indiana.edu.

[§]Department of Computer Science, Indiana University, 215 Lindley Hall, Bloomington, Indiana 47405. email: rawlins@cs.indiana.edu.

instances of the MCOP limited to those that can be represented in a fixed radix model, Hu and Shing's algorithm for the MCOP [20, 21] is asymptotically optimal.

A naïve information-theoretic lower bound fails: solving the MCOP distinguishes one of a Catalan number of possible valid parenthesizations. It is well known that the n^{th} Catalan number, to within a polynomial factor, is $\Theta(4^n)$, and this is the number of ways to parenthesize n items. This gives a $\Theta(n)$ information-theoretic lower bound. All algorithms for the MCOP and their analyses explicitly or implicitly assume that an n-matrix instance of the MCOP can have an optimal solution that is any of the Catalan number of parenthesizations. These assumptions are made without any other considerations, for example see the references above. This paper shows that to have an instance of the MCOP with an optimal parenthesization of depth $\Theta(n)$ in the worst case requires the matrix dimensions to be exponential in n. That is, to generate certain solutions for the MCOP, we must have very expensive inputs. Therefore, if we want to have all parenthesizations as possible solutions to the MCOP, then efficiency is of extreme importance.

Further, on the atomic comparison based model an $\Omega(n \lg n)$ lower bound is also given that assumes that the matrix product costs are atomic. That is, the matrix product costs can only be compared with each other and every comparison is of constant cost.

To the authors' knowledge there were no lower bounds for the MCOP better than $\Omega(n)$ prior to this paper. This paper assumes all matrix dimensions are distinct and takes all logarithms as base 2.

1.1 Main Results of this Paper

First, this paper shows that instances of the MCOP that have specific optimal parenthesizations of depth $\Theta(n)$ require the matrix dimensions to be exponential in n. This gives an $\Omega(n^2)$ lower bound considering bit operations.

Next, without considering bit complexity, is an $\Omega(n \lg n)$ work lower bound for the matrix chain ordering problem. To the authors' knowledge, the class of problems that this lower bound applies to includes all presently known algorithms for solving the MCOP. This lower bound shows that Hu and Shing's $O(n \lg n)$ algorithm [19, 20, 21] is asymptotically optimal and Ramanan's parallel algorithm [26, 27] is an $O(\lg^3 n)$ factor from optimal. In addition, Bradford, Rawlins and Shannon's parallel algorithm [7, 8] is an $O(\lg n)$ factor from optimal and would be asymptotically optimal if an asymptotically optimal $O(\lg n)$ time parallel algorithm for computing row minima on totally monotone matrices is found [1, 3, 2].

Finally, this paper gives a trade-off between the bit complexity lower bound and the atomic comparison based lower bound. This trade-off is essentially based on the optimal product tree depth. It shows that hard instances of the comparison based model are easy instances for the bit complexity model and vice versa.

1.2 Previous Results

Prior to this paper algorithms and their analyses didn't account for bit operations for solving the MCOP. However, finding lower bounds for the MCOP has been the focus of some research.

A. C.-C. Yao's work on decision trees has played an important role in the development of good lower bounds for a variety of problems. Building on Yao's work, Ramanan gives tight lower bounds to problems *related* to the matrix chain ordering problem in [24]. In addition, in [25], Ramanan shows these techniques give a lower bound such that, "If we could extend our lower bound technique to bounded degree algebraic decision trees, we would have a tight $\Omega(n \lg n)$ lower bound for the [MCOP]."

Ramanan's lower bound techniques work on problems that seem to be close relatives of the matrix chain ordering problem, although they have never clinched an $\Omega(n \lg n)$ lower bound for it.

1.3 Structure of this Paper

Section 2 gives a class of instances of the MCOP whose optimal product trees are of depth $\Theta(n)$ and these instances must have matrix dimensions exponential in n. Section 3 shows that there are instances of the MCOP that require $\Omega(n \lg n)$ work on the comparison based model assuming atomic product costs among matrices. This model appears to include all presently known algorithms for the MCOP.

2 Arbitrary Instances of the Matrix Chain Ordering Problem

This section shows there are instances of the MCOP that must have matrix dimensions exponential in n. Subsection 2.1 gives an algorithm for generating instances of the MCOP that have optimal alternating products. This algorithm is shown to generate such instances with exponentially large dimensions. Subsections 2.2, 2.3, 2.4, and 2.5 give a class of instances of the MCOP with dimensions asymptotically as large as those generated by the algorithm of Subsection 2.1. Subsection 2.6 generalizes the class of hard instances of the MCOP given in Subsections 2.2, 2.3, and 2.4.

An associative product of the form

$$((\cdots(((M_1 \bullet M_2) \bullet M_3) \bullet M_4) \cdots) \bullet M_n)$$

is called a *linear* product and an associative product of the form:

$$(M_1 \bullet ((M_2 \bullet ((M_3 \bullet \cdots \bullet M_{n-3}) \bullet M_{n-2})) \bullet M_{n-1})) \bullet M_n$$

is called an *alternating* product. Products of the form,

$$(((M_1 \bullet M_2) \bullet (M_3 \bullet M_4)) \bullet ((M_5 \bullet M_6) \bullet (M_7 \bullet M_8)))$$

are called *full balanced* trees since their binary tree representation is a balanced full tree.

The depth of a parenthesization is the depth of the tree representing the associative product. The tree of a (trivial) one matrix associative product has depth 1. The depth of a linear product and an alternating product of n matrices is $\Theta(n)$ and the depth of a balanced full tree product is $\Theta(\lg n)$.

An instance of the MCOP can be any list of $n + 1$ integers. This paper follows the standard assumption that every possible parenthesization is the solution of some instance of the MCOP.

2.1 Generating Instances of the MCOP

This subsection gives an algorithm for generating special instances of the MCOP. This algorithm is shown to produce instances of the MCOP with exponential dimensions in n.

Given an instance of the MCOP we will look at the corresponding problem of finding an optimal triangulation of a convex polygon with integer vertices.

Theorem 1 (Deimel and Lampe [14]; Hu and Shing [20]) Determining the optimal order to multiply n matrices can be done by finding an optimal triangulation of a corresponding $(n+1)$-gon.

See also [11].

Any chain of n pairwise compatible matrices $M_1, M_2, \cdots, M_n$, has $n+1$ dimensions which we write as $d_1, d_2, \cdots, d_{n+1}$. Matrix M_i is of dimension $d_i \times d_{i+1}$ and this paper takes the cost of multiplying a $d_i \times d_j$ matrix by a $d_j \times d_k$ matrix as $d_i d_j d_k$. Following the correspondence given by Theorem 11, the matrix dimensions are mapped to polygon weights, and the polygon weights are labeled by their relative size. In an $(n+1)$-gon, the ith largest weight is w_i, so overall the smallest weight is w_1 and the largest weight is w_{n+1}.

Given an instance of the MCOP, the corresponding $(n+1)$-gon is made by putting the dimensions, in their given order, above the x-axis at a height corresponding to their size, and placing equal length edges between d_i and $d_{(i \bmod (n+1))+1}$, for i such that $n+1 \geq i \geq 1$, see [11, 20]. (To aviod intersecting lines, some geometric manipulation may have to be done.) Now the dimensions are renumbered as described above and they are called polygon weights which are written as ws. By Theorem 1, finding an optimal triangulation of this $(n+1)$-gon solves the corresponding instance of the MCOP, where a triangle of the three vertices w_i, w_j, and w_k is taken to cost $w_i w_j w_k$.

Theorem 1 also leads directly to a simple parallel lower bound for the MCOP directly from Berkman et al.'s parallel processor lower bound for triangulating a monotone polygon [5]. That is, the parallel approximation algorithms of Czumaj [12] and Bradford [6] cannot run faster than $\Omega(\lg\lg n)$ time using $O(n \lg^c n)$ processors on the CRCW PRAM for c a constant where $c > 0$. Berkman et al. show that finding *any* triangulation of a monotone polygon can't be done faster than $\Omega(\lg\lg n)$ using $O(n \lg^c n)$ processors. This lower bound follows because these approximation algorithms give a triangulation of a monotone polygon.

For the next theorem we briefly go back and talk about matrix dimensions directly. Let d_i, d_{i+1}, and d_{i+2} be three adjacent matrix dimensions in an instance of the MCOP where $d_i < d_{i+1}$ and $d_{i+1} > d_{i+2}$, in addition, let $d_* = \min_{1 \leq i \leq n+1} \{ \ d_i \ \}$.

Theorem 2 (Chin [10]; and Hu and Shing [18, 17]) If

$$d_* d_i d_{i+2} + d_i d_{i+1} d_{i+2} \quad < \quad d_* d_i d_{i+1} + d_* d_{i+1} d_{i+2}$$

then the product $(M_i \bullet M_{i+1})$ is in an optimal parenthesization.

A proof of this theorem is left to the literature, see [10] and [18, 17] for different proofs. The basic intuition is that since d_* is the smallest matrix dimension, then replacing d_* with any larger matrix dimension will not change the inequality of Theorem 2.

Theorem 2 leads directly to sequential and parallel approximation algorithms for solving the MCOP to within 15.5 % of optimal [10, 18, 12, 6].

Given only a sequence of integers representing the depth of parentheses in an associative product (for example see the bottom row in Figure 1), then using a stack, for each parenthesis we can compute its matching parenthesis (see the top row in Figure 1). That is, we can compute the parenthesization of this associative product by solving the following all nearest smaller value (ANSV) problem [4, 5]: Given $w_1, w_2, \ldots, w_{n+1}$ drawn from a totally ordered set, for each w_i find the largest j, where $1 \leq j < i$, and smallest k where $i < k \leq n$, so that $w_j < w_i$ and $w_k < w_i$ if such values exist.

(	(	(	)	)	(	(	(	)	)	)	)
0	1	2	2	1	1	2	3	3	2	1	0

Figure 1: Parentheses and their Depths

Suppose w_j has the two nearest smaller values, w_i and w_k, where $i < j < k$. Then as in [4, 5] call w_i and w_k a *match* and denote each match by a pair $[w_i, w_k]$. The basic intuition behind solving the MCOP problem by applying the ANSV problem is to let matrix dimensions approximate the depth of parentheses [6].

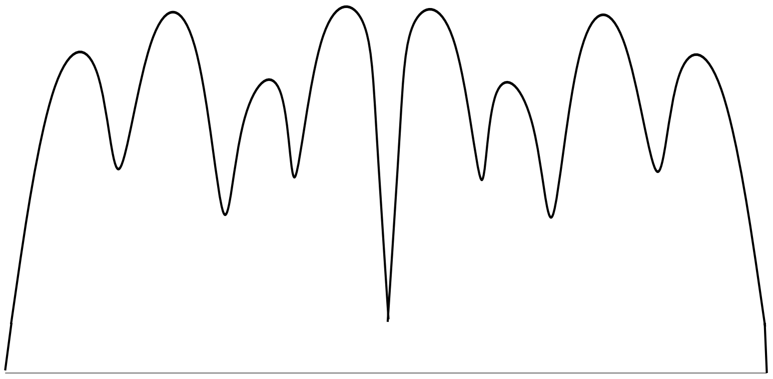

Figure 2: The Contour of an $(n + 1)$-gon Corresponding to an 8-Modal Instance of the MCOP

A *multi-modal* instance of the MCOP has a corresponding instance of the $(n + 1)$-gon triangulation problem that has a weight list that forms several "bumps." For example, the depths of the parentheses in Figure 1 have two bumps. The contour of an $(n + 1)$-gon corresponding to an 8-modal instance of the MCOP is given in Figure 2.

