A PARALLEL ALGORITHM FOR FINDING ALL MINIMAL MAXIMUM SUBSEQUENCES VIA RANDOM-WALK THEORY

By

ZHU WANG

Bachelor of Science in Industrial Foreign Trade
University of Electronic Science and Technology
Chengdu, China
1993

Master of Science in Information Systems
Dakota State University
Madison, SD
2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2015

A PARALLEL ALGORITHM FOR FINDING ALL MINIMAL MAXIMUM SUBSEQUENCES VIA RANDOM-WALK THEORY

Dissertation Approved:

Dr. H. K. Dai

Dissertation Adviser

Dr. Douglas R. Heisterkamp

Dr. M. H. Samadzadeh

Dr. Guoliang Fan

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my advisor, Dr. H. K. Dai, for his inspiring guidance, continuous encouragement, and valuable critiques throughout my program of study at Oklahoma State University.

I would also like to thank my committee members, Dr. Douglas R. Heisterkamp, Dr. M. H. Samadzadeh, and Dr. Guoliang Fan for their friendly guidance and helpful suggestion.

Lastly, I would like to thank my parents, my wife, and my son for their support and encouragement.

Name: Zhu Wang

Date of Degree: JULY, 2015

Title of Study: A PARALLEL ALGORITHM FOR FINDING ALL MINIMAL MAXIMUM SUBSEQUENCES VIA RANDOM-WALK THEORY

Major Field: Computer Science

Abstract: A maximum contiguous subsequence of a real-valued sequence is a contiguous subsequence with the maximum cumulative sum. A minimal maximum contiguous subsequence is a minimal contiguous subsequence among all maximum ones of the sequence. Time- and space-efficient algorithms for finding the single or multiple contiguous subsequences of a real-valued sequence with large cumulative sums, in addition to its combinatorial appeal, have major applications such as in bioinformatics, pattern matching, and data mining. We have designed and implemented a domain-decomposed parallel algorithm on cluster systems with Message Passing Interface that finds all minimal maximum subsequences of a random sample sequence from a normal distribution with negative mean. We find the structural decomposition of the sequence with overlapping common subsequences for adjacent processors that allow hosting processors to compute the minimal maximum subsequences of the sequence independently. Our study employs the theory of random walk to derive an approximate probabilistic length bound for common subsequences in an appropriate probabilistic setting, which is incorporated in the algorithm to facilitate the concurrent computation of all minimal maximum subsequences in hosting processors. We also present an empirical study of the speedup and efficiency achieved by the parallel algorithm with synthetic random data.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1. All Minimal Maximum Subsequences Problem

For a given sequence of real numbers, the *maximum subsequence problem* is the task to find the contiguous subsequence that has the maximum cumulative sum. The maximum subsequence problem has many applications in pattern matching, data mining, biomolecular sequence analysis, and other areas.

According to Jon Bently in *Programming Pearls* [Ben00], the maximum subsequence problem was proposed by Ulf Grenander at Brown University in the two-dimensional form when he was developing a pattern-matching procedure for the digitized pictures. He needed to find the rectangular subarray with the maximum sum that could be used as the maximum likelihood estimator of a certain kind of pattern in the picture. The two-dimensional problem is often called *maximum subarray problem*. The initial algorithm by Grenander to solve the maximum subarray problem ran in $O(n^6)$ time, so he simplified it to the one-dimensional form to obtain the insight into its structure.

Given a sequence $X = (x_\eta)_{\eta=1}^{n}$ of n real-valued terms, the cumulative sum of a non-empty contiguous subsequence $(x_\eta)_{\eta=i}^{j}$ is $\sum_{\eta=i}^{j} x_\eta$, where $1 \leq i \leq j \leq n$, and the cumulative sum of the empty subsequence is 0. All subsequences addressed in our study are contiguous, so the terms

"subsequence" and "supersequence" will hereafter abbreviate "contiguous subsequence" and "contiguous supersequence", respectively. The subsequence *S* with the maximum cumulative sum is the maximum subsequence of *X*. All nonempty prefixes and suffixes of *S* must have non-negative cumulative sums, otherwise we can find a subsequence of *S* with higher cumulative sum by removing its prefix or suffix having the negative cumulative sum. A minimal maximum subsequence of X is the minimal one with respect to subsequential containment among all maximum subsequences of *X*. The minimal constraint does not allow non-empty prefix or suffix that has non-positive cumulative sum. According to the above definition, all non-empty prefix or suffix of the minimal maximum subsequence must have positive cumulative sums. It is easy to see that X contains no positive term if and only if the empty subsequence is the unique minimal maximum subsequence of X. In our study, empty minimal maximum subsequence is not allowed.

For example, consider the sequence

$$X = (-1, 4, -3, -1, 5, -4, 2, 3, -2, 1),$$

the cumulative sums of the prefixes are plotted in Figure 1.1.

The sequence *X* has two maximum subsequences: $(4, -3, -1, 5, -4, 2, 3)$ and $(5, -4, 2, 3)$, and the minimal maximum subsequence is $(5, -4, 2, 3)$. The other maximum subsequence has a non-empty prefix with zero cumulative sum, so it is not minimal.

Grenander's initial algorithm to solve the minimal maximum subsequence problem was in cubic time, then Michael Shamos designed an $O(n \log n)$ algorithm. After Shamos described the problem and its history to statistician Jay Kadane, he gave a linear time algorithm [Ben00]. The similar descriptions of the algorithm were also given by Bates and Constable [BC85] and Manber [Man89]. Smith also designed a recursive algorithm that ran in linear time based on divide-and-conquer strategy [Smi87].

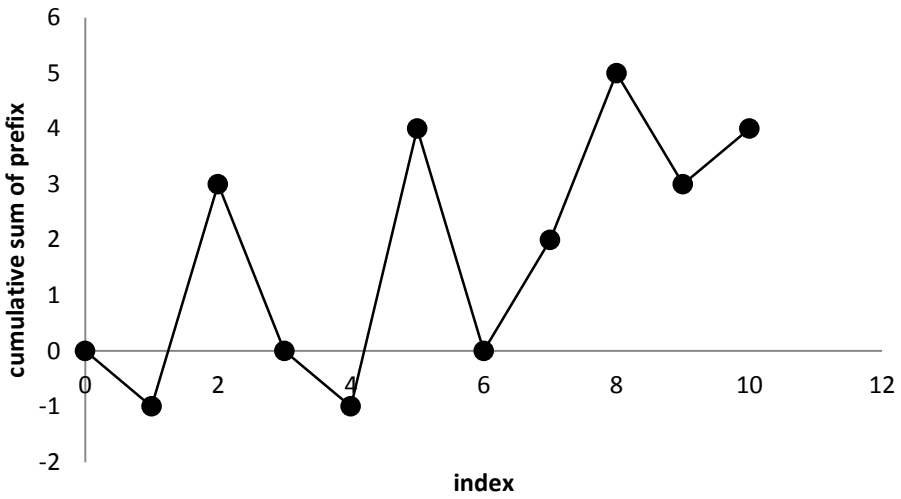

Figure 1.1: Cumulative sums of prefixes

Like many other maximum problems, it is worth finding the 2^{nd}, 3^{rd}, . . . , k^{th} minimal maximum subsequence of X. But what is exactly the 2^{nd} minimal maximum subsequence? In the above example, the cumulative sum of the minimal maximum subsequence S_1 is 6. Subsequence (5) has the 2^{nd} largest cumulative sum among all the subsequences of X. However subsequence (5) overlaps S_1, so it is hard to know how much independent information it can provide. Instead, subsequence (4) is the 2^{nd} minimal maximum subsequence which is disjoint from S_1. Very often in practical applications it is required to find many or all pairwise disjoint subsequences having cumulative sums above a prescribed threshold. Intuitively, we define the sequence of all successive minimal maximum subsequences $(S_1, S_2, \ldots, S_i)$ of X inductively as follows:

1. The sequence S_1 is a (non-empty) minimal maximum subsequence of X, and

2. Assume that the sequence $(S_1, S_2, \ldots, S_i)$ of non-empty subsequences of X, where $i \geq 1$, has been constructed, the subsequence S_{i+1} is a (non-empty) minimal subsequence (with respect to subsequential containment) among all non-empty maximum

subsequences (with respect to cumulative sum) that are disjoint from each of $\{S_1, S_2, \ldots, S_i\}$.

As in the definition of minimal maximum subsequence, the minimality constraint on the maximum cumulative sums of S_1, S_2, ... is equivalent to the nonexistence of non-empty prefixes nor suffixes with non-positive cumulative sums. For the above example, the sequence of all successive minimal maximum subsequences of X are $S_1 = (5, -4, 2, 3)$, $S_2 = (4)$, and $S_3 = (1)$.

If the order is not important, we can define *all minimal maximum subsequences problem* to be the task that finds the set of all successive minimal maximum subsequences. Ruzzo and Tompa proposed a linear time sequential algorithm for this problem [RT99]. If the k^{th} minimal maximum subsequence is to be selected, then sorting algorithms can be applied after Ruzzo and Tompa's algorithm.

The single or all minimal maximum subsequences can be solved sequentially in linear time, parallel algorithms are the only ways to speed up. Alk and Guenther developed a parallel algorithm on the parallel random access machine (PRAM) to solve minimal maximum subsequence problem in $O(\log n)$ time using a total of $O(n)$ operations [AG91]. Similar $O(\log n)$ time parallel algorithms were also developed on the PRAM model in [Wen95], [PD95], [QA99]. Alves, Caceres, and Song presented a parallel algorithm on the bulk synchronous parallel/coarse grained multicomputer (BSP/CGM) model that runs in $O(n/p)$ parallel time with p processors [ACS03].

For the all minimal maximum subsequences problem, Dai and Su presented a parallel algorithm in $O(\log n)$ time and $O(n)$ operations on the PRAM model [DS06]. Alves, Caceres, and Song

[ACS13] developed a parallel algorithm on the BSP/CGM model of p processors in $O(n/p)$ computation time and $O(1)$ communication rounds.

The detailed background and researches for maximum subarray problem in two-dimensional form were described in Bae's dissertation [Bae07]. Our studies only focus on maximum subsequences in one-dimensional form.

1.2. Applications

The original application of maximum subarray problem was for the pattern matching in digitized pictures which are consists of the two-dimensional array of pixels. After assign different scoring schemes, then some pattern matching problems can be converted into maximum subarray problem. For example, the brightness of a pixel can be measured with selected relative luminance formula. Then to find the brightest area of the picture is actually to find the maximum subarray. But the best algorithm so far runs in near cubic time [TT98], so applications in computer vision based on maximum subarray problem are still a big challenge.

Efficient algorithms for computing the sequence of all successive minimal maximum subsequences of a given sequence are essential for statistical inference in large-scale biological sequence analysis. In biomolecular sequences, high (sub)sequence similarity usually implies significant structural or functional similarity (the first fact of biological sequence analysis in [Gus97]). When incorporating good scoring schemes, this provides a powerful statistical paradigm for identifying biologically significant functional regions in biomolecular sequences ([KA90], [KD92], [KA93], and [RT99]), such as transmembrane regions [BBN$^+$92], DNA binding domains [KB92], and regions of high charges [KBB91], [KB92] in protein analyses.

A common approach is to employ an application-dependent scoring scheme that assigns a score to each single constituent of an examined biomolecular sequence, and then find all successive

minimal maximum subsequences of the underlying score sequence having large cumulative sums above a prescribed threshold. For example, transmembrane regions are rich in hydrophobic residues, so Kyte and Doolittle [KD92] designed hydropathy index for 20 amino acids ranging from -4.5 (least hydrophobic) to +4.5 (most hydrophobic). Then the transmembrane regions can be observed for the maximal subsequences. Karlin and Brendel [KB92] used the hydropathy index ranging from -5 to +3 to the human β_2−adrenergic receptor sequence, and observed the maximal subsequences are corresponding to the known transmembrane regions.

A theory of logarithmic odds ratios, developed in [KA90], yields an effective logarithmic likelihood-ratio scoring function in this context. The non-positivity of the expected score of a random single constituent tends to delimit unrealistic long runs of contiguous positive scores. Karlin and Brendal assigned the log likelihood ratio to each residue of the human β_2-adrenergic receptor sequence, and they found that the maximum subsequences were similar to the ones obtained using hydropathy index, but were more pronounced. Karlin and Altschul [KA93] used the same scoring function in other protein sequences, tried to find multiple disjoint maximum subsequences, and the problem can be solved with Ruzzo and Tompa's algorithm.

Pasternack and Roth [PR09] applied maximum subsequence problem in extracting article text from HTML pages. To extract the article from the original HTML document is not easy because the large amount of less informative or unrelated elements such as navigation menus, forms, visual styles, images, advertisements, and etc. are also mixed in the same document. Pasternack and Roth tokenized the page into tags, words, and symbols, and each token is assigned a value by the token classifier. For example tags will have negative values, while words and symbols have positive values. Then the maximum subsequence of the page token sequence is corresponding to the text block.

The maximum subsequence problem can also be applied in data mining [TVP05], [Bae07]. Here is a trivial example. The sequence X is (0, 1, 2, 0, 0, 0), where the i^{th} term for $1 \leq i \leq 6$ of X is the number of customers whose age is within $[10i,\ 10(i+1))$, and bought "ham". After subtract 1/2 from each term, then the maximum subsequence of X suggests that the corresponding age group from 20 to 39 is more likely to buy ham.

1.3. **Dissertation Outlines**

In chapter II, we review the sequential and parallel algorithms that solve minimal maximum subsequence problem and all minimal maximum subsequences problem.

In chapter III, we study the structural decompositions of sequence X. The different structural decompositions of X lead to different parallel algorithms to find all minimal maximum subsequences. We find a decomposition scheme that can decrease the communications among different processors. The basic idea is to introduce an overlapping common subsequence for two adjacent processors.

In chapter IV, we analyze the bound on the length of common subsequences probabilistically for random sequences of *normally*-distributed terms via the theory of random walk.

In Chapter V, we give a parallel algorithm on cluster systems to compute all minimal maximum subsequences based on previous structural decomposition with common subsequences. We implement the algorithm with Message Passing Interface (MPI) on the cluster at Oklahoma State University. The empirical study of the speedup and efficiency achieved by the parallel algorithm with synthetic random data is included.

In Chapter VI, we provide the general conclusions and future work.

CHAPTER II

PRELIMINARIES

In this chapter we review the algorithms that find the single minimal maximum sequence and all minimal maximum sequences for the input sequence X.

2.1. Basic Definitions

For the input sequence $X = (x_\eta)_{\eta=1}^{n}$ of n real-valued terms, let $ps_i(X)$ and $ss_i(X)$ denote the prefix sum and suffix sum of X, that is $ps_i(X) = \sum_{\eta=1}^{i} x_\eta$, $ss_i(X) = \sum_{\eta=i}^{n} x_\eta$ for $i \in [1, n]$, and $ps_0(X) = 0$. Also let $sm_i(X)$ denote the suffix maxima of X, i.e., for all $i \in [1, n]$, $sm_i(X) = max\{ps_\eta \mid \eta \in [i, n]\}$. The maximum cumulative sum for the subsequence with starting index i, denote by $m_i(X)$, can be calculated by $m_i(X) = sm_i(X) - ps_i(X) + x_i$, for each $i \in [1, n]$. When the context is clear, we abbreviate the above notations to ps_i, ss_i, sm_i, and m_i for $i \in [1, n]$. Hence the maximum cumulative sum m of the minimal maximum subsequence of X is given by $m = max\{m_\eta \mid \eta \in [1, n]\}$.

For a subsequence Y of X, let $\alpha(Y)$, $\beta(Y)$, and $\gamma(Y)$ denote its starting index, ending index, and index subrange $[\alpha(Y), \beta(Y)]$ in the context of X respectively. If Y is empty, then $\gamma(Y) = \emptyset$. Also let $\gamma_+(Y)$ denote the set of all indices in $\gamma(Y)$ yielding positive terms of Y. In our discussions, the indices are always in the context of X for both its subsequence Y and itself.

2.2. Kadane's Algorithm

Kadane's algorithm sequentially scans the input sequence X from left end to right end to find the minimal maximal subsequence of X. It is described in the following Algorithm 1.

Algorithm 1: Kadane's Algorithm

Input: sequence $X = (x_\eta)_{\eta=1}^{n}$

Output: the maximum cumulative sum m, starting index α and ending index β of the first minimal maximum subsequence with the maximum cumulative sum m.

Begin

1. $m := 0, \alpha := 0, \beta := 0;$
2. $csum := 0, c\alpha := 1;$
3. for $i := 1$ to n
   if $csum + x_i \leq 0$ then
   $csum := 0, c\alpha := i + 1;$
   else
   $csum := csum + x_i;$
   end if
   if $m < csum$ then
   $m := csum, \alpha := c\alpha, \beta := i;$
   end if
   end for

End

Kadane's algorithm follows the divide-and-conquer strategy. If we have found the maximum cumulative sum for the subsequence $(x_\eta)_{\eta=1}^{i-1}$, then how do we find the maximum cumulative sum for the subsequence $(x_\eta)_{\eta=1}^{i}$? The answer is to find the larger one between the maximum cumulative sum for the first $i-1$ terms and the current cumulative sum with ending index i,

which is done by the second if-statement inside the for-loop. The first if-statement is to find the starting index of the next possible minimal maximum subsequence. If the cumulative sum of a subsequence is less than or equal to 0, then it cannot be the prefix of a minimal maximum subsequence. It is easy to see that the run time is linear.

Kadane's algorithm is the first linear algorithm to solve the minimal maximum subsequence that is simple and fast.

2.3. Smith's Algorithm

Smith's recursive algorithm also runs in linear time to solve the minimal maximum subsequence problem.

Algorithm 2: Smith's Algorithm

Input: sequence $X = (x_\eta)_{\eta=1}^{n}$

Output: the maximum cumulative sum m of the minimal maximum subsequence.

Begin

1. $(left, best, sum, right) := MaxSum\ (1, n)$;
2. $m := best$;

End

Function *MaxSum* (i, j)

Input: subsequence $(x_\eta)_{\eta=i}^{j}$

Output: tuple $(left, best, sum, right)$, where $left = max\{\sum_{\eta=i}^{k} x_\eta \,|k \in [i,\ j]\}$, $right = max\{\sum_{\eta=k}^{j} x_\eta \,|k \in [i,\ j]\}$, $sum = \sum_{\eta=i}^{j} x_\eta$, and $best$ is the maximum cumulative sum m of the minimal maximum subsequence of the input.