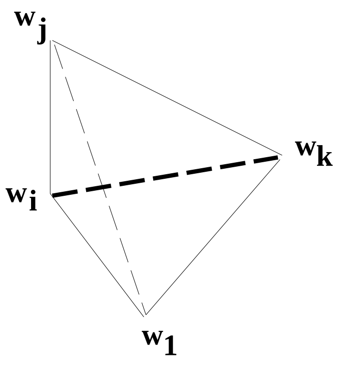

Figure 3: Computing a New Matrix Dimension w_j Given w_1, w_i, and w_k

Suppose in solving the ANSV problem the weight w_j has the match $[w_i, w_k]$. Then, considering polygons the statement of Theorem 2 generalizes to:

$$\text{if } w_1 w_i w_k + w_i w_j w_k < w_1 w_j w_k + w_1 w_i w_j \tag{1}$$

and products $(M_i \bullet \cdots \bullet M_{j-1})$ and $(M_j \bullet \cdots \bullet M_{k-1})$ are both in an optimal parenthesization, then the product $(M_i \bullet \cdots \bullet M_{j-1}) \bullet (M_j \bullet \cdots \bullet M_{k-1})$ is also in an optimal parenthesization.

Therefore, starting with w_1 and two other weights w_i and w_k where $w_1 < w_i$ and $w_1 < w_k$ from which we want to compute a new matrix dimension w_j where $w_j > w_i$ and $w_j > w_k$ (see Figure 3), then we can solve inequality (1) for some w_j. In particular, solving for the least integer w_j that is guaranteed to satisfy inequality (1) gives w_j as,

$$\left\lceil \frac{w_1 w_i w_k}{w_1 w_i + w_1 w_k - w_i w_k} \right\rceil \tag{2}$$

If expression (2) is zero or less, then there is no possible triangulation we can construct that satisfies inequality (1), so choosing such a w_j will be of no use. In addition, if expression (2) is indeterminate, then inequality (1) does not hold.

```
Alternating-Monotone-Polygon($w_1$){
    $k \leftarrow 2$; $i \leftarrow 3$;
    $w_k \leftarrow 1 + w_1$;
    $w_i \leftarrow 1 + w_k$;
    polygon $\leftarrow \{w_1, w_i, w_k\}$;
    $p_1 \leftarrow w_1 w_i w_k$;
    $p_2 \leftarrow w_1(w_i + w_k) - w_i w_k$;
    while ($p_2 \geq 0$) {
        $y \leftarrow Ceiling(p_1/p_2)$;
        $w_k \leftarrow w_i$;
        $k \leftarrow i$;
        $w_i \leftarrow y$;
        $i \leftarrow i + 1$;
        polygon $\leftarrow Concatenate(y,$ polygon$)$;
        $p_1 \leftarrow w_1 w_i w_k$;
        $p_2 \leftarrow w_1(w_i + w_k) - w_i w_k$;
    }
    return(polygon);
}
```

Figure 4: An Algorithm for Generating MCOP Instances with Optimal Alternating Products

Several important things should be noted about recursively generating new w_j values. For example start with some w_1, next assign $w_k = w_1 + 1$ and $w_i = w_k + 1$, finally generating w_j as above. Next, using the three weights w_1, w_j, w_k try to generate the next higher weight and place it appropriately on the list of polygon nodes. Generating weights in this fashion and appropriately alternating them from side to side gives a simple monotone polygon [23]. Taking particular care to put the odd numbered weights on the left side of the polygon and the even numbered weights on the right side of the polygon gives an *alternating* monotone polygon, also see [18].

The algorithm in Figure 4 seems to be due to Chin or Hu and Shing, but to our knowledge it was never published[1]. The algorithm in Figure 4 is invoked with some $w_1 \geq 2$. Modifying the concatenation function (concatenating on the front of the list polygon, on the back, etc.) in this algorithm allows the generation of a variety of different polygons.

In the algorithm in Figure 4, eventually, $w_1w_i + w_1w_k - w_iw_k$ becomes negative and remains negative for all larger values of w_i and w_k. Therefore, this algorithm will eventually stop. In addition, assuming $w_1(w_i + w_k) > w_iw_k$ gives $(w_i + w_k) > (w_iw_k)/w_1$ which implies $2\max\{\ w_i, w_k\ \} > (w_iw_k)/w_1$. Now, without loss, assuming $w_i > w_k$, which means $2w_1 > w_k = \min\{\ w_i, w_k\ \}$, therefore we can choose any w_j larger than or equal to expression (2) to satisfy inequality (1). But, choosing a very large w_j will backfire since after one more iteration of the algorithm in Figure 4, w_i is assigned w_j and w_k is assigned w_i so $w_i > 2w_1$ and $w_k > 2w_1$ which makes inequality (1) fail, thus stopping the algorithm.

Lemma 1 To ensure local optimality, for all weights in any monotone alternating $(n+1)$-gon it must be that, $w_{i+1} - w_i > w_i - w_{i-1}$.

Proof: Given any four weights that form a quadrilateral in an optimal alternating product, say w_1, w_2, w_3 and w_4, then the following must hold to have local optimality:

$$\begin{aligned} & w_1w_2(w_2+w_3) \geq w_2w_3(w_1+w_2) \\ \Longrightarrow\quad & w_3w_4(w_1-w_2) \geq w_1w_2(w_3-w_4) \\ \Longrightarrow\quad & w_3w_4(w_2-w_1) < w_1w_2(w_4-w_3) \end{aligned}$$

and since, $w_3w_4 > w_1w_2$ it must be that $w_4 - w_3 > w_2 - w_1$. Now, the lemma follows by induction. □

Simple examination of the proof of the last lemma and noting that $w_2 - w_1 \geq 1$ implies $w_4 - w_3 \geq 2$ holds for local optimality in any monotone alternating $(n+1)$-gon. If any monotone alternating $(n+1)$-gon is optimal, then it always must be locally optimal. This is the principle of optimality.

By modifying the algorithm in Figure 4 it turns out that locally optimal $(n+1)$-gons can be generated with their weights growing at most linearly.

Corollary 1 Given a locally optimal $(n+1)$-gon, then it must be that $w_4 - w_3 \geq 2$.

The last lemma and corollary hold in general, but let's go back to the details of the algorithm in Figure 4.

Now, the question arises as to how many alternating vertices this algorithm can generate before it is forced to stop. The next answer to this question also gives a lower bound on the size of the vertex weights this algorithm generates. With the algorithm in Figure 4 in mind, take the following inequality again,

$$w_j \quad \geq \quad \left\lceil \frac{w_1w_iw_k}{w_1w_i + w_1w_k - w_iw_k} \right\rceil \tag{3}$$

[1]See the appendix for a Scheme program implementing this algorithm.

If $w_i > 2w_1$ and $w_k > 2w_1$, then the denominator of inequality (3) becomes negative, therefore no new value w_j that has a positive contribution to a polygon can be generated. In addition, without loss say $w_i > w_k$, which gives

$$\begin{aligned}\left\lceil \frac{w_1 w_i w_k}{w_1 w_i + w_1 w_k - w_i w_k} \right\rceil &\geq \frac{w_1 w_i w_k}{2w_1 w_i - w_i w_k} \\ &= \frac{w_1 w_k}{2w_1 - w_k}\end{aligned}$$

and $2w_1 > w_k > w_1$, therefore, since we are generating a bound for w_j we bound the ratio w_j/w_k below by

$$\begin{aligned}\frac{\left(\frac{w_1 w_k}{2w_1 - w_k}\right)}{w_k} &= \frac{w_1}{2w_1 - w_k} \\ &= c_k\end{aligned}$$

Since $w_1 > 2w_1 - w_k$ we know that $c_k > 1$ and this means $w_j/w_k \geq c_k$. That is, w_j must be larger than or equal to w_k by a multiplicative factor of c_k. Now, following the algorithm, if $w_2 = w_1 + 1$ and $w_3 = w_1 + 2$, then w_5 is at least $w_3 + 2$ by Corollary 1. Next, computing the minimal possible value for c_3 gives:

$$\begin{aligned}c_3 &\geq \frac{w_1}{2w_1 - (w_1 + 2)} \\ &= \frac{w_1}{w_1 - 2}\end{aligned}$$

and by the definition of c_j, we know that $w_5 \geq c_3 w_3$, which is $w_5 \geq w_1(w_1+2)/(w_1-2)$. Computing a bound on c_5 gives

$$\begin{aligned}c_5 &\geq \frac{w_1}{2w_1 - w_1(w_1 + 2)/(w_1 - 2)} \\ &= \frac{w_1 - 2}{w_1 - 6} \\ &\geq \frac{w_1}{w_1 - 4}\end{aligned}$$

and $c_7 \geq w_1/(w_1 - 8)$ by substituting $(w_1 - 2)/(w_1 - 6)$, for w_k in the expression $w_1/(2w_1 - w_k)$. In fact, $(w_1 - y)/(w_1 - x) \geq w_1/(w_1 - x + y)$, for $w_1 > x \geq y$ since multiplying both sides by the product of the denominators gives $xy - y^2 \geq 0$.

Therefore, we can inductively show that,

$$c_j \geq \frac{w_1}{w_1 - 2^{j-2}} \tag{4}$$

and since $w_{j+1} \geq c_{j-1} w_{j-1}$ we know that by the time we compute $c_{\lceil \lg w_1 \rceil} + 1$, the right side of inequality (4) becomes negative or indeterminate. Therefore, if we want to have a product of at least n alternating elements from the algorithm in Figure 4, then we need w_1 such that $\lceil \lg(w_1) \rceil + 2 \geq n$. Or equivalently, $2^{\lceil \lg(w_1) \rceil + 2} \geq 2^n$, and this means $4w_1 \geq 2^n$ so $w_1 \geq 2^{n-2}$.

Theorem 3 Generating an instance of the MCOP with an optimal alternating product, using the algorithm in Figure 4, gives an instance of the MCOP that has dimensions that are exponential in n.

Directly by Theorem 2 the algorithm in Figure 4 produces a polygon that has an optimal alternating product. There can be other alternating polygons that have weights of size less than or equal to those given by the algorithm in Figure 4, but not by much. This is shown the next subsections.

2.2 The Exponential Size of a Class of Inputs for the MCOP

This subsection shows that there are instances of the MCOP that must have their dimensions exponential in n.

The notion of H-arc and V-arc, from Hu and Shing [19, 20], is central to lots of work on the MCOP. Given four weights $w_{i+1} > w_i > w_{i-1}$ and w_{i-2} which form a quadrilateral in a monotone $(n+1)$-gon, then there are two possibilities in triangulating this quadrilateral. Exactly one of the arcs w_i—w_{i-1} or w_{i+1}—w_{i-2} but not both, exist in any such triangulation of this quadrilateral. Considering our numbering of the weights, since w_i—w_{i-1} is horizontal, it is an H-arc and similarly because w_{i+1}—w_{i-2} is vertical it is a V-arc, see Figure 5.

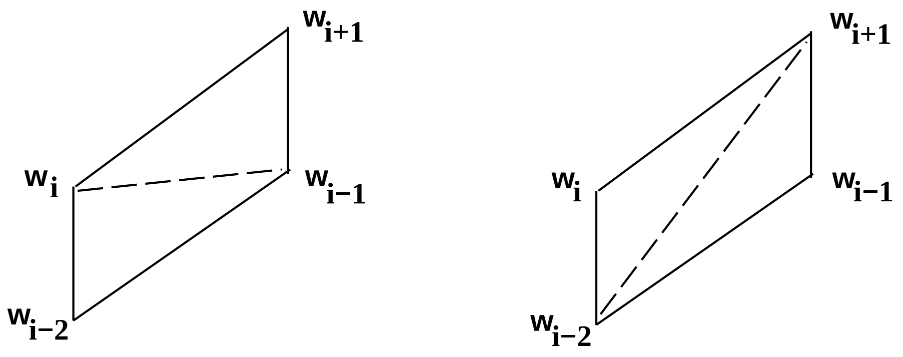

Figure 5: An H-arc and a V-arc in a Monotone $(n+1)$-gon

In general, take any four weights w_i, w_j, w_s, and w_t inscribing a quadrilateral in an $(n+1)$-gon. With this, the formal definition of H-arcs and V-arcs is [19, 20],

- w_s—w_t is an H-arc iff $\min\{ w_i, w_j \} < \min\{ w_s, w_t \}$ and $\max\{ w_s, w_t \} < \max\{ w_i, w_j \}$.
- w_s—w_t is a V-arc iff $\min\{ w_i, w_j \} > \min\{ w_s, w_t \}$ and $\max\{ w_s, w_t \} > \max\{ w_i, w_j \}$.