Begin

1. if $i == j$ then
   return (x_i, x_i, x_i, x_i);
   end if
2. $(left1, best1, sum1, right1) := MaxSum\ (i, (i + j)/2)$;
3. $(left2, best2, sum2, right2) := MaxSum\ ((i + j)/2 + 1, j)$;
4. return $(\max(left1, sum1 + left2), \max(best1, best2, right1 + left2),$
   $sum1 + \text{sum2}, \max(right2, right1 + sum2))$;

End

Smith's algorithm is also based on divide-and-conquer strategy [Smi87]. Kadane's algorithm splits the input sequence X into $(x_\eta)_{\eta=1}^{n-1}$ and x_n, while Smith's algorithm splits X into halves. The maximum cumulative sum m of Smith's algorithm can be searched recursively in the left half X_1, the right half X_2, or the center of X. If the minimal maximum subsequence is in the center of X, then maximum cumulative sum m is the sum of $right1$ (the maximum suffix sum of X_1) and $left2$ (the maximum prefix sum of X_2). It can be proven by contradiction. For example, if the suffix sum of X_1 is not the maximum one, then a larger suffix sum can be found which yields a larger cumulative sum of X than the maximum cumulative sum m, which is impossible.

Bentley created a similar recursive algorithm that searches the maximum cumulative sum among the three maxima in the center, the left half, and the right half respectively [Ben00]. The task to find the cumulative sum in the center will take $O(n)$ time which will in turn search the maximum suffix sum in the left half and the maximum prefix sum in the right half. Therefore the total time of his algorithm is $T(n) = 2T(n/2) + O(n)$, and solution is $T(n) = O(n \log n)$. Smith's algorithm can avoid the linear search of maximum suffix sum and maximum prefix sum by adding the total cumulative sum of the subsequence to the output. For example in Step 4,

$left := \max(left1, sum1 + left2)$, which runs in constant time. The total time of Smith's algorithm is $T(n) = 2T(n/2) + O(1)$, and solution is $T(n) = O(n)$.

2.4. Parallel Algorithm to find Minimal Maximum Subsequence

Akl and Guenther's parallel algorithm [AG91] on the PRAM model solves the single maximum subsequence problem in $O(\log n)$ parallel time using a total of $O(n)$ operations (work-optimal). For the sequence $X = (x_\eta)_{\eta=1}^{n}$, the maximum cumulative sum with starting index i can be calculated by $m_i(X) = sm_i(X) - ps_i(X) + x_i$, for $i \in [1, n]$, and the maximum cumulative sum of X is $m = max\{m_\eta | \eta \in [1, n]\}$. Prefix-sums algorithm can be used to compute the sequence of prefix sum $(ps_\eta)_{\eta=1}^{n}$, suffix maxima $(sm_\eta)_{\eta=1}^{n}$, maximum cumulative sum $(m_\eta)_{\eta=1}^{n}$ with starting index η, and the maximum cumulative sum *m* of *X*. With balanced binary tree method, prefix-sums can be solved in $O(\log n)$ time on exclusive read exclusive write (EREW) PRAM model [JáJ92].

Wen implements Smith's algorithm on EREW PRAM model, and it can also run in $O(\log n)$ time using $\lceil n/\log n \rceil$ processors [Wen95].

2.5. Ruzzo and Tompa's Algorithm

For the problem of finding all minimal maximum subsequences sequentially of $X = (x_\eta)_{\eta=1}^{n}$, the above linear-time sequential Kadane's algorithm can be applied recursively by divide-and-conquer strategy. The pairwise disjointness of all the minimal maximum subsequences suggests that we can first compute a minimal maximum subsequence, remove it, then search recursively in the remaining subsequences. The algorithm has a worst-case time complexity of $\Theta(n^2)$. Empirical analyses of the algorithm [RT99] on synthetic data sets (sequences of independent and

identically distributed uniform random terms with negative mean) and score sequences of genomic data indicate that the running time grows at $\Theta(n \log n)$.

In order to circumvent the iterative dependency in computing the sequence of all minimal maximum subsequences, Ruzzo and Tompa [RT99] prove a structural characterization of the sequence as follows. Denote by $\mathrm{MAX}(X)$ the set of all minimal maximum subsequences or their corresponding index subranges (when the context is clear) of a real-valued sequence X.

Theorem 1 [RT99] *For a non-empty real-valued sequence X, a non-empty subsequence S of X is in* $\mathrm{MAX}(X)$ *if and only if:*

1. [*Monotonicity*] *The subsequence S is monotone: every proper subsequence of S has its cumulative sum less than that of S, and*
2. [*Maximality of Monotonicity*] *The subsequence S is maximal in X with respect to monotonicity, that is, every proper supersequence of S contained in X is not monotone.*

Hence, we also term $\mathrm{MAX}(X)$ as the set of all maximal monotone subsequences of X. This gives a structural decomposition of X into $\mathrm{MAX}(X)$:

1. Every non-empty monotone subsequence of X is contained in a maximal monotone subsequence in $\mathrm{MAX}(X)$; in particular, every positive term of X is contained in a maximal monotone subsequence in $\mathrm{MAX}(X)$, and
2. The set $\mathrm{MAX}(X)$ is a pairwise disjoint collection of all maximal monotone subsequences of X.

Based on the structural characterization of $\mathrm{MAX}(X)$, they presented a sequential algorithm that computes $\mathrm{MAX}(X)$ in $O(n)$ optimal sequential time. The algorithm generalizes the above Kadane's algorithm in a similar inductive and on-line fashion. It scans sequence X from left to

right, and for each successive prefix $P_i = \left(x_\eta\right)_{\eta=1}^{i}$ of X for $i \in [1,\ n-1]$, the algorithm maintains the prefix sum ps_i and a complete list of $k(i)$ pairwise disjoint subsequences of P_i in $\mathrm{MAX}(P_i)$: $S_1,\ S_2,\ \ldots,\ S_{k(i)}$. The sufficient statistics for each $S \in \mathrm{MAX}(P_i)$ are its index subrange $[\alpha(S),\ \beta(S)]$, starting prefix sum $L(S) = ps_{\alpha(S)-1}$ and ending prefix sum $R(S) = ps_{\beta(S)}$. The algorithm will find the next subsequence $S_{k(i)+1}$ which contains the next positive term to the right of x_i. Then $S_{k(i)+1}$ will be integrated with the list in the following process:

1. The list is searched from right to left for the first S_j satisfying $L(S_j) < L(S_{k(i)+1})$.
2. If there is no such S_j, then add $S_{k(i)+1}$ to the end of the list.
3. If there is such a S_j, and $R(S_j) \geq R(S_{k(i)+1})$, then add $S_{k(i)+1}$ to the end of the list.
4. Otherwise (i.e., there is such a S_j, but $R(S_j) < R(S_{k(i)+1})$), let $\beta(S_j) = \beta(S_{k(i)+1})$, and remove $S_{j+1}, \ldots, S_{k(i)}$ from the list (none of them is maximal) and reconsider the newly extended subsequence S_j as in Step 1.

For example, if input sequence $X = (-1, 4, -3, -1, 5, -4, 2, -2, 5, -2, 1)$, and after read $(-1, 4, -3, -1, 5, -4, 2, -2)$, the list of disjoint subsequences is $S_1 = (4)$, $S_2 = (5)$, and $S_3 = (2)$. The next subsequence is $S_4 = (5)$. The cumulative sums of the prefixes are plotted in following Figure 2.1. Since $L(S_2) < L(S_4)$ and $R(S_2) < R(S_4)$, according to Step 4, S_2 will be augmented to include S_4, then list will become to $S_1 = (4)$ and $S_2 = (5, -4, 2, -2, 5)$.

A correct implementation of Steps 2 and 3 is needed to achieve the linear run time. In Step 2, if no S_j exists, then all subsequences $S_1,\ S_2,\ \ldots,\ S_{k(i)}$ are in $\mathrm{MAX}(X)$, and they can be removed from the list. Also if Step 3 appended a subsequence $S_{k(i)+1}$ to the end of list, then the discovered S_j should be preserved, thus we can avoid the redundant searching in Step 1 of the next iteration.

For example, a list of left smaller match of the starting prefix sum can be created for all the subsequences with the nearest smaller value algorithm [HH01].

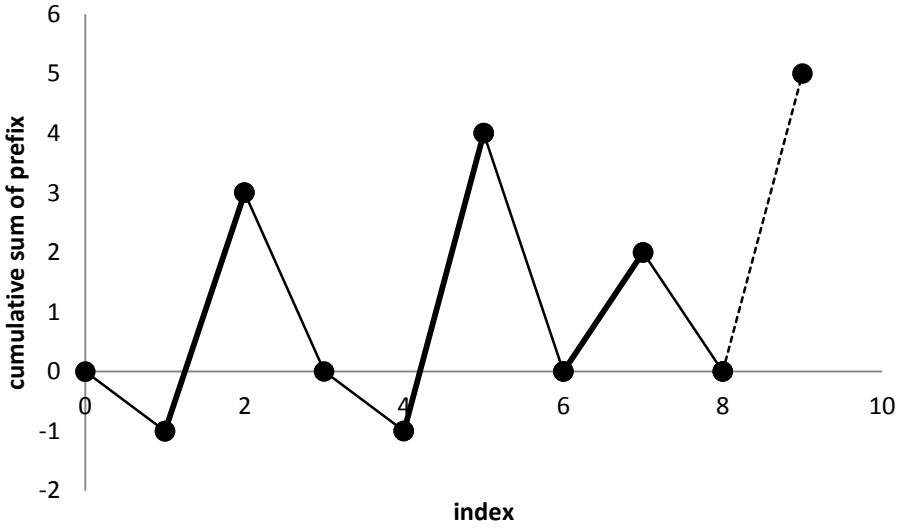

Figure 2.1: Example of Ruzzo and Tampa's algorithm. Bold lines are the subsequences in the current list, and the dotted line is the subsequence to be processed.

In the next chapter, we introduce other structural decompositions of a sequence X that lead to computing MAX(X) with: (1) a parallel algorithm on the PRAM model [DS06] in logarithmic parallel time and optimal linear work, and (2) a domain-decomposed parallel algorithm implemented on cluster systems with MPI.

CHAPTER III

STRUCTURAL DECOMPOSITIONS OF A SEQUENCE

3.1 Characterization of Monotonicity

Theorem 1 in Chapter II shows that every minimal maximal subsequence S of X is monotone: every proper subsequence of S has less cumulative sum than that of S, and S is a maximal monotone subsequence of X. The algorithm to find the minimal maximal subsequences is equivalent to find those maximal monotone subsequences of X. The following characterization of monotonicity [DS06] yields an effective computation of the index subrange of a non-trivial monotone subsequence containing a given term of X.

Lemma 2 *Let X be a non-empty real-valued sequence and Y be a non-empty subsequence of X (with index subrange $[\alpha(Y), \beta(Y)]$). The following statements are equivalent:*

1. *Y is monotone in X.*
2. *The starting prefix sum $ps_{\alpha(Y)-1}(X)$ of Y is the unique minimum and the ending prefix sum $ps_{\beta(Y)}(X)$ of Y is the unique maximum of all $ps_i(X)$ for all $i \in [\alpha(Y), \beta(Y)]$.*
3. *All non-empty prefixes and non-empty suffixes of Y have positive cumulative sums.*

The monotonicity constraint applies only to the starting and ending indices of a monotone subsequence, but not to the entire index subrange. The monotone subsequence can contain zero or negative terms, but those terms cannot be the either end of the monotone subsequence. For the subsequence $X = (-1, 4, -3, -1, 5, -4, 2, -2, 5, -2, 1)$ in Figure 2.1, the family of all monotone subsequences are (4), (5), (2), (5), $(5, -4, 2, -2, 5)$, and (1).

3.2 Maximal Monotone Subsequence with Starting Positive Term

The parallel algorithm of Alk and Guenther [AG91] to find a single maximum subsequence focuses on computing the maximum cumulative sum concurrently with each possible starting index, but it disregards the monotonicity condition of all such subsequences. More work is required in order to reveal the structural decomposition of X into $\mathrm{MAX}(X)$.

The key to the parallel implementation of finding $\mathrm{MAX}(X)$ for sequence $X = \left(x_\eta\right)_{\eta=1}^{n}$ lies in the concurrent computation of the ending index of the maximal monotone subsequence constrained with the starting index $i \in [1,\ n]$. According to Lemma 2, we only need to consider the positive terms of X for the desired computation.

Let $\epsilon : \gamma_+(X) \to \gamma(X)$ be the function such that $\epsilon(i)$ denotes the ending index of the maximal monotone subsequence of X constrained with the starting index i. The function ϵ is composed of the following two functions:

1. The function $\epsilon' : \gamma_+(X) \to [2, n+1]$ defined by:

$$\epsilon' = \begin{cases} \min\{\, \eta \in [i+1, n] \,|\, ps_{i-1} \geq ps_\eta \} & \text{if the minimum exists,} \\ n+1 & \text{otherwise,} \end{cases}$$

   locates the least index $\eta \in [i+1, n]$ such that $ps_{i-1} \geq ps_\eta$ if it exists.

2. The function $\epsilon'' : \{\eta \in [1, n]^2 |\ i \leq j\} \to [1, n]$ defined by:

$$\epsilon''(i,j) \;=\; \min \operatorname{argmax}\{ps_\eta \mid \eta \in [i,j]\}\,,$$

locates the least index $\eta \in [i,j]$ such that ps_η is the maximum prefix sum of those of X over the index subrange $[i,j]$.

The following lemma shows that the function ϵ is the composition of ϵ' *and* ϵ''.

Lemma 3 *For the functions* ϵ, ϵ', *and* ϵ'' *defined above,* $\epsilon(i) = \epsilon''(i, \epsilon'(i) - 1)$ *for all* $i \in [1,n]$.

According to the definition, $\epsilon''(i, \epsilon'(i) - 1)$ is the index of the unique maximum prefix sum on $[i, \epsilon'(i) - 1]$. At the index $\epsilon'(i)$, prefix sum decreased, so it cannot be $\epsilon(i)$. Assume $\epsilon(i)$ is larger than $\epsilon'(i)$, then the cumulative sum on index subrange $[i, \epsilon(i)]$ must be larger than the cumulative sum on $[\epsilon'(i) + 1, \epsilon(i)]$, i.e., $ps_{\epsilon(i)} - ps_{i-1} > ps_{\epsilon(i)} - ps_{\epsilon'(i)}$. This is impossible according to the definition of the function ϵ', so $\epsilon(i)$ must be equal to $\epsilon''(i, \epsilon'(i) - 1)$.

For the above input subsequence $X = (-1, 4, -3, -1, 5, -4,\ 2, -2,\ 5, -2,\ 1)$ in Figure 2.1, $\epsilon(2) = \epsilon''(2,\ \epsilon'(2) - 1) = \epsilon''(2,\ 3) = 2$, $\epsilon(5) = \epsilon''(5,\ 11) = 9$, $\epsilon(7) = \epsilon''(7,\ 7) = 7$, $\epsilon(9) = \epsilon''(9,\ 11) = 9$, and $\epsilon(11) = \epsilon''(11,\ 11) = 11$.

The concurrent computation of ϵ for all the positive terms x_η in X, generates the statistics $\mathrm{MON}(X) = \{[i, \epsilon(i)] \mid i \in \gamma_+(X)\}$ which is the set of all index subranges of all maximal monotone subsequences of X constrained with given positive starting terms. The following theorem [DS06] reveals the structural decomposition of X into $\mathrm{MON}(X)$, which refines $\mathrm{MAX}(X)$ and provides a basis for a parallel computation of $\mathrm{MAX}(X)$ from $\mathrm{MON}(X)$.

Theorem 4 *For a real-valued sequence* X, $\mathrm{MON}(X)$ *satisfies the following parenthesis structure:*

1. *Every positive term of X has its index as the starting index of a unique index subrange in* $MON(X)$,

2. *For every pair of index subranges in* $MON(X)$, *either they are disjoint or one is a subrange of the other, and*

3. *For every maximal monotone subsequence of X in* $MAX(X)$, *its index subrange is in* $MON(X)$.

For the above example, there are three minimal maximal subsequences: (4), $(5, -4,\ 2, -2,\ 5)$, and (1), and their index subranges are $[2, \epsilon(2)] = [2,\ 2]$, $[5, \epsilon(5)] = [5,\ 9]$, and $[11, \epsilon(11)] = [11,\ 11]$ respectively. Index subranges $[7, \epsilon(7)] = [7,\ 7]$ and $[9, \epsilon(9)] = [9,\ 9]$ are also in $\mathrm{MON}(X)$, but they are not the index subranges of the maximal subsequences in $\mathrm{MAX}(X)$, instead they are contained by $[5,\ 9]$.

3.3 PRAM Algorithm to Compute MAX(X)

For two index subranges $[i_1, j_1]$ and $[i_2, j_2]$, if they are either disjoint or one is contained in the other, then we can define the range-composition function $\circ$ to select the rightmost or outmost index subrange from them:

$$[i_1, j_1] \circ [i_2, j_2] = \begin{cases} [i_2, j_2] & \text{if } j_1 < i_2\ ([i_1, j_1] \cap [i_2, j_2] = \emptyset) \\ [i_1, j_1] & \text{if } j_2 < i_1\ ([i_1, j_1] \cap [i_2, j_2] = \emptyset) \\ [i_2, j_2] & \text{if } [i_1, j_1] \subseteq [i_2, j_2] \\ [i_1, j_1] & \text{if } [i_1, j_1] \supseteq [i_2, j_2] \end{cases}$$

Dai and Su designed the following PRAM algorithm to compute $\mathrm{MON}(X)$, and compute $\mathrm{MAX}(X)$ from $\mathrm{MON}(X)$ using the range-composition function $\circ$.

Algorithm 3: Compute_MAX

Input: Input sequence X in an array of n real values $x[1..n]$.

Output: The sequence of all minimal maximum subsequences (that is, all maximal monotone subsequences) of X occupying the low-order subarray of an array $M[1..\lceil n/2 \rceil]$

Begin

1. Compute the prefix sums of X in an array $s[1..n]$ such that $s[i] = \sum_{\eta=1}^{i} x[\eta]$ for all $i \in [1, n]$;

2. Compute the function ϵ in an array $\epsilon[1..n]$ such that $\epsilon[i]$ denotes the ending index of the maximal monotone subsequence of X constrained with the starting index i as follows:

   2.1. Compute the function ϵ' in an array $\epsilon'[1..n]$, in which $\epsilon'[i]$ is the least index $\eta \in [i+1, n]$ such that $s[i-1] \geq s[\eta]$ if it exists, and $n+1$ otherwise;

   2.2. Compute the function ϵ'' in an array $\epsilon''[1..n, 1..n]$, in which $\epsilon''[i, j]$ is the least index $\eta \in [i, j]$ such that $s[\eta] = \max\{s[k] \mid k \in [i, j]\}$;

   2.3. Compose ϵ from ϵ' and ϵ'' as follows:

   for all $i \in [1, n]$ in parallel do

   $\epsilon[i] = \epsilon''[i, \epsilon'[i] - 1]$;

   end for;

3. Compute $\text{MON}(X) = \{[i, \epsilon[i]] \mid i \in [1, n] \text{ with } x[i] > 0\}$ (ordered according to the starting index) and pack all the index subranges of MON(X) in the low-order subarray of the array $[1..\lceil n/2 \rceil]$ ($|\text{MAX}(X)| \leq \lceil n/2 \rceil$).