This formal definition includes squares inscribable in multi-modal $(n+1)$-gons.

Theorem 4 (Hu and Shing [20]) Any $(n+1)$-gon whose optimal triangulation corresponds to the solution of an instance the MCOP is made of only H-arcs and V-arcs.

The next corollary is an immediate consequence of the last theorem.

Corollary 2 All monotone $(n+1)$-gons that correspond to instances of the MCOP with optimal alternating products have only H-arcs.

Triangles in a simple monotone $(n+1)$-gon can contribute to or detract from an alternating product. If a triangle contributes nothing and detracts nothing, then $w_1 w_j(w_k + w_i) - w_i w_k(w_1 + w_j) = 0$. In order to have as many alternating products as possible, we include the product made up by the dimensions w_i, w_j, and w_k. (See Hu and Shing [21, Page 233] for a similar notion.)

The contribution of a triangle with vertices w_i, w_j, and w_k, relative to w_1, such that $w_j = \max\{ w_i, w_j, w_k \}$, is measured by

$$t[1]_j \quad = \quad w_1 w_j w_k + w_1 w_i w_j - (w_1 w_i w_k + w_i w_j w_k)$$

For a single monotone polygon, we have $k = j-1$ and $i = j-2$. The $t[1]$s are directly based on inequality (1). In fact, the $t[1]$ values are the key to the approximation algorithms for the MCOP [10, 18, 12, 6].

Now, by inequality (1) we have,

if $t[1]_j < 0$, then the product cost $w_{j-2}w_{j-1}w_j$ is a burden in the polygon p
if $t[1]_j > 0$, then the product cost $w_{j-2}w_{j-1}w_j$ contributes to the polygon p

These contributions or burdens can be considered to be relative to either the triangles $w_{j-2}w_{j-1}w_j$ or the H-arcs w_{j-1}—w_{j-2}.

In general, the contribution of the triangle with vertices w_j, w_{j-1}, and w_{j-2}, relative to w_s, where $\min\{ w_j, w_{j-1}, w_{j-2} \} > w_s \geq w_1$ is measured by:

$$t[s]_j \quad = \quad w_s w_j w_{j-2} + w_s w_j w_{j-1} - (w_s w_{j-1} w_{j-2} + w_j w_{j-1} w_{j-2})$$

$$\begin{array}{ccccc}
t[1]_4 & t[1]_5 & t[1]_6 & \cdots & t[1]_{n+1} \\
 & \wedge & \wedge & \cdots & \wedge \\
 & t[2]_5 & t[2]_6 & \cdots & t[2]_{n+1} \\
 & & \wedge & \cdots & \wedge \\
 & & t[3]_6 & \cdots & t[3]_{n+1} \\
 & & & & \wedge \\
 & & & \ddots & \vdots \\
 & & & & \wedge \\
 & & & & t[n-2]_{n+1}
\end{array}$$

Figure 6: A t-table for an Alternating Monotone Instance of the MCOP

A table of all t values for an instance of the MCOP is a *t-table*. For example, Figure 6 is a t-table for an alternating monotone $(n+1)$-gon.

The final total cost of an instance of the MCOP that has an optimal alternating product is,

$$\sum_{i=4}^{n+1} (t[i-3]_i + w_i w_{i-3}(w_{i-1} + w_{i-2}))$$

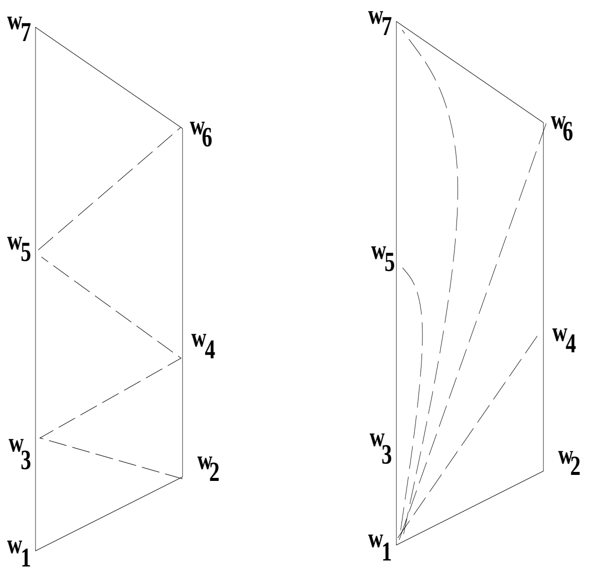

Figure 7: Polygon Representation of Alternating and Linear Products

Next is a lemma that illustrates some key ideas behind the values of the ts. This lemma is central for intuition about the t-table.

Take the monotone alternating 7-gon in Figure 7 which has the following $t[1]$ values,

$$t[1]_4 = \underline{w_1w_2w_4} + w_1w_3w_4 - (\overline{w_1w_2w_3} + \overline{w_2w_3w_4})$$

$$t[1]_5 = \underline{w_1w_3w_5} + w_1w_4w_5 - (w_1w_3w_4 + \overline{w_3w_4w_5})$$

$$t[1]_6 = \underline{w_1w_4w_6} + w_1w_5w_6 - (w_1w_4w_5 + \overline{w_4w_5w_6})$$

$$t[1]_7 = \underline{w_1w_5w_7} + \underline{w_1w_6w_7} - (w_1w_5w_6 + \overline{w_5w_6w_7})$$

Where for appropriate i and k, the expression $\underline{w_1w_iw_{i+2}}$ is the cost of a triangle in the linear product in Figure 7 and $\overline{w_jw_kw_{k+1}}$ is the cost of a triangle in the alternating product of the same figure. Notice that the product costs without underlines or overlines cancel out.

This leads directly to a proof of the next lemma,

Lemma 2 Given a monotone alternating polygon P, then $\sum_{i=4}^{n+1} t[1]_i$ measures the difference between the cost of the linear product of P and the cost of the alternating product of P.

This lemma also holds for all t values in a particular range, see Figure 8. In Figure 8, suppose $0 > t[3]_6 + t[3]_7 + t[3]_8 + t[3]_9$. Then in the subpolygon between w_3 and w_9, the optimal triangulation is not the alternating triangulation. Hence such an entire monotone alternating $(n+1)$-gon cannot have an optimal alternating triangulation. This is due to the principle of optimality, because if an alternating triangulation is optimal, then any alternating subtriangulation must be optimal. In general, taking the partial prefix sum of all rows of such a t-table, if any of these partial sums are negative, then an alternating triangulation is not optimal.

Corollary 3 Let P be an alternating monotone $(n+1)$-gon. If $0 > \sum_{i=4}^{k} t[1]_i$ for some k such that $n+1 \geq k \geq 4$, then the alternating triangulation of P is not optimal.

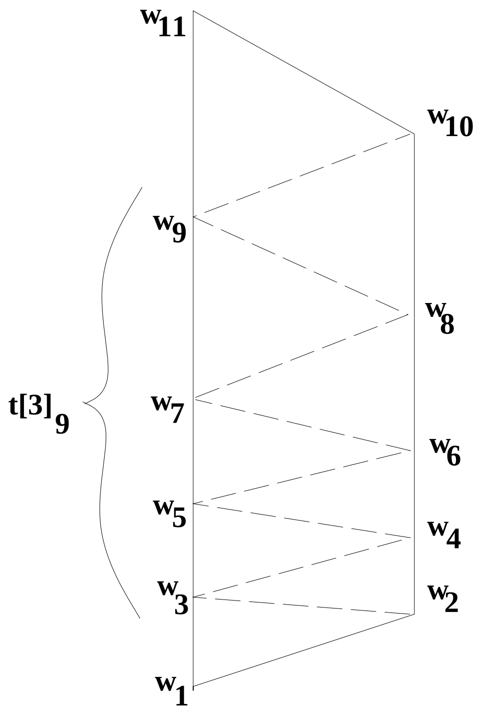

Figure 8: $t[3]_9$ in an Alternating Monotone Polygon

Lemma 3 Given an alternating monotone polygon with at least 4 weights, we have $t[1]_i < t[2]_i < \cdots < t[i-4]_i < t[i-3]_i$.

Proof: Take $t[s]_i = w_s w_i(w_{i-1} + w_{i-2}) - w_{i-1}w_{i-2}(w_s + w_i)$ and $t[u]_i = w_u w_i(w_{i-1} + w_{i-2}) - w_{i-1}w_{i-2}(w_u + w_i)$ where $u > s$. Taking their difference gives,

$$t[u]_i - t[s]_i \quad = \quad [w_u - w_s](w_i w_{i-1} + w_i w_{i-2} - w_{i-1}w_{i-2})$$

and since $w_i > w_{i-1} > w_{i-2}$ and $w_u > w_s$ in our numbering of the weights in an alternating monotone polygon, the lemma holds.
□

Lemma 3 encompasses a facet of the principle of optimality. In particular, if the local triangulation measured by $t[i-3]_i$ is negative, then the two triangles formed by the weights w_i, w_{i-1}, w_{i-2}, and w_{i-3} with an H-arc between w_{i-1} and w_{i-2} are not locally optimal.

Looking at the relationships of the t values, we note that if $t[1]_i \geq 0$ for all i such that $n+1 \geq i \geq 4$, then the algorithm in Figure 4 asymptotically optimally generates such monotone alternating polygons.

Given an alternating monotone polygon, a *block* is any t-table that has all positive t values. A block grows at least as fast as a polygon generated by the algorithm in Figure 4 from the same w_1.

Theorem 5 Given any w_1, the algorithm in Figure 4 generates the most compact alternating triangulation of a block.

Proof: As a consequence of expression (2), at each iteration the algorithm in Figure 4 includes the least possible integer relative to w_1 in the polygon. In addition, initially we choose the second and third dimensions as small as possible.

All local triangulations must be optimal, otherwise by Lemma 3 all t values can't be positive. Specifically, if all of the $t[1]$ values are positive, then all $t[i]$ for $i > 1$ are also positive. Also, if for any valid i and j, say $t[i]_j$ is negative, then $t[1]_j$ is negative. It follows by induction that the algorithm generates the "most compact" polygon, so all t values are positive.
□

The columns of a t-table corresponding to an instance of the MCOP are numbered by increasing values starting with 4 and going up through $n+1$. In some sense, the ith column represents the triangle with the three vertices w_i, w_{i-1}, and w_{i-2}. Column i corresponds to w_i, the ith largest weight. It is important to notice that going from left to right in a t-table for a multi-modal $(n+1)$-gon, the columns are of varying sizes. This means that column i and column $i+1$ may not represent adjacent dimensions in a multi-modal matrix chain. As an example, take the multi-modal $(n+1)$-gon in Figure 9.

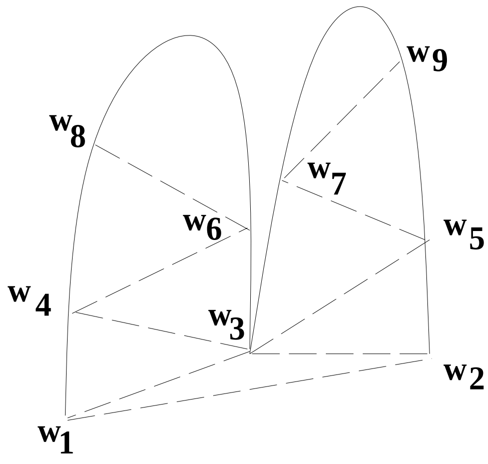

Figure 9: A Polygon with Two Extended Canonical Sets of Columns

Take a multi-modal instance of the MCOP with H-arcs in different "bumps." In this case, each "bump" must be considered separately in a t-table. This is because the key to the growth of the size of the weights is the number of H-arcs eventually above one another. Therefore, an *extended canonical* set of columns is a set of columns of a t-table that are all directly associated with one bump of a weight list. For example, Figure 9 contains a polygon that has a t-table with two extended canonical sets of columns. This figure assumes that each of the two bumps in the weight list forms an optimal alternating product on its own [20].