4. Compute MAX(X) in the array $M[1..\lceil n/2 \rceil]$ as follows:

   4.1. Compute the prefix sums of the (non-trivial) low-order subarray of $M[1..\lceil n/2 \rceil]$ using the range-composition function $\circ$ for the prefix computation;

4.2. Pack all the distinct elements (pairwise disjoint index subranges) in the (non-trivial) low-order subarray of $M[1..\lceil n/2 \rceil]$ in place, while maintaining their relative order (according to the starting index);

End

Step 1 is implemented by the prefix-sums algorithm [LF80] that runs in $O(\log n)$ time, using $O(n)$ operations on the EREW PRAM.

Step 2.1, the computation of ϵ', is reduced to the problem of all nearest smaller values of the sequence, which can be solved by an algorithm ([BBG+89], [Che95]) that runs in $O(\log n)$ time, using $O(n)$ operations on the EREW PRAM.

Step 2.2, the computation of ϵ'', is reduced to the problem of range-minima, which can be solved by an algorithm [JáJ92] that runs in $O(\log n)$ time, using $O(n)$ operations on the EREW PRAM. Step 2.3, the computation of ϵ, is executed in $O(1)$ time, using $O(n)$ operations on the EREW PRAM.

Step 3 is reduced to the problem of array packing, which can be solved by the prefix-sums algorithm.

As for Step 4, Step 4.1 is a direct application of the prefix-sums algorithm, and Step 4.2 is reduced to array packing as in Step 3.

Therefore for a length-n real-valued sequence X, the algorithm Compute_MAX computes the set MAX(X) of all minimal maximum subsequences (that is, all maximal monotone subsequences) of X in $O(\log n)$ time using $O(n)$ operations (work-optimal) on the EREW PRAM model.

3.4 Domain Decomposition of X

One of our initial research tasks was to adapt the logarithmic-time optimal-work parallel algorithm on practical parallel systems. However, in view of the efficient linear-time sequential algorithm [RT99], we devise and implement a domain-decomposed parallel algorithm computing $\mathrm{MAX}(X)$ that employs the optimal sequential algorithm in subsequence-hosting processors.

An ideal domain decomposition of a sequence X is a partition of X into a pairwise disjoint family $\mathcal{X}$ of non-empty subsequences of X that are length-balanced and MAX-independent: $\mathrm{MAX}(X) = \bigcup_{Y \in \mathcal{X}} \mathrm{MAX}(Y)$ (Y as a sequence in its own right). We first find a sufficient condition for the MAX-independence that $\mathrm{MAX}(Y)$ can be computed locally in subsequence-hosting processors.

Lemma 3 shows that the ending index of the minimal maximum subsequence constrained with the starting index i can be found in $[i+1,\ \epsilon'(i)-1]$, where $ps_{i-1} \geq ps_{\epsilon'(i)}$. The function ϵ' is equivalent to the following function $\mathrm{rm}_X : \gamma_+(X) \to [\alpha(X)+1, \beta(X)] \cup \{\beta(X)+1\}\ (= [2, n+1])$ that is the nearest-smaller-or-equal right-match of ps_{i-1} of X :

$$\mathrm{rm}_X(i) = \begin{cases} \min\{\eta \in [i+1, \beta(X)] \mid ps_{i-1} \geq ps_\eta\} & \text{if the minimum exists,} \\ \beta(X)+1\ (= n+1) & \text{otherwise.} \end{cases}$$

Similarly let function $\mathrm{lm}_X : \gamma_+(X) \to [\alpha(X), \beta(X)-1] \cup \{\alpha(X)-1\}\ (= [0, n-1])$ denote the nearest-smaller left-match of ps_{i-1} of X :

$$\mathrm{lm}_X(i) = \begin{cases} \max\{\eta \in [\alpha(X), i-1] \mid ps_{i-1} > ps_{\eta-1}\} & \text{if the maxmum exists,} \\ \alpha(X)-1\ (= 0) & \text{otherwise.} \end{cases}$$

For the above input subsequence $X = (-1, 4, -3, -1, 5, -4,\ 2, -2,\ 5, -2,\ 1)$, Figure 3.1 and Figure 3.2 illustrate the computation of the two functions lm_X and rm_X.

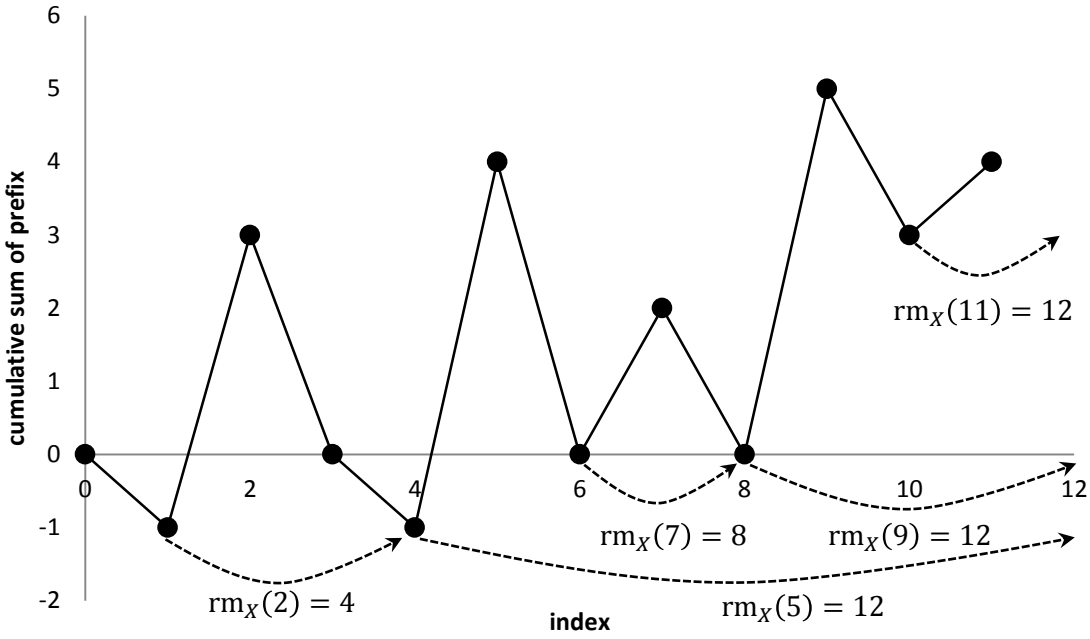

Figure 3.1: Example of $\mathrm{rm}_X(i)$. Dotted lines connected ps_{i-1} and $ps_{rm_X(i)}$ if $\mathrm{rm}_X(i)$ exists.

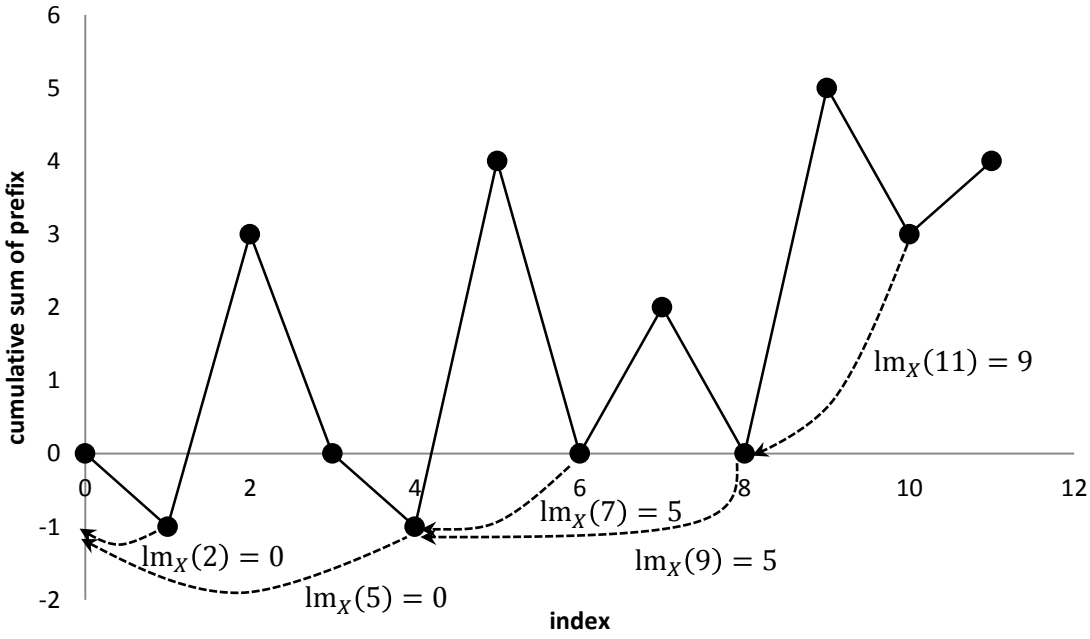

Figure 3.2: Example of $\mathrm{lm}_X(i)$. Dotted lines connected ps_{i-1} and $ps_{\mathrm{lm}_X(i)-1}$ if $\mathrm{lm}_X(i)$ exists.

From the definitions of $\mathrm{lm_X}$ and $\mathrm{rm_X}$, we note that both $\{[\,\mathrm{lm_X}(i), i] \mid i \in \gamma_+(X)\}$ and $\{[\,i, \mathrm{rm_X}(i)] \mid i \in \gamma_+(X)\}$ satisfy the parenthesis structure similar to that of MON — but permitting abutting index subranges (at subrange ends). The functions $\mathrm{lm_X}$ and $\mathrm{rm_X}$ help locate the starting index and ending index respectively of a maximal monotone subsequence of X containing the positive term x_i. In Ruzzo and Tompa's algorithm, $\mathrm{lm_X}$ is used to determine if multiple maximal monotone subsequences covering the index subrange $[\mathrm{lm}_X(i),\ i]$ should be merged. In Compute_MAX algorithm, rm_X is used to search the ending index of a maximal monotone subsequence with given starting index i in the index subrange $[i, \mathrm{rm}_X(i) - 1]$.

Consider a sequential partition of a sequence X into m consecutive subsequences $X_1, X_2, \ldots, X_m$ with which $\alpha_i^+ \in \gamma(X_i)$ denotes the index of the first positive term of X_i (if exists) for $i \in [1, m]$. Our goal is to find a sequential partition which is MAX-independent such that for each subsequence X_i, its hosting processor can find $\mathrm{MAX}(\mathrm{X}_i)$ independently, and $\mathrm{MAX}(X)$ is the union of all $\mathrm{MAX}(\mathrm{X}_i)$. If such a sequential partition exists, then subsequences $X_1, X_2, \ldots, X_m$ can be hosted by m processors and each processor will run optimal sequential algorithm to find $\mathrm{MAX}(\mathrm{X}_i)$ independently. If the sequential partition that is MAX-independent, then the communication time among the processors can be minimized.

When running Compute_MAX algorithm sequentially on a single processor, we can obtain the partition $P_0(X) = (X_1, X_2, \ldots, X_m)$, where the index subrange of X_1 is $\left[\alpha(X), \mathrm{rm}_{X_1}(\alpha_1^+)\right]$, and index subrange of X_i is $\left[\mathrm{rm}_{X_{i-1}}(\alpha_{i-1}^+) + 1, \mathrm{rm}_{X_i}(\alpha_i^+)\right]$ for $i \in [2, m]$. Notice that $\mathrm{MAX}(X_1) \subseteq \mathrm{MAX}(X)$, because for any $j \in \gamma_+(X_1)$, the cumulative sum of $\left(x_\eta\right)_{\eta=j}^{\mathrm{rm}_{X_1}(\alpha_1^+)} = ps_{\mathrm{rm}_{X_1}(\alpha_1^+)} - ps_{j-1} \leq ps_{\mathrm{rm}_{X_1}(\alpha_1^+)} - ps_{\alpha_1^+ - 1} \leq 0$, so the maximal monotone subsequences of X_1 must be maximal monotone in X according to Theorem 1. Because of the pairwise disjoint property of $\mathrm{MAX}(X)$, after excluding $\mathrm{MAX}(X_{i-1})$ from $\mathrm{MAX}(X)$, we can recursively obtain that $\mathrm{MAX}(X_i) \subseteq$

$\mathrm{MAX}(X)$ for $i \in [2, m]$. On the other hand, for any maximal monotone subsequence in $\mathrm{MAX}(X)$, we can find a maximal monotone subsequence with the same starting index in one of the X_i, and they must have the same ending index using the same reasoning above. Therefore the partition $P_0(X)$ is MAX-independent. From partition $P_0(X)$ we can also obtain other partitions by concatenating multiple consecutive subsequences of $P_0(X)$ into a longer subsequence. Generally we have the following two lemmas.

Lemma 5 *Let* $\left(X_\eta\right)_{\eta=1}^{m}$ *be a sequential partition of a real-valued sequence X with* X_η*, for* $\eta \in [1, m]$*, represented as a sequence in its own right over its index range* $\gamma\left(X_\eta\right)$*. If the partition satisfies the* rm-*closure condition: for all* $i \in [1, m-1]$ *and all* $j \in \gamma_+(X_i)$, $\mathrm{rm}_{X_i}(j) \in [j+1, \beta(X_i)]$*, then the partition is* MAX-*independent*: $\mathrm{MAX}(X) = \cup_{\eta=1}^{m} \mathrm{MAX}\left(X_\eta\right)$.

Proof. Let $Y \in \mathrm{MAX}(X_i)$ for some $i \in [1, m]$ be arbitrary, so Y is monotone (in X_i and X). Consider a supersequence Z of Y with $Z \in \mathrm{MAX}(X)$. We show that $\gamma(Z) \subseteq \gamma(X_i)$, and the maximality of the monotonicity of Y in X_i and the monotonicity of Z give that $Y = Z \in \mathrm{MAX}(X)$.

Let $\alpha(Z) \in \gamma(X_j)$, for some $j \in \left[1, i\right]$. If $j\ (= i) = m$, we have $\gamma(Y) \subseteq \gamma(Z) (\subseteq \gamma(X_m))$ as desired in subsequence X_m. Otherwise, if $j \leq m-1$, note that $\alpha(Z) \in \gamma_+(X_j)$ and the assumption of rm_{X_j} gives the instance for $\alpha(Z)$: $\mathrm{rm}_{X_j}(\alpha(Z)) \in \left[\alpha(Z)+1, \beta\left(X_j\right)\right]$, which says that for all $\eta \in \left[\alpha(Z)+1, \mathrm{rm}_{X_j}\left(\alpha(\mathrm{Z})\right)-1\right]$,

$$ps_{\alpha(Z)-1} < ps_\eta, \text{ but } ps_{\alpha(Z)-1} \geq ps_{\mathrm{rm}_{X_j}(\alpha(Z))}.$$

Because $\mathrm{rm}_{X_j}(\alpha(Z)) \leq \beta\left(X_j\right)$, nearest-smaller right-match of $ps_{\alpha(Z)-1}$ exists, so in the context of rm_X the above inequalities are equivalent to:

$$ps_{\alpha(Z)-1} < ps_\eta, \text{ but } ps_{\alpha(Z)-1} \geq ps_{rm_X(\alpha(Z))};$$

that is, $\mathrm{rm}_X\big(\alpha(Z)\big) = \mathrm{rm}_{X_j}\big(\alpha(Z)\big) \in \big[\alpha(Z)+1, \beta\big(X_j\big)\big]$ as expected. Now applying Lemma 2 to the monotone subsequence Z in X that $ps_{\alpha(Z)-1}$ is the unique minimum of all ps_η for all $\eta \in [\alpha(Z)-1, \beta(Z)]$, we must have $\mathrm{rm}_X\big(\alpha(Z)\big) > \beta(Z)$, i.e., $\beta(Z) \in \big[\alpha(Z), \mathrm{rm}_X\big(\alpha(Z)\big)-1\big] \subseteq \gamma(X_j)$. Since $\gamma(Y) \subseteq \gamma(Z)$, $\beta(Z) \geq \beta(Y) \in \gamma(X_i)$, we have $j = i$ and $\gamma(Z) \subseteq \gamma\big(X_j\big) = \gamma(X_i)$ as desired.

To see the reverse containment, let $Y \in \mathrm{MAX}(X)$ be arbitrary with $\alpha(Y) \subseteq \gamma(X_i)$ for some $i \in [1, m]$. Applying an analogous argument using the assumption of rm_{X_i} as above (for the maximal monotone subsequence Z of X that $\gamma(Z) \subseteq \gamma(X_i)$ to Y results that $\gamma(Y) \subseteq \gamma(X_i)$ and $Y \in \mathrm{MAX}(X_i)$. ■

Lemma 6 *For a non-empty subsequence Y of real-valued sequence X, the right-match function $rm_Y: \gamma_+(Y) \to [\alpha(Y)+1, \beta(Y)] \cup \{\beta(Y)+1\}$, satisfies the rm-closure condition stated in Lemma 5 (for all $j \in \gamma_+(Y)$, $rm_Y(j) \in [j+1, \beta(Y)]$,) if and only if the sequence Y satisfies the minimum prefix-sum condition: the ending prefix sum of Y, $ps_{\beta(Y)}$, is a global minimum of all ps_i for all $i \in [\alpha(Y)-1, \beta(Y)]$.*

Proof. For the "only if"-part, assume that Y satisfies the rm-closure condition. If $\gamma_+(Y) = \emptyset$, i.e., all the terms Y are non-positive, then clearly $ps_{\alpha(Y)-1} \geq ps_{\alpha(Y)} \geq \ldots \geq ps_{\beta(Y)}$. Otherwise, select the least index $i_1 \in \gamma_+(Y)$, and note that $ps_{\alpha(Y)-1} \geq ps_{\alpha(Y)} \geq \ldots \geq ps_{i_1-1}$. The rm-closure condition, when applying to the index i_1, yields that $\mathrm{rm}_Y(i_1) \in [i_1+1, \beta(Y)]$ with $ps_{\mathrm{rm}_Y(i_1)} = \min\{ps_\eta \mid \eta \in [i_1-1, \mathrm{rm}_Y(i_1)]\}$. Then select the least index $i_2 \in \gamma_+(Y)$ such that $i_2 > i_1$, and apply the rm-closure condition to it. Continuing this process results in the following lower-envelope sequence of increasing indices:

$$\alpha(Y) \leq i_1 < \mathrm{rm}_Y(i_1) < i_2 < \mathrm{rm}_Y(i_2) < \ldots < i_j < \mathrm{rm}_Y\big(i_j\big) \leq \beta(Y)$$

for some positive integer j such that: (1) $i_1, i_2, \ldots, i_j \in \gamma_+(Y)$ where i_1 is the least indices in $\gamma_+(Y)$, and for any $k \in \gamma_+(Y)$, $k \in \cup_{\eta=1}^{j}[i_\eta, \mathrm{rm}_Y(i_\eta) - 1]$, (2) the prefix sum ps_η is a decreasing function of η over $[\alpha(Y) - 1, i_1 - 1] \cup (\cup_{k=1}^{j-1}[\mathrm{rm}_Y(i_k), i_{k+1} - 1]) \cup [\mathrm{rm}_Y(i_j), \beta(Y)]$, and (3) for all $\eta \in [1, j]$, $ps_{\mathrm{rm}_Y(i_\eta)} = \min\{ps_k \mid k \in [i_\eta - 1, \mathrm{rm}_Y(i_\eta)]\}$. Combining (2) and (3) above, we have $ps_{\beta(Y)} = \min\{ps_\eta \mid \eta \in [\alpha(Y) - 1, \beta(Y)]\}$.

For the "if"-part, assume that the sequence Y satisfies the minimum prefix-sum condition: $ps_{\beta(Y)} = \min\{ps_k \mid k \in [\alpha(Y) - 1, \beta(Y)]\}$. For every $j \in \gamma_+(Y)$, $ps_{j-1} \geq ps_{\mathrm{rm}_Y(j)} \geq ps_{\beta(Y)}$, we have $\mathrm{rm}_Y(j) \in [j + 1, \beta(Y)]$. ■

Figure 3.3 illustrates a partition satisfying rm-closure condition.

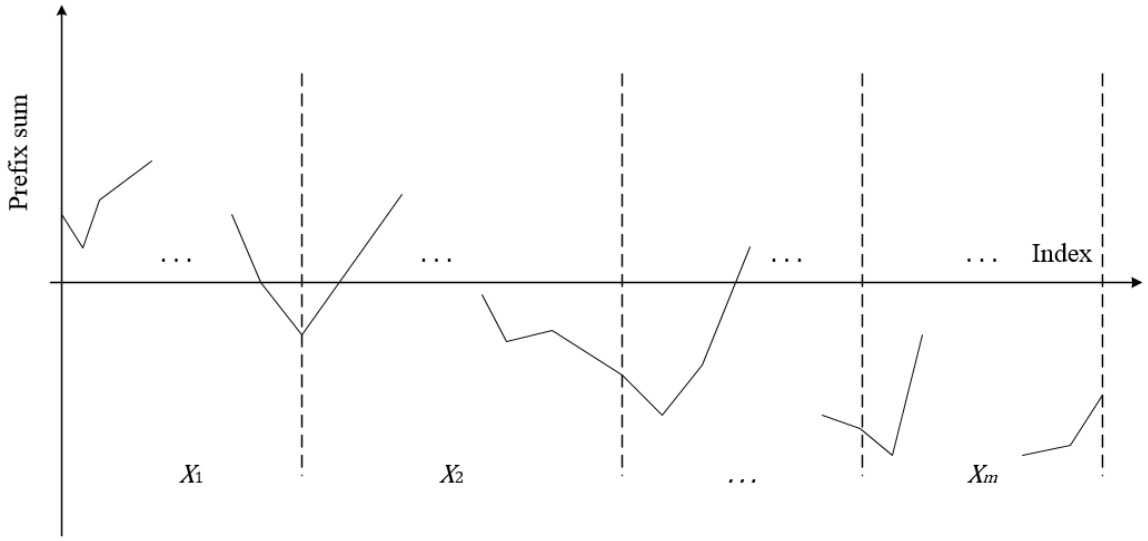

Figure 3.3: Partition satisfies rm-closure condition. For all $i \in [1, m - 1]$ and all $j \in \gamma_+(X_i)$, $rm_{X_i}(j) \in [j + 1, \beta(X_i)]$.

The minimum prefix-sum condition, equivalent to the rm-closure condition as shown in Lemma 6, exposes a stringent sufficiency for MAX-independence of a priori sequential partition of a sequence X: for all $i \in [1, m - 1]$, the ending prefix sum $ps_{\beta(X_i)}$ is a global minimum of all

prefix sums ps_η of X_i for $\eta \in [\alpha(X_i) - 1, \beta(X_i)]$. For a sequential partition, $\beta(X_i) = \alpha(X_{i+1}) - 1$, therefore $ps_{\beta(X_1)} \geq ps_{\beta(X_2)} \geq \ldots \geq ps_{\beta(X_{m-1})}$ is required if the partition satisfies rm-closure condition. We incorporate the minimum prefix-sum condition into constructing a posteriori sequential partition of X that forms the basis in designing a domain-decomposed parallel algorithm in computing MAX(X).

3.5 Domain Decomposition of X with Common Subsequences

The rm-closure partition $(X_1, X_2, \ldots, X_m)$ of input sequence X allows each partition X_i for $i \in [1, m]$ to calculate $\text{MAX}(X_i) \subseteq \text{MAX}(X)$ independently, but the partition is not necessarily length-balanced. On the other hand, if the sequence X is partitioned onto m processors with equal lengths, the subsequence X_i on i^{th} processor for $i \in [1, m]$ may not satisfy minimum prefix-sum condition. To create a length-balanced rm-closure partition is crucial for our parallel algorithm.

If the terms of input sequence X are random variables, we have pointed out that the expected value of a single term x_i of sequence X is negative in some applications of the minimal maximal subsequence problem [KA90], [KA93] in Chapter I. If the expected value of x_i is positive, then the minimal maximal subsequence would likely contain most terms of sequence X, which may not provide useful information to the applications. If the expected value of x_i is negative, then the prefix sums of sequence X tend to be decreasing from the starting index to the ending index of X. For a subsequence Y of sequence X with negative expected value of x_i, if the ending prefix sum $ps_{\beta(Y)}$ is not the minimal prefix sum ps_η $(\eta < \beta(Y))$ for all prefix sums of Y, then it is likely to find a prefix sum $ps_j \leq ps_\eta$ with $j > \beta(Y)$ in the supersequence of Y. The partition we are going to introduce is based on the intuition that the supersequence of Y has larger probability to satisfy rm-closure condition if the prefix sums tend to decrease.

For two sequences X and Y, denote the concatenation of X and Y by the juxtaposition XY. Let X be a nonempty real-valued sequence with a sequential partition:

$$\mathcal{P}(X) = \left(X_1, X_{1,2}, X_2, X_{2,3}, X_3, \ldots, X_{m-1}, X_{m-1,m}, X_m\right).$$

For notational simplicity, let $X_{0,1} = \emptyset$, and $X_{m,m+1} = \emptyset$.

For every $i \in [1, m-1]$, denote by β_i^* the maximum or right-most index $\eta \in \gamma_+\left(X_{i-1,i}X_i\right)$, if non-empty, such that $ps_{\eta-1}$ is the minimum prefix sum of those of $X_{i-1,i}X_i$ over $\gamma_+\left(X_{i-1,i}X_i\right)$, that is,

$$\beta_i^* = \max \operatorname{argmin}\left\{ps_{\eta-1} \mid \eta \in \gamma_+\left(X_{i-1,i}X_i\right), \gamma_+\left(X_{i-1,i}X_i\right) \neq \emptyset\right\}.$$

The partition $\mathcal{P}(X)$ satisfies the rm-locality condition if for every $i \in [1, m-1]$ with non-empty $\gamma_+\left(X_{i-1,i}X_i\right)$, $\mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*) \in \left[\beta_i^* + 1, \beta\left(X_{i-1,i}X_iX_{i,i+1}\right)\right]$. For every $i \in [1, m-1]$, and for every $j \in \gamma_+\left(X_{i-1,i}X_i\right)$, $\mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(j) \in \left[j+1, \beta\left(X_{i-1,i}X_iX_{i,i+1}\right)\right]$, and according to the definition of rm_X, for every positive term x_η, $\eta \in \left[\beta\left(X_{i-1,i}X_i\right) + 1, \mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*)\right]$, $\mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\eta) \leq \mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*)$. Thus the concatenation of $X_{i-1,i}X_i$ and the prefix of $X_{i,i+1}$ with ending index $\mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*)$ satisfies rm-closure condition by Lemma 5.

From the rm-localized sequential partition $\mathcal{P}(X)$, we derive a MAX-independent sequential partition $\tilde{\mathcal{P}}(X) = \left(X''_{i-1,i}X_iX'_{i,i+1}\right)_{i=1}^{m}$ where $X''_{i-1,i}$ and $X'_{i,i+1}$ are respectively the suffix of $X_{i-1,i}$ and prefix of $X_{i,i+1}$ that are determined by rm-computation as follows. Recall that $X_{0,1} = \emptyset$ and $X_{m,m+1} = \emptyset$ for notational simplicity, let $X''_{0,1} = \emptyset$ and $X'_{m,m+1} = \emptyset$ accordingly. For every $i \in [1, m-1]$, define:

$$X'_{i,i+1} = \begin{cases} \emptyset & \text{if } \gamma_+(X_{i-1,i}X_i) = \emptyset, \\ \emptyset & \text{if } \gamma_+(X_{i-1,i}X_i) \neq \emptyset, \text{ and} \\ & \mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*) \in [\beta_i^* + 1, \beta(X_{i-1,i}X_i)], \\ [\alpha(X_{i,i+1}), \mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*)] & \text{otherwise (i.e., } \gamma_+(X_{i-1,i}X_i) \neq \emptyset, \text{ and} \\ & \mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*) \in \gamma(X_{i,i+1}), \end{cases}$$

and $X''_{i,i+1}$ to be the (remaining) suffix of $X_{i,i+1}$ such that $X'_{i,i+1}X''_{i,i+1} = X_{i,i+1}$. Observe that the first two cases in defining $X'_{i,i+1}$ may be absorbed into the third case. Figure 3.4 gives an example partition of $X_{i,i+1}$.

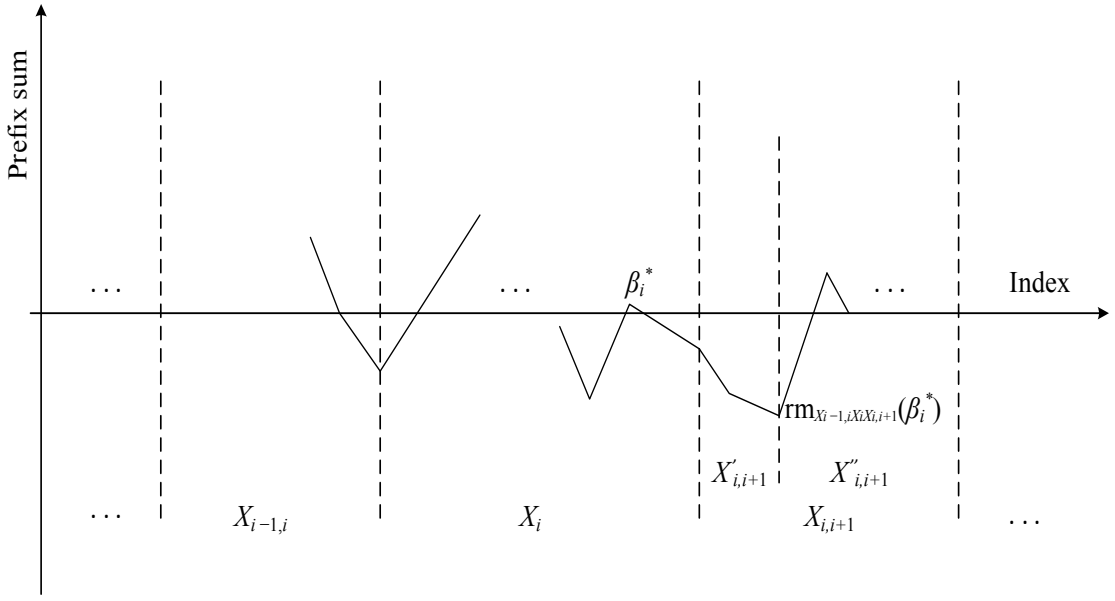

Figure 3.4: Partitioning $X_{i,i+1}$ into $X'_{i,i+1}X''_{i,i+1}$.

The definition of $\beta_i^* = \max \mathrm{argmin}\{ps_{\eta-1} \mid \eta \in \gamma_+(X_{i-1,i}X_i), \gamma_+(X_{i-1,i}X_i) \neq \emptyset\}$ is to search β_i^* over $\gamma_+(X_{i-1,i}X_i)$ for $i \in [1, m-1]$, and the calculation of $\mathrm{rm}_{X_{i-,i}X_iX_{i,i+1}}(\beta_i^*)$ is over $[\beta_i^* + 1, \beta(X_{i-1,i}X_iX_{i,i+1})]$. This creates a domain-decomposition of input X that X will be separated onto m processors with the i^{th} processor hosting the subsequence $X_{i-1,i}X_iX_{i,i+1}$ for $i \in [1, m-1]$. The common subsequence $X_{i,i+1}$ is hosted on successive i^{th} and $(i+1)^{\text{th}}$ processors.

However the subsequence $X''_{i-1,i}X_iX'_{i,i+1}$ of the sequential partition $\tilde{\mathcal{P}}(X)$ only requires the suffix $X''_{i-1,i}$ of $X_{i-1,i}$ for $i \in [2,m]$. Therefore one might have constructed an rm-localized MAX-independent sequential partition inductively on i as follows:

For $i = 1$, let $\beta_1^* = \max \operatorname{argmin}\{ps_{\eta-1} \mid \eta \in \gamma_+(X_1), \gamma_+(X_1) \neq \emptyset\}$, and define the prefix $X'_{1,2}$ and suffix $X''_{1,2}$ of $X_{1,2}$ similar to the former version based on the calculation of $\mathrm{rm}_{X_1X_{1,2}}(\beta_1^*)$. For $i = 2, 3, \ldots, m-1$, $\beta_i^* = \max \operatorname{argmin}\{ps_{\eta-1} \mid \eta \in \gamma_+(X''_{i-1,i}X_i), \gamma_+(X''_{i-1,i}X_i) \neq \emptyset\}$, and define the prefix $X'_{i,i+1}$ and suffix $X''_{i,i+1}$ of $X_{i,i+1}$ similar to the former version based on the calculation of $\mathrm{rm}_{X''_{i-1,i}X_iX_{i,i+1}}(\beta_i^*)$. In this way, input sequence X can be decomposed into m disjoint subsequences $X''_{i-1,i}X_iX'_{i,i+1}$ for $i \in [1,m]$. But for $i \in [2,m]$, the determination of $X''_{i-1,i}$ for subsequence $X''_{i-1,i}X_iX'_{i,i+1}$ depends on the calculation of $X'_{i-1,i}$ from the last subsequence $X''_{i-2,i-1}X_{i-1}X_{i-1,i}$. The partition is naturally suitable for the implementation of sequential algorithm, not the parallel one.

In order to circumvent the iterative dependency for developing a domain-decomposed parallel algorithm for computing $\mathrm{MAX}(X)$, the former definition of $\beta_i^* = \max \operatorname{argmin}\{ps_{\eta-1} \mid \eta \in \gamma_+(X_{i-1,i}X_i), \gamma_+(X_{i-1,i}X_i) \neq \emptyset\}$ permits the computation of β_i^* for all $i \in [1,m-1]$ independently and in parallel by all subsequence-hosting processors. Also the calculation of $X'_{i,i+1}$ can be done in parallel by the i^{th} processor hosting subsequence $X_{i-1,i}X_iX_{i,i+1}$ for all $i \in [1,m-1]$, then in turn $\mathrm{MAX}(X_{i-1,i}X_iX'_{i,i+1})$ can be done in parallel by the i^{th} processor for all $i \in [1,m]$. Note that $\mathrm{MAX}(X_{i-1,i}X_iX'_{i,i+1})$ contains maximal monotone subsequences with the starting indices in prefixes $X'_{i-1,i}$ for all $i \in [2,m]$, which should be eventually discarded after the i^{th} processor obtains $\beta(X'_{i-1,i})$ from the $(i-1)^{\text{th}}$ processor.

Theorem 7 *Let X be a non-empty real-valued sequence with an* rm*-localized sequential partition* $\mathcal{P}(X) = \left(X_1, X_{1,2}, X_2, X_{2,3}, X_3, \ldots, X_{m-1}, X_{m-1,m}, X_m\right)$ *and its derived sequential partition* $\tilde{\mathcal{P}}(X) = \left(X''_{\eta-1,\eta} X_\eta X'_{\eta,\eta+1}\right)_{\eta=1}^{m}$. *Then*:

1. *The partition* $\tilde{\mathcal{P}}(X)$ *is* MAX*-independent*: $MAX(X) = \bigcup_{\eta=1}^{m} MAX\left(X''_{\eta-1,\eta} X_\eta X'_{\eta,\eta+1}\right)$, *and*

2. *For all* $i \in \{1, 2, \ldots, m\}$,

$$\begin{aligned}&\mathrm{MAX}\left(X''_{i-1,i} X_i X'_{i,i+1}\right)\\ &= \mathrm{MAX}\left(X_{i-1,i} X_i X'_{i,i+1}\right) - \left\{Y \in \mathrm{MAX}\left(X_{i-1,i} X_i X'_{i,i+1}\right) \mid \alpha(Y) \in \gamma\left(X'_{i-1,i}\right)\right\}\end{aligned}$$

Hence,

$$\begin{aligned}&MAX(X) = \bigcup_{\eta=1}^{m} MAX\left(X''_{\eta-1,\eta} X_\eta X'_{\eta,\eta+1}\right)\\ &= \bigcup_{\eta=1}^{m}\left(MAX\left(X_{\eta-1,\eta} X_\eta X'_{\eta,\eta+1}\right) - \left\{Y \in MAX\left(X_{\eta-1,\eta} X_\eta X'_{\eta,\eta+1}\right) \mid \alpha(Y) \in \gamma\left(X'_{\eta-1,\eta}\right)\right\}\right).\end{aligned}$$

Proof. For part 1, in order to apply Lemma 5 to prove the MAX-independence of $\tilde{\mathcal{P}}(X)$, it suffices, via the equivalence in Lemma 6, to show that for all $\mathrm{i} \in \{1, 2, \ldots, \mathrm{m}-1\}$, the subsequence $X''_{i-1,i} X_i X'_{i,i+1}$ satisfies the minimum prefix-sum condition (at the ending index), that is,

$$ps_{\beta\left(X''_{i-1,i} X_i X'_{i,i+1}\right)} = \min\left\{ps_\eta \,\middle|\, \eta \in \left[\alpha\left(X''_{i-1,i} X_i X'_{i,i+1}\right) - 1, \beta\left(X''_{i-1,i} X_i X'_{i,i+1}\right)\right]\right\}.$$

For $i = 1$, we have $X''_{0,1} = \emptyset$ and $X''_{i-1,i} X_i X'_{i,i+1} = X_1 X'_{1,2}$. Consider the emptiness of $\gamma_+\left(X_{i-1,i} X_i\right) = \gamma_+(X_1)$ for the existence of $\beta_i^* = \beta_1^*$ and determining $X'_{i,i+1} = X'_{1,2}$ via $\mathrm{rm}_{X_{i-1,i} X_i X_{i,i+1}}(\beta_i^*) = \mathrm{rm}_{X_1 X_{1,2}}(\beta_1^*)$.

Case when $\gamma_+(X_1) = \emptyset$: We have $X'_{1,2} = \emptyset$, and $X''_{i-1,i}X_iX'_{i,i+1} = X_1$. Note that, as $\gamma_+(X_1) = \emptyset$, $ps_{\alpha(X_1)-1} \geq ps_{\alpha(X_1)} \geq \ldots \geq ps_{\beta(X_1)}$, and the subsequence $X''_{0,1}X_1X'_{1,2} = X_1$ satisfies the minimum prefix-sum condition.