In particular, the columns of weights w_4, w_6, and w_8 form one extended canonical set and the columns of weights w_5, w_7, and w_9 form the other set. Figure 10 gives the t-table for the bi-modal polygon in Figure 9. It is important to note that between $t[1]_8$ and $t[3]_8$ there is no t value due to the structure of the polygon in Figure 9.

$$\begin{array}{ccccc}
t[1]_5 & t[1]_6 & t[1]_7 & t[1]_8 & t[1]_9 \\
 & & \wedge & \wedge & \wedge \\
 & & t[2]_7 & & t[2]_9 \\
 & & & \wedge & \wedge \\
 & & & t[3]_8 & t[3]_9
\end{array}$$

Figure 10: The t-table of the Bi-Modal Polygon of the Last Figure

Some polygons with optimal triangulations may have V-arcs interspersed with H-arcs. In such polygons, the only possible chance that w_{i-1}—w_{i-2} is an H-arc in an optimal triangulation is when $t[i-3]_i \geq 0$. Of course, if $t[i-3]_i \geq 0$, then the H-arc w_{i-1}—w_{i-2} is locally optimal, but $t[i-3]_i \geq 0$ does not indicate that the H-arc w_{i-1}—w_{i-2} is in a well formed substructure of the entire optimal triangulation. This follows directly from the principle of optimality.

For the rest of this paper, we only consider t-tables as depicted in Figure 6. That is, instead of considering t-tables as in Figure 10 we can just consider each extended canonical set of columns in such a t-table which is of the form as the one in Figure 6.

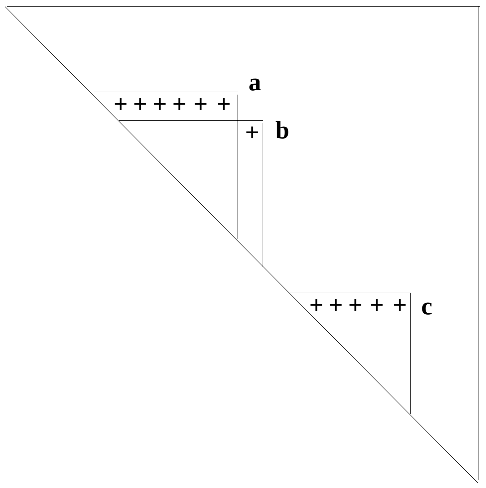

Figure 11: A t-table with Subblocks a, b and c

Subblocks are maximal blocks in a t-table. All subblocks are along the largest diagonal of a t-table. Figure 11 shows three subblocks of a t-table, where blocks a and b are intersecting. The *size* of a subblock is the number of rows, or equivalently the number of columns, of the subblock.

The next theorem illuminates the struggle between local and global optimality that, in some sense, characterizes the difficulty of solving the MCOP.

Theorem 6 For all valid i, the value $t[i-3]_i$ must be positive in any monotone $(n+1)$-gon that has an optimal alternating product.

A proof is trivial, since if a polygon is not locally optimal, then by the principle of optimality it cannot be globally optimal. It is important to keep in mind that Theorem 6 only holds when an optimally triangulated monotone $(n+1)$-gon has a H-arc between w_{i-1} and w_{i-2}.

Theorem 6 is tantamount to saying that all instances of monotone $(n+1)$-gons with optimal alternating products are locally made up of blocks. This is because each $t[i-3]_i$ value is the contribution of a block consisting of four weights. The only two types of monotone alternating $(n+1)$-gons corresponding to alternating instances of the MCOP are blocks or blocks above one another. Therefore, to have an optimal alternating monotone polygon we must have at least trivial blocks down the largest diagonal. That is, an optimal alternating monotone polygon must be made up of blocks and these blocks must "cover" the diagonal.

2.2.1 Generating Instances of the MCOP

Instances of the MCOP can be generated in a variety of ways. For instance, start by applying the algorithm in Figure 4 to generate a list of weights $l = w_1, w_2, \cdots, w_{n+1}$. Then, for some k such that $n-1 > k > 1$, remove weights $w_1, w_2, \cdots, w_k$ and call this new list l'. The list l' has $n+1-k$ elements. Next, run the algorithm in Figure 4 with an initial weight w_* such that $w_* > w_1$ and generate a new alternating weight list l'' which we cut off below w_{k+1} being sure to preserve local optimality between the top of l'' and the bottom of l'. Now we can insert l'' into l' below the H-arc between w_{k+1} and w_{k+2} giving a new polygon. This technique generates instances of the MCOP that exhibit the "father-son" relationship of [21], which was used for the sequential solution of the MCOP by Hu and Shing [20, 21]. In parallel, different methods can be applied for dealing with these instances [7, 9].

Instances of the MCOP with some negative t values can also be generated with a similar method to the one just outlined. For instance, we can start with an $(n+1)$-gon generated by the algorithm of Figure 4. Next, take the top weight w_{n+1} and lower it while making sure that the polygon still has an optimal alternating product. In this case, some of the t values will become negative. Lowering w_{n+1} to much will give an $(n+1)$-gon with an optimal linear product. Other weights can also be lowered. In addition binary search can be used to find weights such that there are some negative t while an alternating product is still optimal.

Generalizing these techniques to generate multi-modal $(n+1)$-gons follows from a main theorem of Hu and Shing. This is Theorem 11 in Subsection 2.6. Its application is straightforward, so it is not discussed here.

2.3 Columns with Non-Negative Sums

This subsection shows that if there are $\Omega(\lg^{1+\epsilon} n)$ total columns in a t-table with non-negative sums, then the corresponding instance of the MCOP has exponentially large dimensions.

Lemma 4 Given an instance of the MCOP with an optimal alternating monotone product, if its t-table has a subblock of size $\Omega(\lg^{1+\epsilon} n)$, for $\epsilon > 0$, then the subproduct of the MCOP that this corresponds to has dimensions of exponential size in n.

A proof of this lemma follows directly from Theorems 3 and 5, and Lemma 3 and the fact that for all appropriate i and j in such a subblock we have $t[i]_j \geq 0$. In addition, any such block will have dimensions that are exponential in size, since $2^{c \lg^{1+\epsilon} n}$ is (barely) exponential for the two constants $\epsilon > 0$ and $c > 0$.

Consider extended canonical sets with minor adaptation of the proof of Theorem 3. That is, for any set of columns the weights must grow exponentially to have positive $t[1]$s. Now the argument of Lemma 4 immediately generalizes to the following theorem.

Theorem 7 Given a monotone instance of the MCOP with an optimal alternating product, if its t-table has $\Omega(\lg^{1+\epsilon} n)$ columns with all positive elements, then the subproduct of the MCOP that this corresponds to has dimensions of exponential size in n.

In this subsection we still must address columns containing negative elements but with non-negative sums. By Theorem 7, any optimal alternating instance of the MCOP that has $\Omega(\lg^{1+\epsilon} n)$ columns of all positive elements has exponential dimensions. Therefore, we next check the case where there are $\Omega(\lg^{1+\epsilon} n)$ columns containing non-negative sums, where these columns have some negative elements.

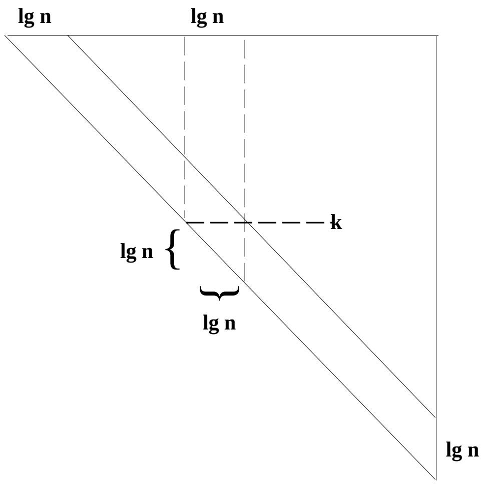

Figure 12: Blocks with Columns Having Negative Values

In the following discussions assume, without loss, that $i > \lg^{1+\epsilon} n$, so the ith column has $\Omega(i - \lg^{1+\epsilon} n)$ negative values. Otherwise there will be a subblock of size $\Omega(\lg^{1+\epsilon} n)$ of all positive t values in the t-table, see Figure 12. Wishing to prevent subblocks of size $\Omega(\lg^{1+\epsilon} n)$, there must be many columns close enough together that have negative values near the largest diagonal of the t-table. The largest diagonal contains the values $t[i-3]_i$ for all i such that $n+1 \geq i \geq 4$. In particular, having a subblock of size $\Omega(\lg^{1+\epsilon} n)$ bounded above by row k indicates that $t[k-3]_i > 0$ for all i, such that $r \lg^{1+\epsilon} n > i \geq k$ for some constants $\epsilon > 0$ and $r > 0$. (See Figure 12.)

Corollary 4 Given a monotone $(n+1)$-gon that has an optimal alternating product, if starting in the main diagonal any row does not have a negative value within $\Omega(\lg^{1+\epsilon} n)$ columns, then the matrix dimensions grow exponentially in n.

Otherwise, suppose the corollary does not hold. Then for some $\epsilon > 0$, there is a row has no negative values within $\Omega(\lg^{1+\epsilon} n)$ columns from the main diagonal. Then there would be a block with weights having exponential growth.

Next, we characterize the relationship of elements in columns by linear functions. In particular, the values in each column are related as a linear function.

By Corollary 2, finding a monotone polygon with an optimal alternating product we only have to consider H-arcs. Thus, from here on only consider columns in a t-table that correspond to H-arcs. So taking any such $t[i-3]_i > 0$, we know,

$$w_{i-3}(w_iw_{i-1} + w_iw_{i-2}) \quad > \quad w_{i-1}w_{i-2}w_{i-3} + w_iw_{i-1}w_{i-2}$$

but, without loss, assume that eventually for some $d \geq 4$ that $t[i-d]_i < 0$ and

$$w_{i-d}(w_iw_{i-1} + w_iw_{i-2}) \quad < \quad w_{i-1}w_{i-2}w_{i-d} + w_iw_{i-1}w_{i-2}$$

Therefore, in each column there are two constants m_i and b_i, for all valid $i > 3$, where

$$\begin{aligned} m_i &= w_iw_{i-1} + w_iw_{i-2} - w_{i-1}w_{i-2} \\ b_i &= w_iw_{i-1}w_{i-2} \end{aligned}$$

giving the equation $y = m_ix - b_i$. The equation $y = m_ix - b_i$ has slope m_i and it's not hard to see that $m_{i+1} > m_i$ and $b_{i+1} > b_i$. Also, note that in general the slopes are very steep since $m_i > w_iw_{i-1}$. From here on, we will treat the column equations and the elements of a column as the same.

The next relationship holds among the nodes in column i.

$$m_iw_1 - b_i < m_iw_2 - b_i < \quad \cdots \quad < m_iw_{i-4} - b_i < m_iw_{i-3} - b_i \tag{5}$$

As we have just seen, each column has a different linear equation that quantifies how elements in the column relate to each other. This is key in the subsequent developments in this section.

The intuition behind the final results in this subsection are now given for a t-table containing only trivial blocks. Trivial blocks consist of only 1 positive t value surrounded by negative t values. That is, if $t[i-3]_i > 0$ and $t[i-4]_i < 0$ and $t[i-3]_{i+1} < 0$, then $t[i-3]_i$ is a trivial block.