Case when $\gamma_+(X_1) \neq \emptyset$ and β_1^* exists in $\gamma_+(X_1)$: The proof of lemma 6 shows the existence of the following finite lower-envelope sequence of increasing indices in $\gamma(X_1)$: $\alpha(X_1) \leq i_1 < \mathrm{rm}_{X_1}(i_1) < i_2 < \mathrm{rm}_{X_1}(i_2) < \ldots < i_j = \beta_1^* \leq \beta(X_1)$ for some positive integer *j* such that: (1) $i_1, i_2, \ldots, i_j \in \gamma_+(X_1)$ where i_1 is the least indices in $\gamma_+(X_1)$, and $i_j = \beta_1^*$, (2) the prefix sum ps_η is a decreasing function of η over $[\alpha(X_1) - 1, i_1 - 1] \cup \left(\cup_{k=1}^{j-1}\left[\mathrm{rm}_{X_1}(i_k), i_{k+1} - 1\right]\right)$, and (3) for all $\eta \in [1, j]$, $ps_{\mathrm{rm}_{X_1}(i_\eta)} = \min\{ps_k \mid k \in \left[i_\eta - 1, \mathrm{rm}_{X_1}(i_\eta)\right]\}$. Combining (1), (2) and (3) above, we have $ps_{\beta_1^*-1} = ps_{i_j-1} = \min\{ps_\eta \mid \eta \in [\alpha(X_1) - 1, \beta_1^* - 1]\}$ which satisfies the definition of β_1^*.

We now consider the candidate index position of $\mathrm{rm}_{X''_{i-1,i}X_iX'_{i,i+1}}(\beta_1^*) = \mathrm{rm}_{X_1X_{1,2}}(\beta_1^*)$.

Subcase when $\mathrm{rm}_{X_1X_{1,2}}(\beta_1^*) \in [\beta_1^* + 1, \beta(X_1)]$: We have $X'_{1,2} = \emptyset$. As in the proof of Lemma 6, we continue to extend the lower-envelope sequence above for this case ($i = 1$ with $\gamma_+(X_1) \neq \emptyset$):

$$\alpha(X_1) \leq i_1 < \mathrm{rm}_{X_1}(i_1) < i_2 < \mathrm{rm}_{X_1}(i_2) < \ldots < i_j = \beta_1^*$$
$$< \mathrm{rm}_{X_1}(\beta_1^*) = \mathrm{rm}_{X_1X_{1,2}}(\beta_1^*) \leq \beta(X_1),$$

and the prefix sum ps_η is a decreasing function of η over

$$[\alpha(X_1) - 1, i_1 - 1] \cup \left(\cup_{k=1}^{j-1}\left[\mathrm{rm}_{X_1}(i_k), i_{k+1} - 1\right]\right) \cup \left[\mathrm{rm}_{X_1}(\beta_1^*), \beta(X_1)\right],$$

and $ps_{\mathrm{rm}_{X_1}(\beta_1^*)} = \min\{ps_k | k \in \left[\beta_1^* - 1, \mathrm{rm}_{X_1}(\beta_1^*)\right]\}$. Thus the subsequence $X''_{0,1}X_1X'_{1,2} = X_1$ satisfies the minimum prefix-sum condition: $ps_{\beta(X_1)} = \min\{ps_\eta \mid \eta \in [\alpha(X_1) - 1, \beta(X_1)]\}$.

Subcase when $\mathrm{rm}_{X_1X_{1,2}}(\beta_1^*) \in \gamma(X_{1,2})$: We have $X'_{1,2}$ to be the prefix of $X_{1,2}$ with the index subrange $[\alpha(X_{1,2}), \mathrm{rm}_{X_1X_{1,2}}(\beta_1^*)]$. Translate the lower-envelope sequence above for this subcase in the context of the subsequence $X''_{0,1}X_1X'_{1,2} = X_1X'_{1,2}$ and continue to extend to cover $X'_{1,2}$:

$$\alpha(X_1X'_{1,2}) \le i_1 < \mathrm{rm}_{X_1X'_{1,2_1}}(i_1) < i_2 < \mathrm{rm}_{X_1X'_{1,2}}(i_2) < \ldots < i_j = \beta_1^*$$
$$\le \beta(X_1) < \mathrm{rm}_{X_1X'_{1,2}}(\beta_1^*) = \beta(X_1X'_{1,2}),$$

and the prefix sum prefix sum ps_η is a decreasing function of η over

$$[\alpha(X_1X'_{1,2}) - 1, i_1 - 1] \cup \left(\cup_{k=1}^{j-1}\left[\mathrm{rm}_{X_1X'_{1,2}}(i_k), i_{k+1} - 1\right]\right),$$

and $ps_{\mathrm{rm}_{X_1X'_{1,2}}(\beta_1^*)} = \min\left\{ps_k | k \in \left[\beta_1^* - 1, \mathrm{rm}_{X_1X'_{1,2}}(\beta_1^*)\right]\right\}$. Thus the subsequence $X''_{0,1}X_1X'_{1,2} = X_1X'_{1,2}$ satisfies the minimum prefix-sum condition: $ps_{\beta(X_1X'_{1,2})} = ps_{\mathrm{rm}_{X_1X'_{1,2}}(\beta_1^*)} = \min\{ps_\eta \mid \eta \in [\alpha(X_1X'_{1,2}) - 1, \beta(X_1X'_{1,2})]\}$.

For $i \in \{2, 3, \ldots, m-1\}$, we establish the global minimum prefix-sum condition at the ending index for the subsequence $X_{i-1,i}X_iX'_{i,i+1}$, and therefore deduce that for its suffix $X''_{i-1,i}X_iX'_{i,i+1}$. Applying analogous arguments as for the cases when $i = 1$ with $X''_{0,1}X_1X'_{1,2} = X_1X'_{1,2}$ yields the desired condition for $X''_{i-1,i}X_iX'_{i,i+1}$.

For part 2, when $i = 1$, we have $X_{i-1,i} = X_{0,1} = \emptyset$, $X''_{0,1} = \emptyset$, and $\gamma(X'_{i-1,i}) = \gamma(X'_{0,1}) = \emptyset$. Now,

$$\mathrm{MAX}(X_{0,1}X_1X'_{1,2}) - \{Y \in \mathrm{MAX}(X_{0,1}X_1X'_{1,2}) \mid \alpha(Y) \in \gamma(X'_{0,1}) = \emptyset\}$$
$$= \mathrm{MAX}(X_{0,1}X_1X'_{1,2}) = \mathrm{MAX}(X_1X'_{1,2}) = \mathrm{MAX}(X''_{0,1}X_1X'_{1,2}),$$

as desired in the MAX-set equality.

For $i \in \{2, 3, \ldots, m\}$, we consider the emptiness of $X'_{i-1,i}$. If $X'_{i-1,i} = \emptyset$, then $\gamma\left(X'_{i-1,i}\right) = \emptyset$ and $X''_{i-1,i} = X_{i-1,i}$, and the MAX-set equality is satisfied:

$$\mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right) - \left\{Y \in \mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right) \mid \alpha(Y) \in \gamma\left(X'_{i-1,i}\right) = \emptyset\right\}$$
$$= \mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right) = \mathrm{MAX}\left(X''_{i-1,i}X_iX'_{i,i+1}\right)$$

Assume now that $X'_{i-1,i} \neq \emptyset$, we have $\beta^*_{i-1} = \max \mathrm{argmin}\left\{ps_{\eta-1} \mid \eta \in \gamma_+\left(X_{i-2,i-1}X_{i-1}\right)\right\}$ and $\mathrm{rm}_{X_{i-2,i-1}X_{i-1}}(\beta^*_{i-1}) = \beta\left(X'_{i-1,i}\right) \in \gamma\left(X_{i-1,i}\right)$. An immediate consequence of the non-emptiness of $X'_{i-1,i}$ is the encapsulation of $X'_{i-1,i}$-originating monotone subsequences within $X'_{i-1,i}$. We first prove the following Claim.

Claim *For every monotone subsequence Y of X, if $\alpha(Y) \in \gamma_+(X'_{i-1,i})$, then $\beta(Y) \in \gamma_+\left(X'_{i-1,i}\right)$ also.*

Proof. To see the claim, let Y be a monotone subsequence of X with $\alpha(Y) \in \gamma_+(X'_{i-1,i})$. Note that since every non-empty prefix or suffix of Y has a positive cumulative sum, in particular, both terms $x_{\alpha(Y)}$ and $x_{\beta(Y)}$ are positive. The definition of rm_X function gives that $x_{\mathrm{rm}_{X_{i-2,i-1}X_{i-1}X_{i-1,i}}(\beta^*_{i-1})} < 0$. Thus, we have:

$$\alpha\left(X_{i-2,i-1}X_{i-1}\right) \leq \beta^*_{i-1} \leq \beta\left(X_{i-2,i-1}X_{i-1}\right) = \alpha\left(X'_{i-1,i}\right) - 1 < \alpha\left(X'_{i-1,i}\right)$$
$$\leq \alpha(Y) < \mathrm{rm}_{X_{i-2,i-1}X_{i-1}X_{i-1,i}}(\beta^*_{i-1}) = \beta\left(X'_{i-1,i}\right),$$

The definition of rm_X function implies that $ps_{\alpha(Y)-1} > ps_{\beta^*_{i-1}-1} \geq ps_{\mathrm{rm}_{X_{i-2,i-1}X_{i-1}X_{i-1,i}}(\beta^*_{i-1})}$. This, together with an application of Lemma 2 to the monotonicity of Y in X that $ps_{\alpha(Y)-1}$ is the unique minimum of all ps_η for all $\eta \in [\alpha(Y) - 1, \beta(Y)]$, we must have $\beta(Y) < \mathrm{rm}_{X_{i-2,i-1}X_{i-1}X_{i-1,i}}(\beta^*_{i-1}) = \beta\left(X'_{i-1,i}\right)$, and $\beta(Y)) \in \gamma_+\left(X'_{i-1,i}\right)$. This completes the proof of the claim. ■

Now we first show that:

$$\mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right) - \left\{Y \in \mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right) \mid \alpha(Y) \in \gamma\left(X'_{i-1,i}\right)\right\}$$
$$\subseteq \mathrm{MAX}\left(X''_{i-1,i}X_iX'_{i,i+1}\right).$$

Let $Z \in \mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right) - \left\{Y \in \mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right) \mid \alpha(Y) \in \gamma\left(X'_{i-1,i}\right)\right\}$ be arbitrary; Z is maximal monotone in $X_{i-1,i}X_iX'_{i,i+1}$, and since $\alpha(Z) \notin \gamma\left(X'_{i-1,i}\right)$, Z is a monotone subsequence of $X''_{i-1,i}X_iX'_{i,i+1}$. Need to show that Z is maximal monotone in $X''_{i-1,i}X_iX'_{i,i+1}$. Because Z is monotone in $X''_{i-1,i}X_iX'_{i,i+1}$, there must exists an arbitrary supersequence $W \supseteq Z$ with $W \in \mathrm{MAX}\left(X''_{i-1,i}X_iX'_{i,i+1}\right)$. Since W is monotone in $X''_{i-1,i}X_iX'_{i,i+1}$, it must also be monotone in $X_{i-1,i}X_iX'_{i,i+1}$. The maximality of the monotonicity of Z in $X_{i-1,i}X_iX'_{i,i+1}$ indicates that every proper supersequence of Z in $X_{i-1,i}X_iX'_{i,i+1}$ is not monotone. Then the monotonicity of W in $X_{i-1,i}X_iX'_{i,i+1}$ imply that $Z = W = \mathrm{MAX}\left(X''_{i-1,i}X_iX'_{i,i+1}\right)$, as desired.

For the reverse containment, consider an arbitrary $Z \in \mathrm{MAX}\left(X''_{i-1,i}X_iX'_{i,i+1}\right)$; clearly $\alpha(Z) \notin \gamma\left(X'_{i-1,i}\right)$. Need to show that Z is maximal monotone in $X_{i-1,i}X_iX'_{i,i+1}$. Since Z is monotone in $X''_{i-1,i}X_iX'_{i,i+1}$, so it is monotone in $X_{i-1,i}X_iX'_{i,i+1}$ also. Then there must exists an arbitrary supersequence $W \supseteq Z$ with $W \in \mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right)$ and the candidate index position of $\alpha(W)$. If $\alpha(W) \in \gamma_+\left(X'_{i-1,i}\right)$, then the claim above gives that $\beta(W) \in \gamma_+\left(X'_{i-1,i}\right)$, and this contradicts to the containment of Z by W. Therefore, W is a monotone subsequence in $X''_{i-1,i}X_iX'_{i,i+1}$. The maximality of the monotonicity of Z in $X''_{i-1,i}X_iX'_{i,i+1}$ indicates that every proper supersequence of Z in $X''_{i-1,i}X_iX'_{i,i+1}$ is not monotone. Then the monotonicity of W in $X''_{i-1,i}X_iX'_{i,i+1}$ imply that $Z = W = \mathrm{MAX}\left(X_{i-1,i}X_iX'_{i,i+1}\right)$ and $\alpha(Z) = \alpha(W) \notin \gamma_+\left(X'_{i-1,i}\right)$, as desired. ■

CHAPTER IV

PROBABILISTIC ANALYSIS OF THE LOCALITY CONDITION

VIA RANDOM WALK

The structural decomposition of a non-empty real-valued sequence X in Theorem 7 suggests a basis for an ideal rm-localized decomposition of X with length-balance and MAX-independence. While the derived MAX-independent decomposition $\tilde{\mathcal{P}}(X)$ is the sequential partition $\left(X''_{\eta-1,\eta}X_\eta X'_{\eta,\eta+1}\right)_{\eta=1}^{m}$ in m pairwise disjoint subsequences, our domain-decomposed parallel algorithm computing MAX(X) will employ m processors with the i^{th} processor hosting the subsequence $X_{i-1,i}X_iX_{i,i+1}$ for $\mathrm{i} \in \{1, 2, \ldots, \mathrm{m}\}$. The subsequences $X_{i-1,i}X_iX_{i,i+1}$ and $X_{i,i+1}X_{i+1}X_{i+1,i+2}$ hosted in successive i^{th} and $(i+1)^{\text{th}}$ processors have the common subsequence $X_{i,i+1}$ which is a buffer to capture the rm-locality originated from $X_{i-1,i}X_i$ and a floating separation between the two successive MAX-sets: $\mathrm{MAX}\left(X''_{i-1,i}X_iX'_{i,i+1}\right)$ and $\mathrm{MAX}\left(X''_{i,i+1}X_{i+1}X'_{i+1,i+2}\right)$. A longer common subsequence facilitates the satisfiability of the rm-locality of the preceding subsequence while a shorter one avoids redundant computation among successive processors.

In this chapter we analyze the length bound of the common subsequences probabilistically for random sequences of normally-distributed terms — via the theory of random walk.

4.1 Introduction to Random Walk

The basic introduction to random walks can be found in Chapter XII of Feller [Fel71]. Let X_1, $X_2, \ldots$ be a sequence of independent and identically distributed random variables with common distribution function. Denote by $\left(S_\eta\right)_{\eta=0}^{\infty}$ the sequence of prefix-sum random variables with $S_0 = 0$ and $S_i = \sum_{\eta=1}^{i} X_\eta$ for $i \geq 1$, which corresponds to a general random walk for which S_i gives the position at epoch/index i.

A record value occurs at (random) epoch $i \geq 1$ corresponds to the probabilistic event $\{S_i > S_\eta$ for each $\eta \in [0, i-1]\}$. For every positive integer j, the j^{th} strict ascending ladder epoch random variable is the index of the j^{th} occurrence of the probabilistic event above. The first strict ascending ladder epoch T^+ is define by

$$\{T^+ = i\} = \left\{S_1 \leq 0, \ldots, S_{i-1} \leq 0, S_i > 0\right\}.$$

We define analogously the notions of: (1) strict descending ladder epochs by reversing the defining inequality from ">" to "<", and (2) weak ascending and weak descending epochs by replacing the defining inequalities by "≥" and "≤", respectively. For example the first weak descending ladder epoch $\overline{T_1^-}$ is define by

$$\{\overline{T_1^-} = i\} = \left\{S_1 > 0, \ldots, S_{i-1} > 0, S_i \leq 0\right\}.$$

The first strict ascending ladder epoch is the random index of the first entry into $(0, +\infty)$, and the continuation of the random walk beyond this epoch is a probabilistic replica of the entire random walk. Other variants of (strict/weak, ascending/descending) ladder epoch yield similar behavior, therefore the important conclusions concerning the random walks can be derived from a study of the first ladder epoch and its corresponding value of S_i (ladder height). Figure 4.1 depicts a random walk with its strict ascending epochs and weak descending ladder epochs.

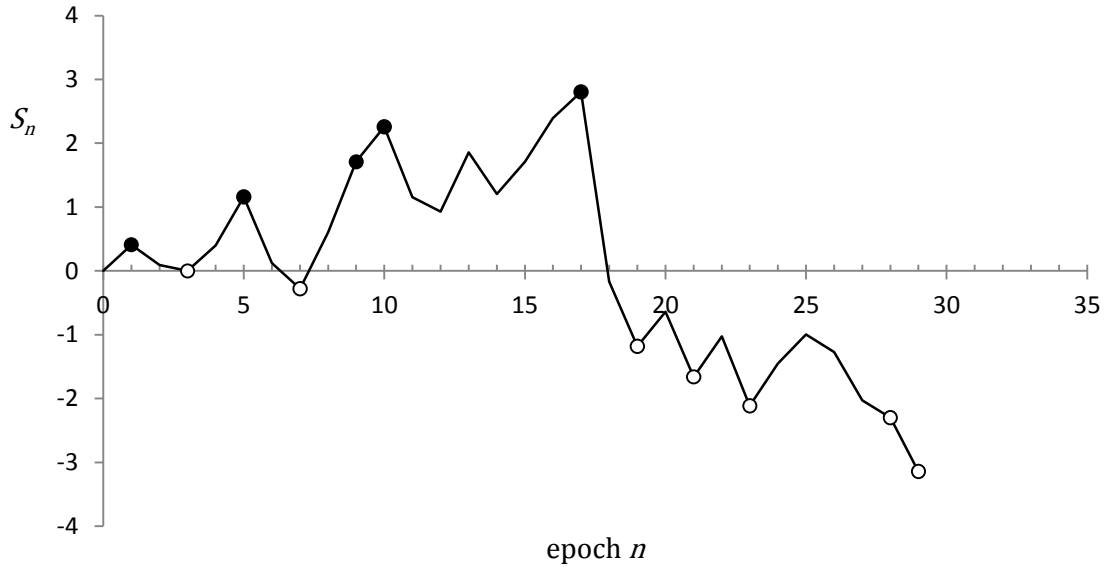

Figure 4.1: Ladder epochs. The black dots correspond to strict ascending ladder epochs and ladder heights, and white dots to weak descending ladder epochs and ladder heights. The first strict ascending ladder epoch occurs at $n = 1$, and the first weak descending ladder epoch occurs at $n = 3$.

4.2 Conditional Weak Descending Ladder Epoch

Viewing the input sequence X in the MAX-computation in a probabilistic context, the nearest right match function can be described using a variant of the first weak descending ladder epoch.

For a sequence of pairwise independent and identically distributed random variables $(X_\eta)_{\eta=1}^{\infty}$, if the j^{th} weak descending ladder epoch occurs at epoch $i_j > 0$, then for the $(j+1)^{\text{th}}$ weak descending ladder epoch at $i_{j+1} > i_j$, we have $S_{i_j} < S_{i_j+1}, \ldots, S_{i_j} < S_{S_{i_{j+1}-1}}$, $S_{i_j} \geq S_{i_{j+1}}$. The descending behavior between the j^{th} and the $(j+1)^{\text{th}}$ weak descending ladder epochs is similar to find the nearest-smaller-or-equal right-match value of ladder height S_{i_j}. Then the expected value of the first weak descending ladder epoch indicates the average length of epoch to find the next smaller or equal ladder heights.

The first weak descending ladder epoch $\overline{T_1^-}$ is the epoch of the first entry into $[0, -\infty)$ of ladder height. There are two different paths that lead to the first weak descending ladder epoch. One path is under the condition that random variable $X_1 > 0$, and the other with $X_1 \leq 0$. The following Figure 4.2 illustrates the two different paths.

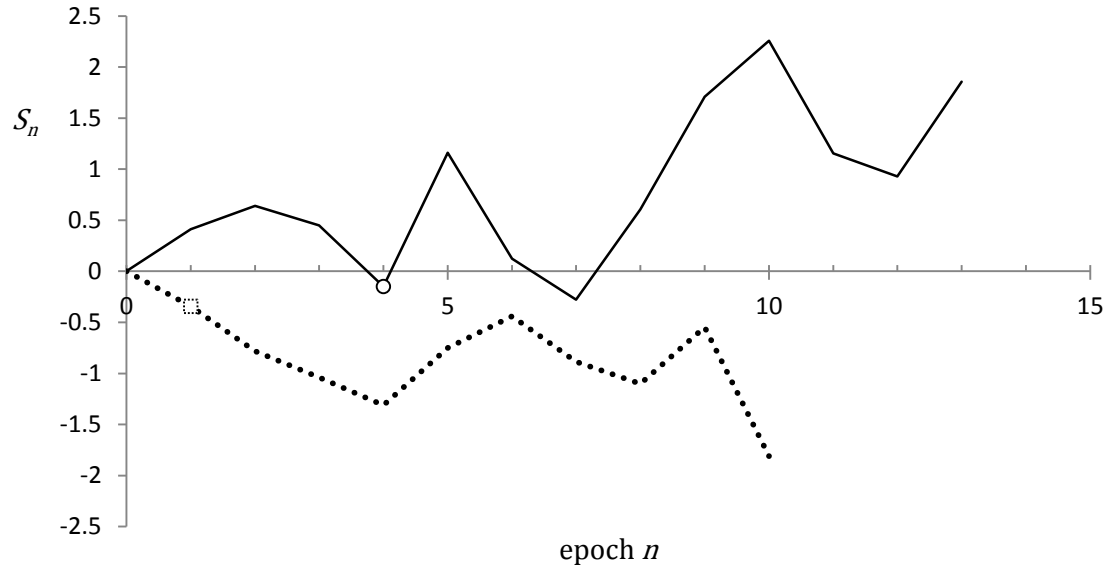

Figure 4.2: The first weak descending ladder epochs. The dot corresponds to the first weak descending ladder epoch with $X_1 > 0$, and the square to the first weak descending ladder epoch with $X_1 \leq 0$.

Recall that the $\mathrm{rm}_X(i)$ function is to find the nearest smaller or equal value of ps_{i-1} for positive term $x_i > 0$ of real-valued sequence X. We assume that X is the sequence of mutually independent and identically distributed random variables $(X_\eta)_{\eta=1}^{\infty}$, then for arbitrary $X_i > 0$, to find the smallest $j > i$ such that $S_j \leq S_{i-1}$ is equivalent to find the first weak descending ladder epoch under the condition that $X_1 > 0$. Denote by T the conditional first weak descending ladder epoch, $T = \overline{T_1^-} \mid X_1 > 0$. Therefore the average index-difference between $(i-1)$ and $\mathrm{rm}_X(i)$ can be measured by the expected value of T.

The construction of the MAX-independent sequential partition $\tilde{\mathcal{P}}(X)$ from an rm-localized sequential partition $\mathcal{P}(X)$ requires that for every $i \in [1, m-1]$ with non-empty $\gamma_+(X_{i-1,i}X_i)$, $\mathrm{rm}_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*) \in [\beta_i^* + 1, \beta(X_{i-1,i}X_iX_{i,i+1})]$. We produce a probabilistic upper bound on the length of the common subsequences in $\tilde{\mathcal{P}}(X)$ via the mean and variance of *T*.

Remark 8 *Ideally in* $\tilde{\mathcal{P}}(X)$*, we desire that*:

$$\begin{aligned}|X_{i,i+1}| &= \left|\left[\alpha(X_{i,i+1}), rm_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*)\right]\right| \\ &\leq |[\beta_i^*, rm_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*)]| = rm_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*) - \beta_i^* + 1.\end{aligned}$$

Thus, if we select the common subsequence $X_{i,i+1}$ *such that* $|X_{i,i+1}| \geq \lceil E(T) + \delta\sqrt{Var(T)} \rceil$ *for some positive real* δ*, then we have the following chain of probabilistic events*:

$$\begin{aligned}&\{rm_{X_{i-1,i}X_iX_{i,i+1}}(\beta_i^*) - \beta_i^* + 1 \geq |X_{i,i+1}|\} \\ &= \{rm_{X_{i-1,i}X_iX_{i,i+1}}(j) - j + 1 \geq |X_{i,i+1}|\} \cap \{j = \beta_i^*\} \\ &\subseteq \{rm_{X_{i-1,i}X_iX_{i,i+1}}(j) - j + 1 \geq |X_{i,i+1}|\} \\ &\subseteq \left\{rm_{X_{i-1,i}X_iX_{i,i+1}}(j) - j + 1 - E(T) \geq \delta\sqrt{Var(T)}\right\},\end{aligned}$$

where j *is a random index, and in accordance with Chebyshev's inequality,*

$$\begin{aligned}&prob\left(random\ index\text{-}difference\ rm_{X_{i-1,i}X_iX_{i,i+1}}(j) - j + 1 \geq |X_{i,i+1}|\right) \\ &\leq prob\left(T - E(T) \geq \delta\sqrt{Var(T)}\right) \leq prob\left(|T - E(T)| \geq \delta\sqrt{Var(T)}\right) \leq \frac{1}{\delta^2},\end{aligned}$$

or, equivalently,

$$\begin{aligned}&prob\left(random\ index\text{-}difference\ rm_{X_{i-1,i}X_iX_{i,i+1}}(j) - j + 1 < |X_{i,i+1}|\right) \\ &\geq prob\left(|T - E(T)| < \delta\sqrt{Var(T)}\right) \geq 1 - \frac{1}{\delta^2}.\end{aligned}$$

These will be applied to bound the likelihood of (non-)satisfiability of the rm-locality condition for $\mathcal{P}(X)$.

4.3 Relations between Conditional and Unconditional First Weak Descending Ladder Epochs

We now relate the conditional first weak descending ladder epoch T to the unconditional one $\overline{T_1^-}$ and then, in an appropriate probabilistic setting, the means and variances of the two random variables.

For a sequence of pairwise independent and identically distributed random variables X_1, X_2, . . . and its associated random-walk sequence $\left(S_\eta\right)_{\eta=0}^{\infty}$ of prefix-sum random variables, assume hereinafter that $\left(X_\eta\right)_{\eta=1}^{\infty}$ follows a common random variable X_1 with $\text{prob}(X_1 > 0) \geq 0$. For notational simplicity, denote by $p_{>0}$ and $p_{\leq 0}$ $(= 1 - p_{>0})$ the probabilities $\text{prob}(X_1 > 0)$ and $\text{prob}(X_1 \leq 0)$, respectively.

The unconditional and conditional ladder epochs $\overline{T_1^-}$ and T $(= \overline{T_1^-} \mid X_1 > 0)$ have sample spaces of $\{1, 2, \ldots.\}$ and $\{2, 3, \ldots\}$, respectively, and for every $t \in \{2, 3, \ldots\}$,

$$
\begin{aligned}
&\text{prob}\,(T = t) \\
&= \text{prob}(\overline{T_1^-} = t \mid X_1 > 0) \\
&= \frac{\text{prob}\,(\overline{T_1^-} = t \cap X_1 > 0)}{\text{prob}\,(X_1 > 0)} \\
&= \frac{1}{p_{>0}} \text{prob}\,(\overline{T_1^-} = t).
\end{aligned}
$$

Line four of the above equation is due to the subset-containment of the probabilistic events: $\{\overline{T_1^-} = t, t \geq 2\} \subseteq \{X_1 > 0\}$, i.e., if $\overline{T_1^-} \geq 2$, then $X_1 > 0$.

Lemma 9 *Assume that the mean and the variance of the unconditional weak descending ladder epoch* $\overline{T_1^-}$ *exist. The means and variances of the unconditional and conditional ladder epochs* $\overline{T_1^-}$ *and* $T = \overline{T_1^-} \mid X_1 > 0$ *are related as follows:*

1. $E(T) = \frac{1}{p_{>0}} E(\overline{T_1^-}) - \frac{p_{\leq 0}}{p_{>0}}$

2. $Var(T) = \frac{1}{p_{>0}} Var(\overline{T_1^-}) - p_{\leq 0}\left(\frac{1}{p_{>0}}(E(\overline{T_1^-}) - 1)\right)^2$

Proof. For the mean-relationship, consider:

$$\begin{aligned}
\mathrm{E}(T) &= \sum_{t\geq 2} t\,\mathrm{prob}(T = t) = \sum_{t\geq 2} t \frac{1}{p_{>0}} \mathrm{prob}\,(\overline{T_1^-} = t) \\
&= \frac{1}{p_{>0}} \sum_{t\geq 1} t\ \mathrm{prob}\,(\overline{T_1^-} = t) - \frac{1}{p_{>0}} \mathrm{prob}\,(\overline{T_1^-} = 1) \\
&= \frac{1}{p_{>0}} \mathrm{E}(\overline{T_1^-}) - \frac{1}{p_{>0}} \mathrm{prob}\,(X_1 \leq 0) \\
&= \frac{1}{p_{>0}} \mathrm{E}(\overline{T_1^-}) - \frac{p_{\leq 0}}{p_{>0}}
\end{aligned}$$

For the variance-relationship, we first note that the variance of *T* is the conditional variance of $\overline{T_1^-}$ on the event $\{X_1 > 0\}$, and when exists:

$$\begin{aligned}
\mathrm{Var}(T) = \mathrm{Var}(\overline{T_1^-} \mid X_1 > 0) &= \mathrm{E}\left(\overline{T_1^-}^2 \middle| X_1 > 0\right) - \left(\mathrm{E}(\overline{T_1^-} \mid X_1 > 0)\right)^2 \\
&= \mathrm{E}(T^2) - \left(\mathrm{E}(T)\right)^2
\end{aligned}$$

Consider:

$$\mathrm{E}(T^2) = \sum_{t\geq 2} t^2 \mathrm{prob}(T = t) = \sum_{t\geq 2} t^2 \frac{1}{p_{>0}} \mathrm{prob}\,(\overline{T_1^-} = t)$$

$$= \frac{1}{p_{>0}} \sum_{t \geq 1} t^2 \text{prob}\,(\,\overline{T_1^-} = t) - \frac{1}{p_{>0}} \text{prob}\,(\,\overline{T_1^-} = 1)$$

$$= \frac{1}{p_{>0}} \mathrm{E}\left(\,\overline{T_1^-}^2\right) - \frac{1}{p_{>0}} \text{prob}\,(\,X_1 \leq 0)$$

$$= \frac{1}{p_{>0}} \mathrm{E}\left(\,\overline{T_1^-}^2\right) - \frac{p_{\leq 0}}{p_{>0}},$$

and

$$\mathrm{Var}(T) = \mathrm{E}(T^2) - \big(\mathrm{E}(T)\big)^2$$

$$= \left(\frac{1}{p_{>0}} \mathrm{E}\left(\,\overline{T_1^-}^2\right) - \frac{p_{\leq 0}}{p_{>0}}\right) - \left(\frac{1}{p_{>0}} \mathrm{E}(\,\overline{T_1^-}) - \frac{p_{\leq 0}}{p_{>0}}\right)^2$$

$$= \frac{1}{p_{>0}} \mathrm{E}\left(\,\overline{T_1^-}^2\right) - \frac{1}{p_{>0}^2} \big(\mathrm{E}(\,\overline{T_1^-})\big)^2 + 2\frac{p_{\leq 0}}{p_{>0}^2} \mathrm{E}(\,\overline{T_1^-}) - \left(\frac{p_{\leq 0}}{p_{>0}}\right)^2 - \frac{p_{\leq 0}}{p_{>0}}$$

$$= \frac{\mathrm{E}\left(\,\overline{T_1^-}^2\right) - \big(\mathrm{E}(\,\overline{T_1^-})\big)^2}{p_{>0}} - \left(\frac{1 - p_{>0}}{p_{>0}^2}\right) \big(\mathrm{E}(\,\overline{T_1^-})\big)^2 + 2\frac{p_{\leq 0}}{p_{>0}^2} \mathrm{E}(\,\overline{T_1^-}) - \frac{p_{\leq 0}^2 + p_{\leq 0} p_{>0}}{p_{>0}^2}$$

$$= \frac{1}{p_{>0}} \mathrm{Var}(\overline{T_1^-}) - \frac{p_{\leq 0}}{p_{>0}^2} \big(\mathrm{E}(\,\overline{T_1^-})\big)^2 + 2\frac{p_{\leq 0}}{p_{>0}^2} \mathrm{E}(\,\overline{T_1^-}) - \frac{p_{\leq 0}}{p_{>0}^2}$$

$$= \frac{1}{p_{>0}} \mathrm{Var}(\overline{T_1^-}) - p_{\leq 0} \left(\frac{1}{\mathrm{p}_{>0}} (E(\,\overline{T_1^-}) - 1)\right)^2.$$

■

Remark 10 *Remark 8 and Lemma 9 suggest to seek lower and upper bounds on* $E(\,\overline{T_1^-})$ *and an upper bound on* $Var(\overline{T_1^-})$ *to obtain non-trivial bounds on* $E(T)$ *and* $Var(T)$. *Note that, by the assumption of* $prob(X_1 > 0)$, *we have* $E(\,\overline{T_1^-}) > 1$.

For our MAX-computing problem, we assume hereinafter (unless explicitly stated otherwise) that the sequence $X = \big(x_\eta\big)_{\eta=1}^n$ is a random sample from a normal distribution with mean $-a$ and variance b^2 for some positive reals a and b. That is, a sequence of pairwise independent and

identically distributed random variables $X_1, X_2, \ldots$ with a common normal distribution with mean $-a$ and variance b^2 gives rise to the observed values $x_1, x_2, \ldots$.

The negativity of the mean $-a$ of the underlying normal distribution is desired in order to avoid yielding unrealistically long minimal maximum subsequences for viable applications. Formally for the induced random-walk sequence $(S_\eta)_{\eta=0}^{\infty}$ of $(X_\eta)_{\eta=1}^{\infty}$, since $\mathrm{E}(X_1)$ is finite and negative, the first weak descending ladder epoch $\overline{T_1^-}$ has a proper probability distribution with finite mean and the random walk drafts to $-\infty$ ([Fel71] Section XII.2).

For notational simplicity, denote by λ the "mean to standard deviation" ratio $\frac{\mathrm{E}(X_1)}{\sqrt{\mathrm{Var}(X_1)}}$; $\lambda = \frac{-a}{b}$ for a common normal distribution X_1 with mean $-a$ and standard deviation b.

4.4 The Bounds on Expectations of Conditional and Unconditional First Weak Descending Ladder Epochs

We first produce upper and lower bounds of $\mathrm{E}(\overline{T_1^-})$ and $\mathrm{E}(T)$.

Theorem 11 *For a sequence of pairwise independent and identically distributed random variables* $(X_\eta)_{\eta=1}^{\infty}$ *with a negative common finite mean* $\mathrm{E}(X_1)$ *and a positive probability* $p_{>0} = \mathrm{prob}(X_1 > 0)$, *the unconditional and conditional first weak descending epochs,* $\overline{T_1^-}$ *and* $T = \overline{T_1^-} \mid X_1 > 0$ *respectively, satisfy the followings*:

1. $E(\overline{T_1^-}) = exp\left(\sum_{\eta=1}^{\infty} \frac{prob(S_\eta > 0)}{\eta}\right)$ *and*

   $E(T) = \frac{1}{p_{>0}} exp\left(\sum_{\eta=1}^{\infty} \frac{prob(S_\eta > 0)}{\eta}\right) - \frac{p_{\leq 0}}{p_{>0}}$

2. *For a common normal distribution of* $(X_\eta)_{\eta=1}^{\infty}$ *with mean* $-a$ *and variance* b^2 *for some positive reals a and b and for every positive integer l:*

$$1 < \left(\prod_{\eta=1}^{l-1}\left(1-\exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{2l}\right)}\right)\right)\right)^{-\frac{1}{2l}} \leq \mathrm{E}(\overline{T_1^-})$$

$$\leq \left(\prod_{\eta=1}^{l}\left(1-\exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{2l}\right)}\right)\right)\right)^{-\frac{1}{2l}},$$

and

$$\frac{1}{p_{>0}}\left(\prod_{\eta=1}^{l-1}\left(1-exp\left(\frac{-\lambda^2}{2sin^2\left(\frac{\eta\pi}{2l}\right)}\right)\right)\right)^{-\frac{1}{2l}} - \frac{p_{\leq 0}}{p_{>0}} \leq E(T)$$

$$\leq \frac{1}{p_{>0}}\left(\prod_{\eta=1}^{l}\left(1-exp\left(\frac{-\lambda^2}{2sin^2\left(\frac{\eta\pi}{2l}\right)}\right)\right)\right)^{-\frac{1}{2l}} - \frac{p_{\leq 0}}{p_{>0}}.$$

Proof. For part 1, we follow the development of the distribution of ladder epochs in [Fel71] Chapter XII. For each epoch $i = 1, 2, \ldots$, let $t_i = \mathrm{prob}(S_1 > 0, S_2 > 0, \ldots, S_{i-1} > 0, S_i \leq 0)$, that is, the probability of the first entrance of the induced random walk into $(-\infty, 0]$ at epoch *i*. Denote by $t(s) = \sum_{\eta=1}^{\infty} t_\eta s^\eta$, where $\mathrm{s} \in [0,1]$, the generating function of $(t_\eta)_{\eta=1}^{\infty}$. Then we have the following equation (([Fel71] Section XII.7):

$$\log\frac{1}{1-t(s)} = \sum\nolimits_{\eta=1}^{\infty}\frac{s^\eta}{\eta}\mathrm{prob}(S_\eta \leq 0).$$