Suppose, there is some $(n+1)$-gon with $t[i-3]_i > 0$, $t[i-4]_i = 0$ and $t[i-5]_i < 0$ for all i such that $n+1 \geq i \geq 4$. (We choose $t[i-4]_i = 0$ to simplify our argument.) Notice that the t values in the main diagonal of the t-table are the only elements that are contributing positively. Insisting that the polygon at hand has an optimal alternating product and each column has a non-negative sum leads to the following argument.

For columns with non-negative sums and only one positive element, the next relationships must hold. Otherwise we will violate Theorem 6, hence we can't have the desired H-arcs. (This is because the H-arc w_{i-1}—w_{i-2} can only be in an optimal monotone $(n+1)$-gon if $t[i-3]_i \geq 0$.)

$$\begin{aligned} t[3]_6 &\geq \mid t[1]_6 \mid \\ t[4]_7 &\geq \mid t[2]_7 + t[1]_7 \mid \\ t[5]_8 &\geq \mid t[3]_8 + t[2]_8 + t[1]_8 \mid \\ t[6]_9 &\geq \mid t[4]_9 + t[3]_9 + t[2]_9 + t[1]_9 \mid \\ t[7]_{10} &\geq \mid t[5]_{10} + t[4]_{10} + t[3]_{10} + t[2]_{10} + t[1]_{10} \mid \\ t[8]_{11} &\geq \mid t[6]_{11} + t[5]_{11} + t[4]_{11} + t[3]_{11} + t[2]_{11} + t[1]_{11} \mid \\ t[9]_{12} &\geq \mid t[7]_{12} + t[6]_{12} + t[5]_{12} + t[4]_{12} + t[3]_{12} + t[2]_{12} + t[1]_{12} \mid \end{aligned}$$

Now, considering the columns (still assuming they each have a non-negative sum) and when $t[i-3]_i > 0$ we assume $t[i-4]_i = 0$ giving

$$\begin{aligned}
&w_3 - w_2 \geq (w_2 - w_1) && \Longrightarrow w_3 - w_2 \geq w_2 - w_1 \geq 1 \\
&w_4 - w_3 \geq (w_3 - w_2) + (w_3 - w_1) && \Longrightarrow w_4 - w_3 \geq 2(w_2 - w_1) \\
&w_5 - w_4 \geq (w_4 - w_3) + (w_4 - w_2) + (w_4 - w_1) && \Longrightarrow w_5 - w_4 > 4(w_2 - w_1) \\
&w_6 - w_5 \geq (w_5 - w_4) + (w_5 - w_3) + (w_5 - w_2) + (w_5 - w_1) && \\
& && \Longrightarrow w_6 - w_5 > 8(w_2 - w_1)
\end{aligned}$$

and in general, it must be that $w_{2i} > 2^i$. The central assumptions here are that in each column there is only one positive value $t[i-3]_i$ and $t[i-4]_i = 0$ so we can subtract $w_{i-4}m_i - b_i$ from the left side of the above inequalities and subtract each element on the right side of the above inequalities from $w_{i-4}m_i - b_i$ without changing anything. This same argument also holds for extended canonical sets of columns in a t-table.

The next lemma provides a generalization of this argument.

Lemma 5 (Columns with Non-Negative Sums) Given a monotone $(n+1)$-gon that has an optimal alternating product, if there are $\Omega(\lg^{1+\epsilon} n)$ columns with non-negative sums, then the matrix dimensions grow exponentially in n.

Proof: First, note that within every $O(\lg^{1+\epsilon} n)$ columns there must be a column with more than $\Omega(i - \lg^{1+\epsilon} n)$ negative values. Otherwise, there will be exponential growth in the weights by Lemma 4.

Next, by Theorem 6, for each i it must be that $t[i-3]_i \geq 0$ to have a monotone $(n+1)$-gon with an optimal alternating triangulation. In fact, to have columns with non-negative sums, it must be that $t[i-3]_i > 0$. Therefore, in t-table of a monotone $(n+1)$-gon with an optimal alternating product, it must be that $m_i w_{i-3} - b_i > 0$ for all valid i.

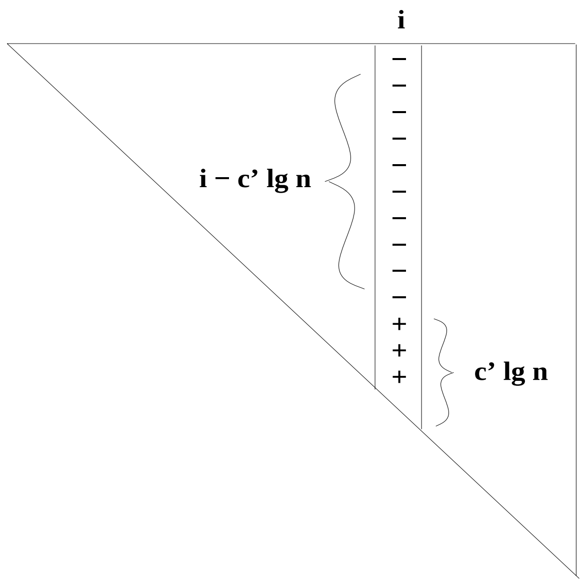

Figure 13: A Column with Mostly Negative Values but a Positive Sum

Now, we show that, for any constant c there is some constant c' such that: If within every $c \lg^{1+\epsilon} n$ columns the ith column has more than $i - c' \lg^{1+\epsilon} n$ negative values, then the weights don't grow exponentially (see Figure 13).

Consider the ith column. Since $m_i w_{i-3} - b_i > 0$, without loss suppose there is some $d \geq 4$ such that $m_i w_{i-d} - b_i = 0$. Therefore, $m_i w_j - b_i < 0$ for all j, such that $i - d - 1 \geq j \geq 4$ and $m_i w_k - b_i > 0$ for all k, such that $i - 3 \geq k \geq i - d + 1$.

Now, because $m_i w_{i-d} - b_i = 0$ we can subtract it from each positive t value and subtract each negative t value from its own copy of $m_i w_{i-d} - b_i$. Since the columns have positive sums

$$\sum_{s=i-d+1}^{i-3} (m_i w_s - b_i - (m_i w_{i-d} - b_i)) \quad \geq \quad \sum_{s=1}^{i-d-1} (m_i w_{i-d} - b_i - (m_i w_s - b_i))$$

must hold and this gives

$$\sum_{s=i-d+1}^{i-3} (w_s - w_{i-d}) \quad \geq \quad \sum_{s=1}^{i-d-1} (w_{i-d} - w_s)$$

which means

$$w_{i-3} - w_{i-d} \quad > \quad \sum_{s=x}^{y} (w_{i-d} - w_s)$$

for some x and y based on the size of d. For example, if $d = 4$, then $y = i - d - 1$ and $x = 1$.

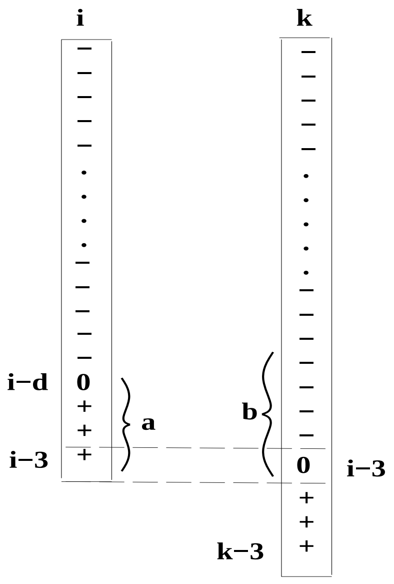

Figure 14: The Two Columns i and k

Now, we know that $i - c' \lg n \geq d \geq 4$, for some constant c', because there are less than $c' \lg^{1+\epsilon} n$ positive values in the column and more than $i - c' \lg^{1+\epsilon} n$ negative values. Therefore, $w_{i-3} - w_{i-d}$ is larger than or equal to the difference of w_{i-d} and more than $(i - c' \lg^{1+\epsilon} n)/c' \lg^{1+\epsilon} n$ other weights by the Pigeonhole Principle. This, with equation (5), means that

$$w_{i-3} - w_{i-d} \quad \geq \quad \sum_{s=i-c'\lg^{1+\epsilon} n - \left\lceil \frac{i-c'\lg^{1+\epsilon} n}{c'\lg^{1+\epsilon} n} \right\rceil}^{i-c'\lg^{1+\epsilon} n} (w_{i-d} - w_s)$$

Next, we will show that

Finally, by Corollary 4 there must be a column within $O(\lg^{1+\epsilon} n)$ columns with a weight that is at least $2(w_{i-3} - w_{i-d})$. This is because within $O(\lg^{1+\epsilon} n)$ columns there must be a t value such that $t[i-3]_k \leq 0$ for some column $k > i$ such that $k < d' \lg^{1+\epsilon} n$ for some $d' > 0$. Otherwise, by Lemma 5 there will be exponential growth in the weight size. Now, without loss, say that $t[i-3]_k = 0$ and we know that is larger than

$$w_{k-3} - w_{i-3} \quad \geq \quad \sum_{s=k-c'\lg^{1+\epsilon} n - \left\lceil \frac{k-c'\lg^{1+\epsilon} n}{c'\lg^{1+\epsilon} n} \right\rceil}^{k-c'\lg^{1+\epsilon} n} (w_{i-3} - w_s) \tag{6}$$

Now, since there are less than $O(\lg^{1+\epsilon} n)$ positive values in column i and for sufficiently large k we know that

$$k/\lg^{1+\epsilon} n = \Omega(\lg^{1+\epsilon} n)$$

indicating that

$$k - \lg^{1+\epsilon} n - \left\lceil \frac{k - c'\lg^{1+\epsilon} n}{c'\lg^{1+\epsilon} n} \right\rceil \quad > \quad i - 3 - (i-d) = d - 3$$

so there are at least a few terms of the form $(w_{i-3} - w_s)$ in equation 6 where $s < i - 3 - d' \lg^{1+\epsilon} n$ for some constant $d' > 0$, see Figure 14. In this figure, notice that the number of t values "covered" by $t[k-3]_k$ is larger than the number of t values "covered" by $t[i-3]_i$. That is, the difference $w_{i-3} - w_{i-d}$ marked by a is asymptotically not as large as the difference $w_{i-3} - w_s$ for lots of w_ss marked by b.

This argument continues inductively, increasing the weights by powers of 2 at each such column. Also, it holds for columns with non-negative sums that have fewer positive values than $\Theta(\lg n)$. This argument even holds in the case where progressive columns having non-negative sums have varying numbers of positive t values.

□

This lemma immediately generalizes to the case of a single extended canonical set of columns of a multi-modal $(n+1)$-gon. If there are columns with negative values (but non-negative sums) every $O(\lg^{1+\epsilon} n)$ columns or less, then the weights grow exponentially. This is because by Lemma 5, we will double the values of the weights at least every $O(\lg^{1+\epsilon} n)$ columns, giving an exponential increase in the size of the weights.

2.4 Columns with Negative Sums

This subsection plays columns with negative sums off of the sums of the rows of a t-table. In other words, if a t-table has many columns with negative sums, then some row must have a negative sum indicating that we can't have an optimal alternating instance of the MCOP by Corollary 3.

The last subsection showed that if any instance of the MCOP with an optimal alternating product has $\Omega(\lg^{1+\epsilon} n)$ columns in its t-table with non-negative sums, for $\epsilon > 0$, then it will have exponential growth in its dimensions. This means, the only remaining way to have an instance of the MCOP with an optimal alternating product that does not have exponential growth in its dimensions is to have $\Omega(n - \lg^{1+\epsilon} n)$ columns with negative sums. This subsection shows that if a t-table has $\Omega(n - \lg^{1+\epsilon} n)$ columns with negative sums, then either the top row of this t-table has a negative sum, or the weights grow exponentially. By Corollary 3, if the top row of a t-table has a negative sum, then the corresponding instance of the MCOP does not have an optimal alternating product.