Therefore, for $s \in (0,1)$,

$$\log\frac{1-t(s)}{1-s} = \log\frac{1}{1-s} - \log\frac{1}{1-t(s)}$$

$$= \sum_{\eta=1}^{\infty} \frac{s^{\eta}}{\eta} - \sum_{\eta=1}^{\infty} \frac{s^{\eta}}{\eta} \mathrm{prob}\big(S_{\eta} \leq 0\big)$$

$$= \sum_{\eta=1}^{\infty} \frac{s^{\eta}}{\eta} \mathrm{prob}\big(S_{\eta} > 0\big)$$

Note that t_i is the distribution of the first weak ladder epoch $\overline{T_1^-}$, thus $\mathrm{E}(\overline{T_1^-}) = t'(1^-)$, and we have:

$$\log \mathrm{E}(\overline{T_1^-}) = \log\left(\lim_{s\to 1^-} \frac{1-t(s)}{1-s}\right) = \lim_{s\to 1^-}\left(\log \frac{1-t(s)}{1-s}\right) = \sum_{\eta=1}^{\infty} \frac{\mathrm{prob}\big(S_{\eta} > 0\big)}{\eta},$$

$$\mathrm{E}(\overline{\mathrm{T}_1^-}) = \exp\left(\sum_{\eta=1}^{\infty} \frac{\mathrm{prob}\big(S_{\eta} > 0\big)}{\eta}\right).$$

The ladder epoch $\overline{T_1^-}$ has a finite mean if and only if the random walk drifts to $-\infty$. Hence,

$$\mathrm{E}(T) = \frac{1}{p_{>0}} \mathrm{E}(\overline{T_1^-}) - \frac{\mathrm{p}_{\leq 0}}{\mathrm{p}_{>0}}$$

$$= \frac{1}{p_{>0}} \exp\left(\sum_{\eta=1}^{\infty} \frac{\mathrm{prob}\big(S_{\eta} > 0\big)}{\eta}\right) - \frac{p_{\leq 0}}{p_{>0}}.$$

For part 2, it suffices to establish lower and upper bounds for $\mathrm{E}(\overline{T_1^-})$. We first observe that, for each $i = 1, 2, \ldots,$ S_i is a normal distribution with mean $i(-a)$ and variance ib^2, and through the standard normal distribution Z with cumulative distributive function F_Z:

$$\mathrm{prob}(\mathrm{S}_i > 0) = \mathrm{prob}\left(\frac{\mathrm{S}_i - i(-a)}{\sqrt{ib^2}} > \frac{0 - i(-a)}{\sqrt{ib^2}}\right)$$

$$= \mathrm{prob}\left(Z > \sqrt{i}|\lambda|\right)$$

$$= \frac{1}{2\pi} \int_{\sqrt{i}|\lambda|}^{\infty} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t$$

$$= 1 - F_Z\big(\sqrt{i}|\lambda|\big).$$

The tail-probability of the standard normal distribution Z can be expressed in terms of the complementary error function (erfc: $\mathbb{R} \to \mathbb{R}$, defined by $\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty \exp(-t^2)\,\mathrm{d}t$ for $x \in \mathbb{R}$) as follows: $1 - F_Z(x) = \frac{1}{2}\mathrm{erfc}\left(\frac{x}{\sqrt{2}}\right)$ for all $x \in \mathbb{R}$. Thus, we have:

$$\begin{aligned}
\mathrm{E}(\overline{T_1^-}) &= \exp\left(\sum_{\eta=1}^{\infty} \frac{\mathrm{prob}(S_\eta > 0)}{\eta}\right) \\
&= \exp\left(\sum_{\eta=1}^{\infty} \frac{1 - F_Z(\sqrt{\eta}|\lambda|)}{\eta}\right) \\
&= \exp\left(\frac{1}{2}\sum_{\eta=1}^{\infty} \frac{1}{\eta}\mathrm{erfc}\left(\sqrt{\frac{\eta}{2}}|\lambda|\right)\right).
\end{aligned}$$

We will develop lower and upper bounds for $\mathrm{erfc}(x)$ for all nonnegative reals *x*, which yield the bounds for $\mathrm{E}(\overline{T_1^-})$.

The function erfc can be represented in an alternative integral form in [Cra91]: for all nonnegative reals *x*,

$$\mathrm{erfc}(x) = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} \exp\left(-\frac{x^2}{\sin^2\theta}\right)\mathrm{d}\theta.$$

For every nonnegative real *x*, $\exp\left(-\frac{x^2}{\sin^2\theta}\right)$ is a strictly increasing function of θ over $\left[0, \frac{\pi}{2}\right]$. Let l be an arbitrary positive integer. The lower and upper Riemann sums of $\exp\left(-\frac{x^2}{\sin^2\theta}\right)$ corresponding to any size-l partition of $\left[0, \frac{\pi}{2}\right]$: $0 = \theta_0 \le \theta_1 \le \ldots \le \theta_l = \frac{\pi}{2}$ gives rise to the lower and upper bounds on erfc(*x*), respectively, as follows: for all nonnegative reals *x*,

$$\sum_{\eta=1}^{l} \exp\left(-\frac{x^2}{\sin^2\theta_{\eta-1}}\right)(\theta_\eta - \theta_{\eta-1}) \le \int_0^{\frac{\pi}{2}} \exp\left(-\frac{x^2}{\sin^2\theta}\right)\mathrm{d}\theta,$$

$$\int_0^{\frac{\pi}{2}} \exp\left(-\frac{x^2}{\sin^2\theta}\right) \mathrm{d}\theta \leq \sum_{\eta=1}^{l} \exp\left(-\frac{x^2}{\sin^2\theta_\eta}\right)(\theta_\eta - \theta_{\eta-1}),$$

where $\exp\left(-\frac{x^2}{\sin^2\theta_0}\right)$ assumes the limiting value $\lim_{\theta\to 0^+} \exp\left(-\frac{x^2}{\sin^2\theta}\right) = 0$ (see [CD02] for a similar upper bound).

When considering a size-l equipartition of $\left[0, \frac{\pi}{2}\right]$: $\theta_i = \theta_0 + i\frac{\pi}{2l} = i\frac{\pi}{2l}$ for $i = 1, 2, \ldots, l$, the bounds above translate to:

$$\frac{\pi}{2l}\sum_{\eta=1}^{l-1} \exp\left(-\frac{x^2}{\sin^2\left(\frac{\eta\pi}{2l}\right)}\right) \leq \int_0^{\frac{\pi}{2}} \exp\left(-\frac{x^2}{\sin^2\theta}\right) \mathrm{d}\theta \leq \frac{\pi}{2l}\sum_{\eta=1}^{l} \exp\left(-\frac{x^2}{\sin^2\left(\frac{\eta\pi}{2l}\right)}\right).$$

We now have:

$$\begin{aligned}
\mathrm{E}(\overline{T_1^-}) &= \exp\left(\frac{1}{2}\sum_{\eta=1}^{\infty}\frac{1}{\eta}\mathrm{erfc}\left(\sqrt{\frac{\eta}{2}}|\lambda|\right)\right) \\
&= \exp\left(\frac{1}{\pi}\sum_{\eta=1}^{\infty}\frac{1}{\eta}\int_0^{\frac{\pi}{2}}\exp\left(-\frac{\eta\lambda^2}{2\sin^2\theta}\right)\mathrm{d}\theta\right).
\end{aligned}$$

Lower-bounding $\mathrm{E}(\overline{T_1^-})$:

$$\begin{aligned}
\mathrm{E}(\overline{T_1^-}) &= \exp\left(\frac{1}{\pi}\sum_{\eta=1}^{\infty}\frac{1}{\eta}\int_0^{\frac{\pi}{2}}\exp\left(-\frac{\eta\lambda^2}{2\sin^2\theta}\right)\mathrm{d}\theta\right) \\
&\geq \exp\left(\frac{1}{2l}\sum_{\eta=1}^{\infty}\left(\frac{1}{\eta}\sum_{k=1}^{l-1}\exp\left(-\frac{\eta\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right)\right)\right) \\
&= \exp\left(\frac{1}{2l}\sum_{k=1}^{l-1}\left(\sum_{\eta=1}^{\infty}\frac{1}{\eta}\exp\left(-\frac{\eta\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right)\right)\right).
\end{aligned}$$

Because $0 < \exp\left(-\frac{\eta\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right) < 1$ for $k = 1,2, \ldots, l-1$, thus:

$$\sum_{\eta=1}^{\infty} \frac{1}{\eta} \exp\left(-\frac{\eta\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right) = -\ln\left(1 - \exp\left(-\frac{\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right)\right).$$

Therefore:

$$\mathrm{E}(\overline{T_1^-}) \geq \exp\left(\frac{1}{2l}\sum_{k=1}^{l-1}\left(-\ln\left(1 - \exp\left(-\frac{\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right)\right)\right)\right)$$

$$= \left(\prod_{\eta=1}^{l-1}\left(1 - \exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{2l}\right)}\right)\right)\right)^{-\frac{1}{2l}} > 1.$$

Upper-bounding $\mathrm{E}(\overline{T_1^-})$ via an analogous derivation:

$$\mathrm{E}(\overline{T_1^-}) = \exp\left(\frac{1}{\pi}\sum_{\eta=1}^{\infty}\frac{1}{\eta}\int_0^{\frac{\pi}{2}}\exp\left(-\frac{\eta\lambda^2}{2\sin^2\theta}\right)\mathrm{d}\theta\right)$$

$$\leq \exp\left(\frac{1}{2l}\sum_{\eta=1}^{\infty}\left(\frac{1}{\eta}\sum_{k=1}^{l}\exp\left(-\frac{\eta\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right)\right)\right)$$

$$= \exp\left(\frac{1}{2l}\sum_{k=1}^{l}\left(\sum_{\eta=1}^{\infty}\frac{1}{\eta}\exp\left(-\frac{\eta\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right)\right)\right)$$

$$= \exp\left(\frac{1}{2l}\sum_{k=1}^{l}\left(-\ln\left(1 - \exp\left(-\frac{\lambda^2}{2\sin^2\left(\frac{k\pi}{2l}\right)}\right)\right)\right)\right)$$

$$= \left(\prod_{\eta=1}^{l} \left(1 - \exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{2l}\right)} \right) \right) \right)^{-\frac{1}{2l}}.$$

Thus, we have:

$$1 < \left(\prod_{\eta=1}^{l-1} \left(1 - \exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{2l}\right)} \right) \right) \right)^{-\frac{1}{2l}} \leq \mathrm{E}(\overline{T_1^-})$$

$$\leq \left(\prod_{\eta=1}^{l} \left(1 - \exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{2l}\right)} \right) \right) \right)^{-\frac{1}{2l}},$$

and

$$\frac{1}{p_{>0}} \left(\prod_{\eta=1}^{l-1} \left(1 - exp\left(\frac{-\lambda^2}{2sin^2\left(\frac{\eta\pi}{2l}\right)} \right) \right) \right)^{-\frac{1}{2l}} - \frac{p_{\leq 0}}{p_{>0}} \leq E(T)$$

$$\leq \frac{1}{p_{>0}} \left(\prod_{\eta=1}^{l} \left(1 - exp\left(\frac{-\lambda^2}{2sin^2\left(\frac{\eta\pi}{2l}\right)} \right) \right) \right)^{-\frac{1}{2l}} - \frac{p_{\leq 0}}{p_{>0}}.$$

■

For our purpose in this study, we consider $l = 6$, and denote by $\overline{m_1^-}$ and $\overline{m_2^-}$ the lower and upper bounds on the mean $\mathrm{E}(\overline{T_1^-})$ obtained in Theorem 11:

$$\overline{m_1^-} = \left(\prod_{\eta=1}^{5} \left(1 - \exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{12}\right)} \right) \right) \right)^{-\frac{1}{12}}$$

and

$$\overline{m_2^-} = \left(\prod_{\eta=1}^{6} \left(1 - \exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{12}\right)} \right) \right) \right)^{-\frac{1}{12}}.$$

4.5 The Bounds on Variances of Conditional and Unconditional First Weak Descending Ladder Epochs

The range-constraint on $\mathrm{E}(\overline{T_1^-})$: $\mathrm{E}(\overline{T_1^-}) \in [\overline{m_1^-}, \overline{m_2^-}]$ induces an upper bound on $\mathrm{Var}(\overline{T_1^-})$ via the first-order and second-order moments of the first weak descending ladder epoch $\overline{T_1^-}$, its associate (first weak descending) ladder height $S_{\overline{T_1^-}}$, and the common distribution X_1.

Remark 12 *Consider the following scenario that will appear in upper-bounding* $Var(\overline{T_1^-})$ *and* $Var(T)$*: a quadratic polynomial p with negative leading coefficient and two real roots* s_1 *and* s_2 ($s_1 \leq s_2$) *serves as an upper bound on a nonnegative quantity v such as a variance* ($0 \leq v \leq p(s)$*, where s is a real-valued statistics*)*, and possibly, additional knowledge on s provides a range-constraint:* $s \in [c_1, c_2]$ *with* $[c_1, c_2] \cap [s_1, s_2] \neq \emptyset$:

1. *Without any range-constraint on s, the nonnegativity of v, which is upper-bounded by* $p(s)$*, gives the range-constraint for admissible values of s:* $s \in [s_1, s_2]$*, and* $v \leq p(s) \leq max\{p(s) \mid s \in \mathbb{R}\} = max\{p(s) \mid s \in [s_1, s_2]\} = p\left(\frac{1}{2}(s_1 + s_2)\right)$.

2. *The additional range-constraint on s:* $s \in [c_1, c_2]$ *yields a tighter range-constraint:* $s \in [c_1, c_2] \cap [s_1, s_2] \neq \emptyset$*, i.e.,* $s \in [max\{c_1, s_1\}, min\{c_2, s_2\}]$*, and*

$$v \le p(s) \le max\{p(s) \mid s \in \mathbb{R}\} \le \begin{cases} p(c_2) & if\ c_2 \le \frac{1}{2}(s_1 + s_2), \\ p\left(\frac{1}{2}(s_1 + s_2)\right) & if\ c_1 \le \frac{1}{2}(s_1 + s_2) \le c_2, \\ p(c_1) & if\ \frac{1}{2}(s_1 + s_2) \le c_1. \end{cases}$$

Denote by $\overline{q^-}$ and q the two quadratic polynomial forms that represent upper bounds on $\mathrm{Var}(\overline{T_1^-})$ and $\mathrm{Var}(T)$, respectively, in Theorem 13 below:

1. $\overline{q^-}(t) = 2\left(-t^2 + \left(1 + \frac{2}{\lambda^2}\right)t\right)$ with two distinct real roots $\overline{r_1^-}$ and $\overline{r_2^-}$ ($\overline{r_1^-} \le \overline{r_2^-}$): $\overline{r_1^-} = 0$ and $\overline{r_2^-} = 1 + \frac{2}{\lambda^2}$ and

2. $q(t) = -\frac{1}{p_{>0}}\left(2 + \frac{p_{\le 0}}{p_{>0}}\right)t^2 + \frac{2}{p_{>0}}\left(1 + \frac{2}{\lambda^2} + \frac{p_{\le 0}}{p_{>0}}\right)t - \frac{p_{\le 0}}{p_{>0}^2}$ with its discriminant:

$$\Delta = \frac{4}{p_{>0}^2}\left(1 + \frac{2}{\lambda^2} + \frac{p_{\le 0}}{p_{>0}}\right)^2 - \frac{4}{p_{>0}}\left(2 + \frac{p_{\le 0}}{p_{>0}}\right)\frac{p_{\le 0}}{p_{>0}^2}$$

$$= \frac{4}{p_{>0}^2}\left(1 + \frac{4}{\lambda^2} + \frac{4}{\lambda^4} + \frac{4p_{\le 0}}{\lambda^2 p_{>0}}\right) > 0,$$

and two distinct real roots r_1 and r_2 ($r_1 \le r_2$):

$$r_1 = \frac{\frac{2}{p_{>0}}\left(1 + \frac{2}{\lambda^2} + \frac{p_{\le 0}}{p_{>0}}\right) - \sqrt{\Delta}}{\frac{2}{p_{>0}}\left(2 + \frac{p_{\le 0}}{p_{>0}}\right)},$$

$$r_2 = \frac{\frac{2}{p_{>0}}\left(1 + \frac{2}{\lambda^2} + \frac{p_{\le 0}}{p_{>0}}\right) + \sqrt{\Delta}}{\frac{2}{p_{>0}}\left(2 + \frac{p_{\le 0}}{p_{>0}}\right)}.$$

Note that $r_1 > \overline{r_1^-} = 0$. From the form of the discriminant:

$$0 < \Delta = \frac{4}{p_{>0}^2}\left(1 + \frac{2}{\lambda^2} + \frac{p_{\le 0}}{p_{>0}}\right)^2 - \frac{4}{p_{>0}}\left(2 + \frac{p_{\le 0}}{p_{>0}}\right)\frac{p_{\le 0}}{p_{>0}^2}$$

$$< \frac{4}{p_{>0}^2}\left(1 + \frac{2}{\lambda^2} + \frac{p_{\leq 0}}{p_{>0}}\right)^2,$$

therefore $r_1 > \overline{r_1^-} = 0$. Similarly we have $r_2 < \overline{r_2^-} = 1 + \frac{2}{\lambda^2}$.