The next theorem is elementary.

Theorem 8 If the sum of all the t values in a column of the t-table is negative, then the column has more negative t values than positive t values.

A proof of this theorem follows from Lemma 1. Theorem 8 implies that if the ith column has a negative sum, then $t[\lfloor i/2 \rfloor]_i < 0$.

Columns in the t-table and the lines representing the relationship of their elements are treated identically.

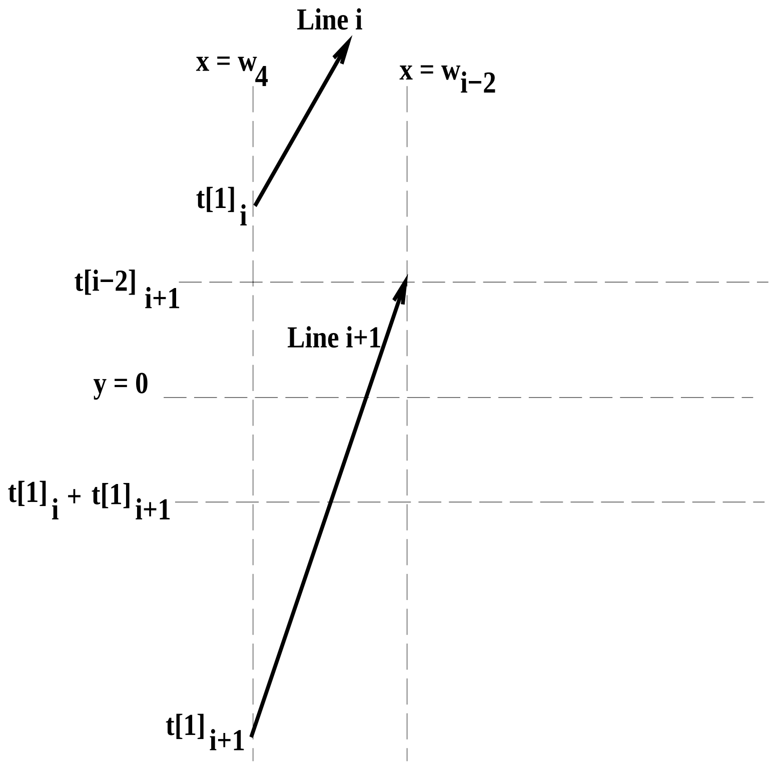

Figure 15: Lines i and $i+1$ have $t[1]_i \geq 0 > t[1]_{i+1}$ and $0 \geq t[1]_i + t[1]_{i+1}$

Assume column i has all positive values and suppose column $i+1$ has a negative sum, see Figure 15. Then by Theorem 8, it must be that $0 > t[\lfloor (i+1)/2 \rfloor]_{i+1}$. Also, it must be that $0 > t[1]_{i+1}$ since column $i+1$ has a negative sum. This means, if $t[1]_i + t[1]_{i+1} \leq t[\lfloor (i+1)/2 \rfloor]_{i+1}$, then $t[1]_i + t[1]_{i+1} < 0$. Therefore, in terms of the sum along the top row (of the corresponding t-table) that must be maintained to have an optimal alternating product, by Corollary 3, column $i+1$'s negative $t[1]$ value cancels out any gain by column i's positive $t[1]$ value. Recalling that the only chance of having subexponential growth of the weights is, if there are at least $\Omega(n - \lg^{1+\epsilon} n)$ columns with negative sums.

The intuition behind the next lemma is the following. Assume most columns have negative sums, then to maintain a positive sum in the top row of the t-table, we must show that the magnitude of the columns with positive $t[1]$s are larger than their (many) negative counterparts. But, for each column with a positive $t[1]$, there are

$$\Omega\left(\frac{n}{\lg^{1+\epsilon} n}\right)$$

columns with negative sums, for some $\epsilon > 0$. The next lemma shows that if $\Omega(n/\lg^{1+\epsilon} n)$ columns with negative sums don't "cancel out" one positive $t[1]$, then the weights grow exponentially. In fact, the next lemma proves that this even holds for only $\Omega(\lg^{1+\epsilon} n)$ such negative columns.

Lemma 6 Suppose the t-table of a monotone $(n+1)$-gon with an optimal alternating triangulation is such that $t[1]_i \geq 0 > t[1]_k$, for $i < k$. If there are $\Omega(\lg^{1+\epsilon} n)$ columns k, where $t[\lfloor k/2 \rfloor]_k < t[1]_i + t[1]_k$, then the weights grow exponentially.

Proof: Starting with $0 > t[s]_k - (t[1]_i + t[1]_k)$ for $k > i$ and let $s = \lfloor (k+1)/2 \rfloor$ giving

$$0 > (w_s - w_1)\left[w_k(w_{k-1} + w_{k-2})\right] + (w_1 - w_s)\left[w_{k-1}w_{k-2}\right] - w_1 w_{i-1}(w_i - w_{i-2}) + w_i w_{i-2}(w_{i-1} - w_1)$$

which is

$$w_i - w_{i-2} > (w_s - w_1)\left[\frac{w_k(w_{k-1} + w_{k-2}) - w_{k-1}w_{k-2}}{w_1 w_{i-1}}\right] + (w_{i-1} - w_1)\frac{w_i w_{i-2}}{w_1 w_{i-1}}$$

giving $w_i - w_{i-2} > c(w_{i-1} - w_1)$ for some constant $c > 1$.
□

Suppose the conditions of Lemma 6 don't hold. In particular, say $t[\lfloor (i+1)/2 \rfloor]_{i+1} \geq t[1]_i + t[1]_{i+1}$ and without loss say $0 > t[\lfloor (i+1)/2 \rfloor]_{i+1}$. Then we know that $0 > t[1]_i + t[1]_{i+1}$. Now, since $0 > t[1]_i + t[1]_{i+1}$, this forces the top row to have a negative sum because we must have $\Omega(n - \lg^{1+\epsilon} n)$ such columns. Therefore, by Corollary 3 if the top row has a negative sum, then we can't have an optimal alternating product.

The results of this subsection give the next theorem.

Theorem 9 Given a t-table with $\Omega(n - \lg^{1+\epsilon} n)$ columns with negative sums, then either there is exponential growth in the size of the weights or the corresponding instance of the MCOP does not have an optimal alternating monotone product.

2.5 The Exponential Nature of the MCOP

A proof of the next theorem follows from the results in the previous subsections.

Theorem 10 Any instance of the MCOP that has an optimal alternating product of n elements has matrix dimensions exponential in n.

The results of this section also apply to extended canonical weight lists. These results even apply to blocks in unimodal $(n+1)$-gons that are interspersed with optimal V-arcs. In this case, the exponential growth of weights is directly related to the number of H-arcs in an optimal triangulation from the top of such a bump down, see Figure 16.

2.6 Arbitrary Instances of the MCOP and the Lower Bounds

This subsection generalizes the results of the last subsection giving larger classes of costly instances of the MCOP.

Directly by the work of Hu and Shing [18, 20, 21], it is easy to see that for any possible parenthesization π of n elements, there is an instance of the MCOP that has π as an optimal parenthesization. In particular, consider the following theorem.

Theorem 11 (Hu and Shing [20]) Given a matrix dimension list $w_1, \ldots, w_{n+1}$ with the three smallest dimensions $w_1 < w_2 < w_3$, then the products $w_1 w_2$ and $w_1 w_3$ are in some associative product(s) in an optimal parenthesization.

Notice that Theorem 11 relies only on the position of the three smallest weights and their relative size. This theorem is useful as a divide-and-conquer tool for building balanced full tree instances of the MCOP.

The next corollary follows from Theorems 10 and 11. In particular, columns in the same extended canonical set of a t-table only force exponential growth when H-arcs are above one another (possibly interspersed with optimal V-arcs) in an optimal product. The next corollary is immediate considering the size of the inputs. And it turns out that the argument in Subsection 2.2 can be mimicked to get an exponential upper bound on the size of the matrix dimensions using the algorithm of Figure 4.

Corollary 5 In the worst case solving the MCOP takes $\Omega(n^2)$ bit operations.

Corollary 5 is asymptotically tight since in [20, 21] Hu and Shing give an algorithm for solving the monotone $(n+1)$-gon triangulation problem in $O(n)$ atomic operations. In addition, just to verify that an $(n+1)$-gon is monotone costs $\Theta(n)$.

The corresponding $(n+1)$-gon of any instance of a balanced full tree of the MCOP is only $O(\lg n)$ H-arcs deep anywhere, see Figure 16. Taking this argument further to a fixed radix model, Hu and Shing's algorithm is asymptotically optimal.

By the proof of Theorem 3 we see that if there are only $O(\lg n)$ H-arcs, then the lower bound on the size of the matrix dimensions is $O(2^{O(\lg n)})$, which is polynomial. We can use the algorithm in Figure 4 along with Theorem 11 to generate such instances. This means that n matrix instances of the MCOP that have optimal product trees of depth $O(\lg n)$ can have dimensions representable by $O(\lg n)$ bits each.

Taking into account Theorem 11 and then applying Theorem 10 to product trees of depth $O(\lg n)$ gives the following corollary.

Corollary 6 In the worst case just to read in any instance of the MCOP with an optimal product tree of depth $\Theta(\lg n)$ requires $\Omega(n \lg n)$ bit operations.

This corollary is the other end of the parameterized worst case lower bound. In particular, the depth of the number of desired H-arcs in an optimal product determine the size of the dimensions, hence the bit complexity of instances of the MCOP.

In the next section, lower bounds are given on the atomic comparison model that can be contrasted with the results of this section.

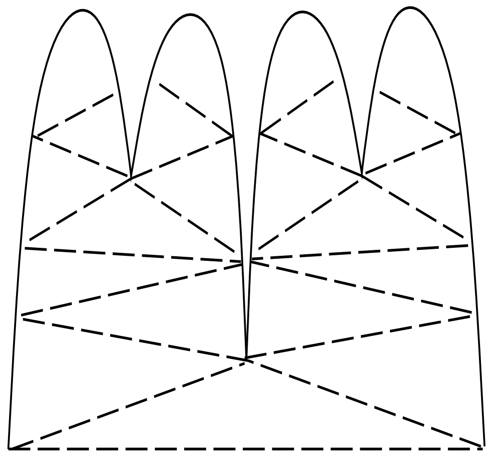

Figure 16: H-arcs in a Balanced Full Tree Instance of the MCOP

3 An $\Omega(n \lg n)$ Lower Bound on the Comparison Based Model

This section focuses on the comparison model where each comparison is of constant cost and all of the product costs are atomic. This section begins with a general tree evaluation problem, of which the MCOP is a special case. Subsection 3.1 starts with an algebraic model and Subsection 3.2 completes the results given here.

The general tree evaluation problem has an $\Omega(n \lg n)$ work lower bound. To the authors' knowledge, all algorithms that solve the MCOP also solve this general tree evaluation problem.

The lower bound of this section has its worst case for instances of the MCOP whose solution may be balanced full binary tree products. In contrast, the lower bound of the last section has its worst case for instances of the MCOP whose solution is an alternating product.

3.1 An Algebraic Model

A (finite) *semigroupoid* $(S, R, \bullet)$ is a nonempty finite set S, a binary relation $R \subseteq |S| \times |S|$ and there is a bijection from $|S|$ to the elements in S. There is an associative binary operator "$\bullet$" on R satisfying the following conditions [22]:

1. If $(a, b) \in R$, then $a \bullet b \in S$.
2. $(a \bullet b, c) \in R$ iff $(a, b \bullet c) \in R$ and $(a \bullet b) \bullet c = a \bullet (b \bullet c)$.
3. If $(a, b) \in R$ and $(b, c) \in R$, then $(a \bullet b, c) \in R$.

A *weighted* semigroupoid $(S, R, \bullet, pc)$ is a semigroupoid $(S, R, \bullet)$ with a non-negative product cost function pc such that if $(i, k) \in R$, then $pc(a_i, a_k)$ is the cost of evaluating $a_i \bullet a_k$.

An *associative product* is any product of the form $a_1 \bullet a_2 \bullet \cdots \bullet a_n$ such that $(i, i+1) \in R$, for $1 \leq i < n$. The minimal cost of evaluating an associative product $a_i \bullet a_{i+1} \bullet \cdots \bullet a_k$ is denoted by $sp(i, k)$. The minimal cost of an associative product can be computed as,

$$sp(i,k) = \min_{i \le j < k} \{ \ sp(i,j) + sp(j+1,k) + f(i,j,k) \ \} \tag{7}$$

where $f(i,j,k) = pc(a_i \bullet a_{i+1} \bullet \cdots \bullet a_j, \ a_{j+1} \bullet a_{j+2} \bullet \cdots \bullet a_k)$ and $sp(i,i) = g(i)$ where $g(i)$ is some function of i.

Given a weighted semigroupoid and an associative product $a_1 \bullet a_2 \bullet \cdots \bullet a_n$ the problem of finding $sp(1,n)$ is the *minimum parenthesization problem* on a weighted semigroupoid. This problem models many problems that are often solved using dynamic programming.

3.2 A Model for the MCOP

This subsection follows [6, 7, 9] very closely. Here a graph problem is given that can model many problems which have dynamic programming solutions. In fact, recurrence (7) can be solved by finding a shortest path on such a graph. Although this graph structure has been used as a framework for solving the MCOP, it can also be used for developing lower bounds.

Let T be an $n \times n$ dynamic programming table for finding an optimal associative product via recurrence (7). Element $T[i,k]$ represents the cheapest cost of generating the associative product $a_i \bullet \cdots \bullet a_k$. That is, for the MCOP, $T[i,k]$ represents the cheapest cost of the matrix product $M_i \bullet \cdots \bullet M_k$. In general, for any such T there is a graph D_n where the cost of a shortest path to node (i,k), denoted $sp(i,k)$, is the same as the final value of $T[i,k]$.

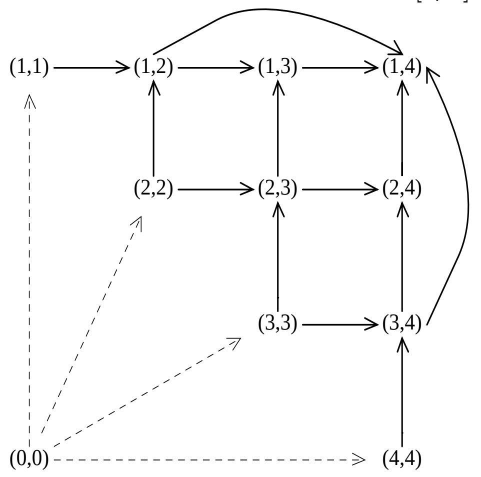

Figure 17: The Weighted Graph D_4

Given an associative product with n elements and an appropriately built D_n graph, finding a shortest path from $(0,0)$ to $(1,n)$ gives an optimal associative product. Finding this shortest path is tantamount to solving recurrence (7). Therefore, starting with a chain of n matrices, finding a shortest path from $(0,0)$ to $(1,n)$ in D_n solves the MCOP [6].

D_n has vertices in the set, $\{ \ (i,j) : 1 \le i \le j \le n \ \} \cup \{ \ (0,0) \ \}$ and edges,

$$\begin{aligned} \{ \ (i,j) \rightarrow (i,j+1) : 1 \le i \le j < n \ \} \quad & \cup \quad \{ \ (i,j) \uparrow (i-1,j) : 1 < i \le j \le n \ \} \\ & \cup \quad \{ \ (0,0) \nearrow (i,i) : 1 \le i \le n \ \} \end{aligned}$$

known as *unit* edges, together with the edges,

$$\{ (i,j) \Longrightarrow (i,t) : 1 < i < j < t \leq n \} \cup \{ (s,t) \Uparrow (i,t) : 1 \leq i < s < t \leq n \}$$

known as *jumpers*, see the jumper from $(1,2)$ to $(1,4)$ in Figure 17. The unit edge $(i,j) \rightarrow (i,j+1)$ represents the product $(a_i \bullet \cdots \bullet a_j) \bullet a_{j+1}$ and weighs $f(i,j,j+1)$. For the MCOP, $(a_i \bullet \cdots \bullet a_j) \bullet a_{j+1} = (M_i \bullet \cdots \bullet M_j) \bullet M_{j+1}$ and $f(i,j,j+1) = w_i w_{j+1} w_{j+2}$, which is taken as the cost of multiplying a $w_i \times w_{j+1}$ matrix and a $w_{j+1} \times w_{j+2}$ matrix. Similarly, the unit edge $(i,j) \uparrow (i-1,j)$ represents the product $M_{i-1} \bullet (M_i \bullet \cdots \bullet M_j)$ and costs $f(i-1,i-1,j) = w_{i-1} w_i w_{j+1}$. A shortest path to (i,k) through the jumpers $(i,j) \Longrightarrow (i,k)$ and $(j+1,k) \Uparrow (i,k)$ both represent the product $(a_i \bullet \cdots \bullet a_j) \bullet (a_{j+1} \bullet \cdots \bullet a_k)$ These jumpers weigh $sp(j+1,k) + f(i,j,k)$ and $sp(i,j) + f(i,j,k)$, respectively, where $sp(j+1,k)$ is the cost of a shortest path to node $(j+1,k)$ and for the MCOP $f(i,j,k) = w_i w_{j+1} w_{k+1}$ and

$$(a_i \bullet \cdots \bullet a_j) \bullet (a_{j+1} \bullet \cdots \bullet a_k) = (M_i \bullet \cdots \bullet M_j) \bullet (M_{j+1} \bullet \cdots \bullet M_k)$$

See Figure 17.

In D_n a *critical node* is (i,k) such that $[w_i, w_{k+1}]$ is an ANSV match. Two critical nodes on the same diagonal are *compatible* if no vertices other than $(0,0)$ can reach both of them by a unit path. Since a path of critical nodes represents a parenthesization, all critical nodes are compatible. Also, D_n has at most $n-1$ critical nodes and there is at least one path from $(0,0)$ to $(1,n)$ that includes all critical nodes [6].

All vertices and edges that can reach (i,t) by a unit path form the subgraph $D(i,t)$. When $D(i,j)$ has a monotonic weight list $w_i, \ldots, w_{j+1}$, then $D(i,j)$ is monotonic. A *band canonical subgraph* $D^{(i,t)}_{(j,k)}$ is the subgraph containing the maximal unit-edge-connected path of critical nodes beginning at critical node (j,k) and terminating at critical node (i,t) with the vertex set $\{ V[D(i,t)] - V[D(j+1,k-1)] \} \cup \{ (0,0) \}$ and associated edges. A canonical subgraph of the form $D^{(i,t)}_{(j,j+1)}$, is a *leaf* canonical subgraph and is written $D^{(i,t)}$. It has the same nodes and edges as $D(i,t)$. It turns out that these bands and leaf subgraphs together form trees [20, 21, 6].

Figure 18 shows a weight list and its subgraphs in D_n. The dashed lines represent four key ANSV matches and the four corresponding critical nodes are circled in D_n. Notice the tree structure that canonical graphs form in Figure 18.

A proof of the next theorem is straightforward examining recurrence (7).

Theorem 12 Any solution to recurrence (7) on D_n forms a tree of band and leaf subgraphs in D_n.

A proof of the next theorem is immediate since there are $\Omega(n^3)$ "f values" to look at in a D_n graph. This is because for all $1 \leq i \leq j < k \leq n$ there is an $f(i,j,k)$. For instance, an adversary could make it so that $f(i,j,k) = 1000$ except for the constants a, b and c where, $i = a$, $j = b$, $k = c$ and $f(a,b,c) = 1$. In addition, any path in D_n from $(0,0)$ to $(1,n)$ has exactly $n-1$ f values in it.

Theorem 13 Solving the minimum parenthesization problem on an arbitrary weighted semigroupoid costs $\Omega(n^3)$.

But, fortunately the structure of the MCOP constrains the various f values. In particular, we define the next subproblem of the minimum parenthesization problem on a weighted semigroupoid. Replace the two last rules from the definition of a weighted semigroupoid with the following two,

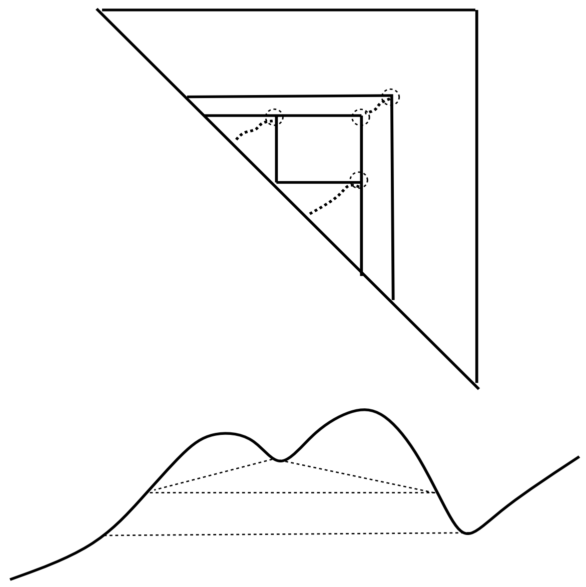

Figure 18: Two Leaf Subgraphs Inside an Band Subgraph with Critical Nodes Shown

CASE 1: If $[sp(i,i), sp(j,j)]$ forms an ANSV match, then $(a_i, a_j) \in R$

CASE 2: If $(a_i, a_j) \in R$ and $(a_{j+1}, a_k) \in R$, then let $a_s = a_i \bullet a_j$ and $a_t = a_{j+1} \bullet a_k$ giving $(a_s, a_t) \in R$.

First notice that this new structure is a weighted semigroupoid still solvable by recurrence (7). With these new restrictions, only special trees of associative products are allowed. Starting with a weighted semigroupoid, replacing the last two rules by CASE 1 gives a problem akin to the tree evaluation problem which can be solved in $\Theta(n)$ time.

Adding CASE 1 along with CASE 2 and its recursive application gives a general problem costing $\Omega(n \lg n)$ time to solve. This problem is a generalization of the tree evaluation problem. In particular, take the standard tree evaluation routines, and replace the binary operation by the minization operation of recurrence (7). The number of minimums in such a generalized tree evaluation or equivalently the number of jumpers along tree edges in D_n but not the number of jumpers in a canonical subgraph. The number of such tree edge jumpers are asymptotically bounded by $h(n)$ where

$$h(n) \leq \min_{1 \leq k < n} \{\lfloor (n-k)/2 \rfloor, \lfloor k/2 \rfloor\} + h(n-k) + h(k)$$

with the solution $h(n) = \Theta(n \lg n)$.

For example, take the following critical nodes in R via CASE 1:

$$R \;=\; \{\ (1,2),(3,4),(5,6),(7,8),(1,4),(5,8),(1,8)\ \}$$

All paths among these initial values in R, require the following set of f values,

$$\{\ f(1,1,2), f(3,3,4), f(5,5,6), f(7,7,8), f(1,2,4), f(5,6,8)\ \}$$

Now, applying CASE 2 gives the additional f values that must be considered in recurrence (7),

$$\{\ f(1,4,6), f(1,6,8), f(1,4,8)\ \}$$

Figure 19 shows some of the initial critical nodes in D_8 with their jumpers and edge weights. Notice that in Figure 19 node $\boxed{(1,6)}$ is not given above as a critical node, but jumpers to and from it in the present model must be considered, since $(7,8)$ and $(5,6)$ are both critical nodes. In addition, some of the jumpers have symmetric counterparts which are depicted as dashed jumpers without edge weights.

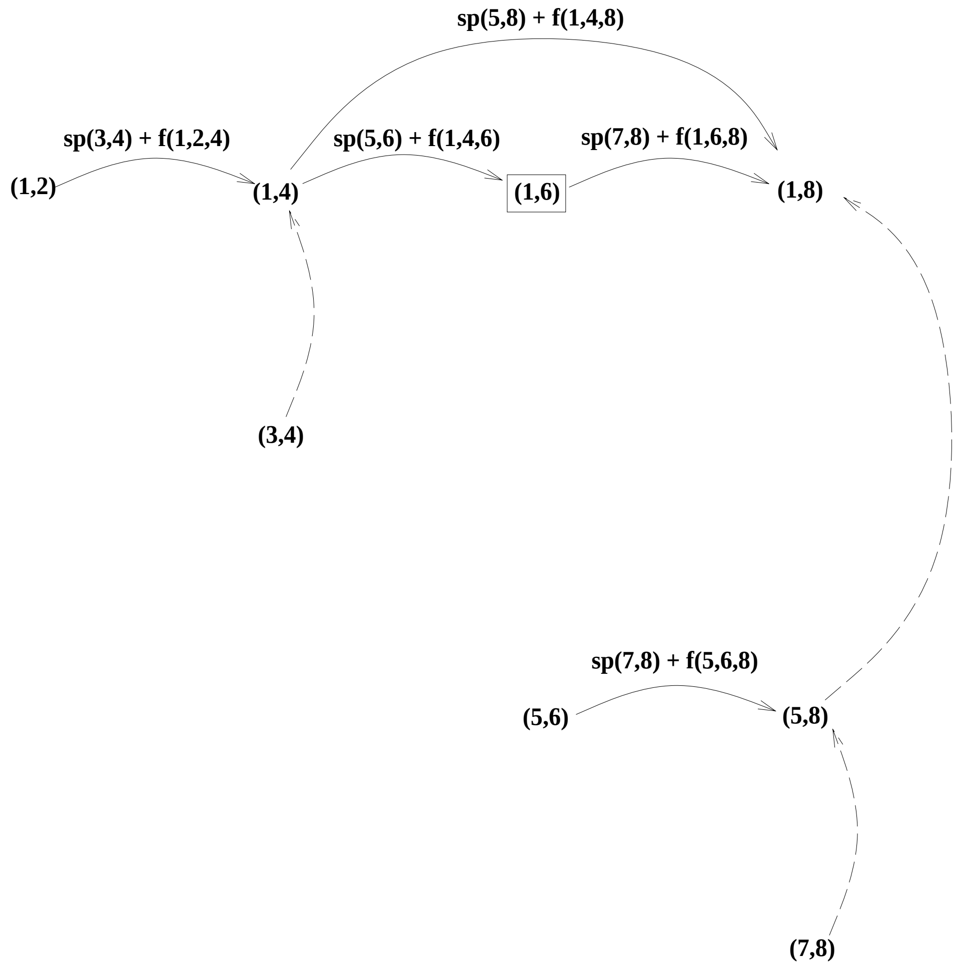

Figure 19: The Jumpers Necessary for a Full Balanced Binary Tree in D_8

Figure 19 has the tree of an alternating instance of the MCOP. This represents an instance of the MCOP that has as its base representation a full balanced tree. Assuming the matrix product costs are atomic, and assuming no other information about the matrix product costs means that solving recurrence (7) is the only way to solve the MCOP.

Simply put the number of tree edges that must be evaluated is $\Omega(n \lg n)$ to solve the MCOP. This is assuming that each edge weight must be compared independently and we have no information about them *a priori*. A proof of the next theorem follows directly from the results of this section.

Theorem 14 On the comparison based model, if we assume the cost of each f value is atomic, then solving the MCOP costs $\Omega(n \lg n)$.

Theorem 14 offers a lower bound for all algorithms the authors' are aware of for solving the MCOP. This lower bound is parameterized by the starting tree structure of an instance of the MCOP. This starting tree structure is only one of a small possible number of optimal product trees and it can be computed in $O(n)$ time. Therefore, some instances of the MCOP with linear optimal product trees do not apply to the worst case lower bound of the last section. On the other hand, if some instance of the MCOP has an optimal product tree of depth $\Omega(\lg n)$, then both the lower bound of this section and the lower bound of the last section apply. More interestingly, in the worst case, trees with alternating products of depth $\Omega(n)$ have bit based worst case lower bounds of $\Omega(n^2)$, but the lower bound for the comparison based model becomes $\Omega(n)$.

The main intuition of the trade-off is that in Figure 18 the larger the canonical subgraph representing an optimal alternating product the more costly it is by Theorem 10. On the other hand, the smaller the canonical subgraphs and the larger the tree of canonical subgraphs the more costly it is by Theorem 14.

4 Conclusions

This paper shows that the typical model for solving the MCOP is not reasonable. In particular, the comparison based model seems appropriate given the standard presentation of the Matrix Chain Ordering Problem. In this regard, the trade-off of the two main lower bounds given here highlights some inadequacies of the present analyses.

Although, if one insists on studying the atomic comparison based model for the MCOP, then to improve present algorithms, different properties of the MCOP must be exploited than those currently used. From the results of this paper, using a fixed or logarithmic radix model to improve these algorithms is not reasonable. But, other properties of the MCOP might give better bounds for this problem. For example, take the following,

Theorem 15 (Bradford et al. [9]) The problem of edge minimizing m jumpers in n unit rows can be reduced to the problem of finding the row minima of an $m \times n$ totally monotone matrix.

In [1] it is shown that solving the row minima problem sequentially in a $n \times m$ totally monotone matrix costs $\Theta(n)$. Theorem 15 shows there may be hope in using algorithms outside of the present model for solving the MCOP that may beat the $\Omega(n \lg n)$ atomic comparison lower bound.

5 Acknowledgments

Many thanks to the following people who were sources of excellent discussions and great inspiration: Alok Aggarwal, Randall Bramley, Ken Chiu, G. Neil Haven, T. C. Hu, Larry Larmore, Larry Moss, Paul W. Purdom Jr., Andrea Rafael, Shankar Swamy and Dirk Van Gucht.

References

[1] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, R. Wilber: "Geometric Applications of A Matrix Search Algorithm," *Algorithmica*, Vol. 2, 195-208, 1987.

[2] A. Aggarwal and J. Park: "Parallel Searching Multidimensional Monotone Arrays," to appear in the *Journal of Algorithms* and parts of 29th FOCS, 497-512, 1988.

[3] M. J. Atallah and S. R. Kosaraju: "An Efficient Parallel Algorithm for the Row Minima of a Totally Monotone Matrix," 394-403, SODA 1991. And to appear in the *Journal of Algorithms*.

[4] O. Berkman, D. Breslauer, Z. Galil, B. Schieber and U. Vishkin: "Highly Parallelizable Problems," *Symposium on the Theory on Computing*, 309-319, 1989.

[5] O. Berkman, B. Schieber and U. Vishkin: "Optimal Doubly Logarithmic Parallel Algorithms Based on Finding All Nearest Smaller Values," *J. of Algorithms*, Vol. 14, 344-370, 1993.

[6] P. G. Bradford: "Efficient Parallel Dynamic Programming," Extended Abstract in the Proceedings of the 30th *Allerton Conference on Communication, Control and Computation*, University of Illinois at Urbana-Champaign, 185-194, 1992. Full version submitted.

[7] P. G. Bradford, G. J. E. Rawlins and G. E. Shannon: "Matrix Chain Ordering in Polylog Time with $n/\lg n$ Processors," Full Version, Oct. 1992.

[8] P. G. Bradford, G. J. E. Rawlins and G. E. Shannon: "Matrix Chain Ordering in Polylog Time with $n/\lg n$ Processors," Technical Report # 369, December 1992, also updated version submitted.

[9] P. G. Bradford, G. J. E. Rawlins and G. E. Shannon: "Efficient Matrix Chain Ordering in Polylog Time," July, 1993.

[10] F. Y. Chin: "An $O(n)$ Algorithm for Determining Near-Optimal Computation Order of Matrix Chain Products," *Communications of the ACM*, Vol. 21, No. 7, 544-549, July 1978.

[11] T. H. Cormen, C. E. Leiserson and R. L. Rivest: *Introduction to Algorithms*, McGraw Hill, 1990.

[12] A. Czumaj: "An Optimal Parallel Algorithm for Computing a Near-Optimal Order of Matrix Multiplications," *SWAT*, Springer Verlag, LNCS # 621 , 62-72, 1992.

[13] A. Czumaj: "Parallel algorithm for the matrix chain product and the optimal triangulation problem," *STACS*, Springer Verlag, LNCS # 665, 294-305, 1993.

[14] L. E. Deimel, Jr. and T. A. Lampe: "An Invariance Theorem Concerning Optimal Computation of Matrix Chain Products," North Carolina State Univ. Tech Report # TR79-14.

[15] S.-H. S. Huang, H. Liu, V. Viswanathan: "Parallel Dynamic Programming," *Proceedings of the 2nd IEEE Symposium on Parallel and Distributed Processing*, 497-500, 1990.

[16] S.-H. S. Huang, H. Liu, V. Viswanathan: "A Sublinear Parallel Algorithm for Some Dynamic Programming Problems," *Theoretical Computer Science*, Vol. 106, 361-371, 1992.

[17] T. C. Hu: *Combinatorial Algorithms*, Addison-Wesley, 1982.

[18] T. C. Hu and M. T. Shing: "An $O(n)$ Algorithm to Find a Near-Optimum Partition of a Convex Polygon," *J. of Algorithms*, Vol. 2, 122-138, 1981.

[19] T. C. Hu and M. T. Shing: "Some Theorems about Matrix Multiplication," *Proceedings of the 21st Annual IEEE Symposium on the Foundations of Computer Science*, 28-35, 1980.

[20] T. C. Hu and M. T. Shing: "Computation of Matrix Product Chains. Part I," *SIAM J. on Computing*, Vol. 11, No. 3, 362-373, 1982.

[21] T. C. Hu and M. T. Shing: "Computation of Matrix Product Chains. Part II," *SIAM J. on Computing*, Vol. 13, No. 2, 228-251, 1984.

[22] M. Marcus: *Introduction to Modern Algebra*, Marcel Dekker, 1978.

[23] F. P. Preparata and M. I. Shamos: *Computational Geometry—An Introduction*, Springer Verlag, 1985.

[24] P. Ramanan: "Obtaining Lower Bounds Using Artificial Components," *Information Processing Letters*, Vol. 24, 243-246, 1987.

[25] P. Ramanan: "A New Lower Bound Technique and its Application: Tight Lower Bounds for a Polygon Triangularization Problem," *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, 281-290, 1991.

[26] P. Ramanan: "An Efficient Parallel Algorithm for Finding an Optimal Order of Computing a Matrix Chain Product," Technical Report, WSUCS-92-2, Wichita State University, June, 1992.

[27] P. Ramanan: "An Efficient Parallel Algorithm for the Matrix Chain Product Problem," Technical Report, WSUCS-93-1, Wichita State University, January, 1993.

[28] W. Rytter: "On Efficient Parallel Computation for Some Dynamic Programming Problems," *Theoretical Computer Science*, Vol. 59, 297-307, 1988.

[29] L. G. Valiant, S. Skyum, S. Berkowitz and C. Rackoff: "Fast Parallel Computation of Polynomials Using Few Processors," *SIAM J. on Computing*, Vol. 12, No. 4, 641-644, Nov. 1983.

[30] A. C.-C. Yao: "Lower bounds for Algebraic Computation Trees with Integer Inputs," *SIAM J. of Computing*, Vol. 20, No. 4, 655-668, 1991.

[31] F. F. Yao: "Speed-Up in Dynamic Programming," *SIAM J. on Algebraic and Discrete Methods*, Vol. 3, No. 4, 532-540, 1982.