Theorem 13 *For a sequence of pairwise independent and identically distributed random variables* $\left(X_\eta\right)_{\eta=1}^{\infty}$ *with a negative common finite mean* $\mathrm{E}(X_1)$*, a finite common third-order absolute moment* $\mathrm{E}(|X_1|^3)$*, and a positive probability* $p_{>0} = \mathrm{prob}(X_1 > 0)$*, the unconditional and conditional first weak descending epochs,* $\overline{T_1^-}$ *and* $T = \overline{T_1^-} \mid X_1 > 0$ *respectively, satisfy the followings*:

1. *For* $\overline{T_1^-}$: $r_1 \leq E(\overline{T_1^-}) \leq r_2$ *and*

$$Var(\overline{T_1^-}) < \overline{q^-}\left(E(\overline{T_1^-})\right) \leq \begin{cases} \overline{q^-}(r_2) & if\ r_2 \leq \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}), \\ \overline{q^-}\left(\frac{1}{2}(\overline{r_1^-} + \overline{r_2^-})\right) & if\ r_1 \leq \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}) \leq r_2, \\ \overline{q^-}(r_1) & if\ \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}) \leq r_1. \end{cases}$$

and for T: $\frac{1}{p_{>0}} r_1 - \frac{p_{\leq 0}}{p_{>0}} \leq E(T) \leq \frac{1}{p_{>0}} r_2 - \frac{p_{\leq 0}}{p_{>0}}$ *and*

$$\mathrm{Var}(T) < q\left(\mathrm{E}(\overline{T_1^-})\right) \leq q\left(\frac{1}{2}(r_1 + r_2)\right).$$

2. *With a common normal distribution of* $\left(X_\eta\right)_{\eta=1}^{\infty}$ *with mean* $-a$ *and variance* b^2 *for some positive reals* a *and* b*:*

For $\overline{T_1^-}$: $\overline{m_1^-} \leq E(\overline{T_1^-}) \leq \overline{m_2^-}$ *and*

$$Var(\overline{T_1^-}) < \overline{q^-}\left(E(\overline{T_1^-})\right) \leq \begin{cases} \overline{q^-}(\overline{m_2^-}) & if\ \overline{m_2^-} \leq \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}), \\ \overline{q^-}\left(\frac{1}{2}(\overline{r_1^-} + \overline{r_2^-})\right) & if\ \overline{m_1^-} \leq \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}) \leq \overline{m_2^-}, \\ \overline{q^-}(\overline{m_1^-}) & if\ \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}) \leq \overline{m_1^-}. \end{cases}$$

and for T: $\frac{1}{p_{>0}}\overline{m_1^-} - \frac{p_{\leq 0}}{p_{>0}} \leq E(T) \leq \frac{1}{p_{>0}}\overline{m_2^-} - \frac{p_{\leq 0}}{p_{>0}}$ *and*

$$Var(T) < q\left(E(\overline{T_1^-})\right) \leq \begin{cases} q(\overline{m_2^-}) & if\ \overline{m_2^-} \leq \frac{1}{2}(r_1 + r_2), \\ q\left(\frac{1}{2}(r_1 + r_2)\right) & if\ \overline{m_1^-} \leq \frac{1}{2}(r_1 + r_2) \leq \overline{m_2^-}, \\ q(\overline{m_1^-}) & if\ \frac{1}{2}(r_1 + r_2) \leq \overline{m_1^-}. \end{cases}$$

Proof. For part 1, we first introduce the following identities and equality that relate the moments of the ladder epoch $\overline{T_1^-}$, its associated ladder height $S_{\overline{T_1^-}}$, and the common distribution X_1:

1. Applying Wald's lemma ([Asm03] Appendix A10):

$$\mathrm{E}\left(S_{\overline{T_1^-}}\right) = \mathrm{E}(X_1)\mathrm{E}(\overline{T_1^-}).$$

2. Applying Wald's second-moment identity ([Asm03] Appendix A10):

$$\mathrm{E}\left(\left(S_{\overline{T_1^-}} - \mathrm{E}(X_1)\overline{T_1^-}\right)^2\right) = \mathrm{Var}(X_1)\mathrm{E}(\overline{T_1^-}).$$

3. Applying [Ale06] Theorem 3:

$$\mathrm{E}\left(S_{\overline{T_1^-}}^2\right) = \mathrm{E}(X_1^2)\mathrm{E}(\overline{T_1^-}) + 2\mathrm{E}(X_1)\mathrm{E}(\overline{T_1^-}) \sum_{\eta=1}^{\infty} \frac{1}{\eta} \int_{0^+}^{\infty} x \mathrm{d}F_{S_\eta}(x),$$

where F_{S_η} denotes the cumulative distribution function of S_η for $\eta = 1, 2, \ldots$. Since $\mathrm{E}(X_1) < 0$ and $\int_{0^+}^{\infty} x \mathrm{d}F_{S_\eta}(x) > 0$ for $\eta = 1, 2, \ldots$, we have:

$$\mathrm{E}\left(S_{\overline{T_1^-}}^2\right) < \mathrm{E}(X_1^2)\mathrm{E}(\overline{T_1^-}).$$

We relate $\mathrm{Var}(\overline{T_1^-})$ to $\mathrm{E}(\overline{T_1^-})$ via the moments of $S_{\overline{T_1^-}}$ and X_1 as follows:

$$\mathrm{Var}(\overline{T_1^-}) = \mathrm{E}\left(\left(\overline{T_1^-} - \mathrm{E}(\overline{T_1^-})\right)^2\right) = \frac{1}{\left(\mathrm{E}(X_1)\right)^2}\mathrm{E}\left(\left(\mathrm{E}(X_1)\overline{T_1^-} - \mathrm{E}(X_1)\mathrm{E}(\overline{T_1^-})\right)^2\right)$$

$$= \frac{1}{\left(\mathrm{E}(X_1)\right)^2}\mathrm{E}\left(\left(\left(S_{\overline{T_1^-}} - \mathrm{E}(X_1)\mathrm{E}(\overline{T_1^-})\right) - \left(S_{\overline{T_1^-}} - \mathrm{E}(X_1)\overline{T_1^-}\right)\right)^2\right)$$

$$= \frac{1}{\left(\mathrm{E}(X_1)\right)^2}\mathrm{E}\left(\left(\left(S_{\overline{T_1^-}} - \mathrm{E}\left(S_{\overline{T_1^-}}\right)\right) - \left(S_{\overline{T_1^-}} - \mathrm{E}(X_1)\overline{T_1^-}\right)\right)^2\right)$$

$$\leq \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(\left(S_{\overline{T_1^-}} - \mathrm{E}\left(S_{\overline{T_1^-}}\right)\right)^2\right) + \mathrm{E}\left(\left(S_{\overline{T_1^-}} - \mathrm{E}(X_1)\overline{T_1^-}\right)^2\right)\right)$$

$$= \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\left(\mathrm{E}\left(S_{\overline{T_1^-}}^2\right) - \left(\mathrm{E}\left(S_{\overline{T_1^-}}\right)\right)^2\right) + \mathrm{E}\left(\left(S_{\overline{T_1^-}} - \mathrm{E}(X_1)\overline{T_1^-}\right)^2\right)\right)$$

$$= \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(S_{\overline{T_1^-}}^2\right) - \left(\mathrm{E}(X_1)\mathrm{E}(\overline{T_1^-})\right)^2 + \mathrm{E}\left(\left(S_{\overline{T_1^-}} - \mathrm{E}(X_1)\overline{T_1^-}\right)^2\right)\right)$$

$$= \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(S_{\overline{T_1^-}}^2\right) - \left(\mathrm{E}(X_1)\right)^2\left(\mathrm{E}(\overline{T_1^-})\right)^2 + \mathrm{Var}(X_1)\mathrm{E}(\overline{T_1^-})\right)$$

$$< \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}(X_1^2)\mathrm{E}(\overline{T_1^-}) - \left(\mathrm{E}(X_1)\right)^2\left(\mathrm{E}(\overline{T_1^-})\right)^2 + \mathrm{Var}(X_1)\mathrm{E}(\overline{T_1^-})\right)$$

$$= \frac{2\left(\left(\mathrm{Var}(X_1) + \left(\mathrm{E}(X_1)\right)^2\right)\mathrm{E}(\overline{T_1^-}) - \left(\mathrm{E}(X_1)\right)^2\left(\mathrm{E}(\overline{T_1^-})\right)^2 + \mathrm{Var}(X_1)\mathrm{E}(\overline{T_1^-})\right)}{\left(\mathrm{E}(X_1)\right)^2}$$

$$= \frac{2\left(-\left(\mathrm{E}(X_1)\right)^2\left(\mathrm{E}(\overline{T_1^-})\right)^2 + \left(2\mathrm{Var}(X_1) + \left(\mathrm{E}(X_1)\right)^2\right)\mathrm{E}(\overline{T_1^-})\right)}{\left(\mathrm{E}(X_1)\right)^2}$$

$$= 2\left(-\left(\mathrm{E}(\overline{T_1^-})\right)^2 + \left(1 + \frac{2}{\lambda^2}\right)\mathrm{E}(\overline{T_1^-})\right)$$

$$= \overline{q^-}\left(\mathrm{E}(\overline{T_1^-})\right)$$

Applying Remark 12: part 1 to the upper bound: $\mathrm{Var}(\overline{T_1^-}) < \overline{q^-}\left(\mathrm{E}(\overline{T_1^-})\right)$, we obtain for $\overline{T_1^-}$ that:

$$\overline{r_1^-} \leq \mathrm{E}(\overline{T_1^-}) \leq \overline{r_2^-}, \text{ and } \mathrm{Var}(\overline{T_1^-}) < \overline{q^-}\left(\mathrm{E}(\overline{T_1^-})\right) \leq \overline{q^-}\left(\frac{1}{2}(\overline{r_1^-} + \overline{r_2^-})\right).$$

From Lemma 9: part 2, we have:

$$\begin{aligned}
\mathrm{Var}(T) &= \frac{1}{p_{>0}}\mathrm{Var}(\overline{T_1^-}) - p_{\leq 0}\left(\frac{1}{\mathrm{p}_{>0}}(\mathrm{E}(\overline{T_1^-}) - 1)\right)^2 \\
&< \frac{2}{p_{>0}}\left(-\left(\mathrm{E}(\overline{T_1^-})\right)^2 + \left(1 + \frac{2}{\lambda^2}\right)\mathrm{E}(\overline{T_1^-})\right) - p_{\leq 0}\left(\frac{1}{\mathrm{p}_{>0}}(\mathrm{E}(\overline{T_1^-}) - 1)\right)^2 \\
&= -\frac{1}{p_{>0}}\left(2 + \frac{p_{\leq 0}}{p_{>0}}\right)\left(\mathrm{E}(\overline{T_1^-})\right)^2 + \frac{2}{p_{>0}}\left(1 + \frac{2}{\lambda^2} + \frac{p_{\leq 0}}{p_{>0}}\right)\mathrm{E}(\overline{T_1^-}) - \frac{p_{\leq 0}}{p_{>0}^2} \\
&= q\left(\mathrm{E}(\overline{T_1^-})\right).
\end{aligned}$$

Similar to the above case of $\mathrm{Var}(\overline{T_1^-}) < \overline{q^-}\left(\mathrm{E}(\overline{T_1^-})\right)$, we apply Remark 12: part 1 to the upper bound: $\mathrm{Var}(T) < q\left(\mathrm{E}(\overline{T_1^-})\right)$, and obtain for T that:

$$r_1 \leq \mathrm{E}(\overline{T_1^-}) \leq r_2 \text{ and } \mathrm{Var}(T) < q\left(\mathrm{E}(\overline{T_1^-})\right) \leq q\left(\frac{1}{2}(r_1 + r_2)\right).$$

Clearly, $[r_1, r_2] \subseteq [\overline{r_1^-}, \overline{r_2^-}]$. Applying Remark 12: part 2 to the upper bounds on $\mathrm{Var}(\overline{T_1^-})$ and $\mathrm{Var}(T)$ and Lemma 9: part 1 to $\mathrm{E}(\overline{T_1^-})$, we have:

For $\overline{T_1^-}$: $r_1 \leq E(\overline{T_1^-}) \leq r_2$ and

$$Var(\overline{T_1^-}) < \overline{q^-}\left(E(\overline{T_1^-})\right) \leq \begin{cases} \overline{q^-}(r_2) & if\ r_2 \leq \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}), \\ \overline{q^-}\left(\frac{1}{2}(\overline{r_1^-} + \overline{r_2^-})\right) & if\ r_1 \leq \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}) \leq r_2, \\ \overline{q^-}(r_1) & if\ \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}) \leq r_1; \end{cases}$$

and for T: $\frac{1}{p_{>0}} r_1 - \frac{p_{\leq 0}}{p_{>0}} \leq E(T) \leq \frac{1}{p_{>0}} r_2 - \frac{p_{\leq 0}}{p_{>0}}$ and

$$\mathrm{Var}(T) < q\left(\mathrm{E}(\overline{T_1^-})\right) \leq q\left(\frac{1}{2}(\mathrm{r}_1 + \mathrm{r}_2)\right).$$

For part 2 with a common normal distribution of $\left(X_\eta\right)_{\eta=1}^{\infty}$ with mean $-\mathrm{a}$ and variance b^2 for some positive reals a and b, Theorem 11: part 2 gives that

$$\overline{m_1^-} \leq \mathrm{E}(\overline{T_1^-}) \leq \overline{m_2^-} \text{ and } \frac{1}{p_{>0}} \overline{m_1^-} - \frac{p_{\leq 0}}{p_{>0}} \leq \mathrm{E}(T) \leq \frac{1}{p_{>0}} \overline{m_2^-} - \frac{p_{\leq 0}}{p_{>0}}.$$

Applying Remark 12: part 2 to the range-constraint $\mathrm{E}(\overline{T_1^-}) \in \left[\overline{r_1^-}, \overline{r_2^-}\right]$ and the upper bounds $\mathrm{Var}(\overline{\mathrm{T}_1^-}) < \overline{q^-}\left(\mathrm{E}(\overline{\mathrm{T}_1^-})\right)$ and $\mathrm{Var}(T) < q\left(\mathrm{E}(\overline{\mathrm{T}_1^-})\right)$ results in, respectively:.

$$\mathrm{Var}(\overline{T_1^-}) < \overline{q^-}\left(\mathrm{E}(\overline{T_1^-})\right) \leq \begin{cases} \overline{q^-}(\overline{m_2^-}) & \text{if } \overline{m_2^-} \leq \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}), \\ \overline{q^-}\left(\frac{1}{2}(\overline{r_1^-} + \overline{r_2^-})\right) & \text{if } \overline{m_1^-} \leq \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}) \leq \overline{m_2^-}, \\ \overline{q^-}(\overline{m_1^-}) & \text{if } \frac{1}{2}(\overline{r_1^-} + \overline{r_2^-}) \leq \overline{m_1^-}, \end{cases}$$

and

$$\mathrm{Var}(T) < q\left(\mathrm{E}(\overline{T_1^-})\right) \leq \begin{cases} q(\overline{m_2^-}) & \text{if } \overline{m_2^-} \leq \frac{1}{2}(r_1 + r_2), \\ q\left(\frac{1}{2}(r_1 + r_2)\right) & \text{if } \overline{m_1^-} \leq \frac{1}{2}(r_1 + r_2) \leq \overline{m_2^-}, \\ q(\overline{m_1^-}) & \text{if } \frac{1}{2}(r_1 + r_2) \leq \overline{m_1^-}. \end{cases}$$

■

Theorem 14 *For a sequence of pairwise independent and identically distributed random variables* $\left(X_\eta\right)_{\eta=1}^{\infty}$ *with a common normal distribution* with mean *a* and variance b^2 for some

positive reals a and b, *a finite common third-order absolute moment* $\mathrm{E}(|X_1|^3)$*, the variance of the unconditional first ascending epoch* $\overline{T_1}$ *satisfies the following*:

$$\begin{aligned}\mathrm{Var}(\overline{T_1}) &\leq 2\left(1+\frac{2}{\lambda^2}\right)\mathrm{E}(\overline{T_1}) \\ &\leq \frac{2}{a^2}\left(\frac{4}{3}\frac{\mathrm{E}((X_1^+)^3)}{a}+b^2\mathrm{E}(\overline{T_1})\right) \\ &< \frac{2}{a^2}\left(\frac{16}{3}\frac{\mathrm{E}((X_1^+)^3)}{a}+2(\mathrm{E}((X_1)^2)+a^2)\mathrm{E}(\overline{T_1})\right),\end{aligned}$$

where $X_1^+ = X_1$ *if* $X_1 \geq 0$ *else* $X_1^+ = 0$.

Proof. Applying [Sug07] Theorem 3.1:

$$\mathrm{Var}(\overline{T_1}) < \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\frac{16}{3}\frac{\mathrm{E}((X_1^+)^3)}{\mathrm{E}(X_1)}+2(\mathrm{E}((X_1)^2)+a^2)\mathrm{E}(\overline{T_1})\right)$$

Applying [Sug07] equation (12):

$$\mathrm{E}\left(\left(S_{\overline{T_1}}\right)^2\right) \leq \frac{4}{3}\frac{\mathrm{E}((X_1^+)^3)}{\mathrm{E}(X_1)}$$

Following the same procedure in the proof of Theorem 13, we have

$$\begin{aligned}\mathrm{Var}(\overline{T_1}) &\leq \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(\left(S_{\overline{T_1}}-\mathrm{E}\left(S_{\overline{T_1}}\right)\right)^2\right)+\mathrm{E}\left(\left(S_{\overline{T_1}}-\mathrm{E}(X_1)\overline{T_1}\right)^2\right)\right) \\ &\leq \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(\left(S_{\overline{T_1}}\right)^2\right)-\left(\mathrm{E}\left(S_{\overline{T_1}}\right)\right)^2+\mathrm{E}\left(\left(S_{\overline{T_1}}-\mathrm{E}(X_1)\overline{T_1}\right)^2\right)\right) \\ &\leq \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(\left(S_{\overline{T_1}}\right)^2\right)+\mathrm{E}\left(\left(S_{\overline{T_1}}-\mathrm{E}(X_1)\overline{T_1}\right)^2\right)\right) \\ &= \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(\left(S_{\overline{T_1}}\right)^2\right)+\mathrm{Var}(X_1)\mathrm{E}\left(S_{\overline{T_1}}\right)\right)\end{aligned}$$

$$\leq \frac{2}{a^2}\left(\frac{4}{3}\frac{\mathrm{E}((X_1^+)^3)}{a} + b^2\mathrm{E}\left(S_{\overline{T_1}}\right)\right)$$

$$< \frac{2}{a^2}\left(\frac{16}{3}\frac{\mathrm{E}((X_1^+)^3)}{a} + 2(\mathrm{E}((X_1)^2) + a^2)\mathrm{E}(\overline{T_1})\right).$$

Applying [Ale06] Theorem 2, we have:

$$\mathrm{E}\left(\left(S_{\overline{T_1}}\right)^2\right) < \mathrm{E}((X_1)^2)\mathrm{E}\left(S_{\overline{T_1}}\right)$$

From above:

$$\mathrm{Var}(\overline{T_1}) \leq \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(\left(S_{\overline{T_1}}\right)^2\right) - \left(\mathrm{E}\left(S_{\overline{T_1}}\right)\right)^2 + \mathrm{E}\left(\left(S_{\overline{T_1}} - \mathrm{E}(X_1)\overline{T_1}\right)^2\right)\right)$$

$$= \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}\left(\left(S_{\overline{T_1}}\right)^2\right) - \left(\mathrm{E}\left(S_{\overline{T_1}}\right)\right)^2 + \mathrm{Var}(X_1)\mathrm{E}(\overline{T_1})\right)$$

$$< \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\mathrm{E}((X_1)^2)\mathrm{E}(\overline{T_1}) - \left(\mathrm{E}\left(S_{\overline{T_1}}\right)\right)^2 + \mathrm{Var}(X_1)\mathrm{E}(\overline{T_1})\right)$$

$$\leq \frac{2}{\left(\mathrm{E}(X_1)\right)^2}\left(\left(\mathrm{Var}(X_1) + \left(\mathrm{E}(X_1)\right)^2\right)\mathrm{E}(\overline{T_1}) + \mathrm{Var}(X_1)\mathrm{E}(\overline{T_1})\right)$$

$$= \frac{2}{a^2}\left((2b^2 + a^2)\mathrm{E}(\overline{T_1})\right)$$

$$= 2\left(1 + \frac{2}{\lambda^2}\right)\mathrm{E}(\overline{T_1}).$$

For the normal distribution:

$$\mathrm{E}((X_1^+)^3) = \frac{b(2b^2 + a^2)}{\sqrt{2\pi}}\exp\left(-\frac{a^2}{2b^2}\right) + \frac{a(3b^2 + a^2)}{2}\left(1 + \mathrm{erf}\left(\frac{a}{\sqrt{2}b}\right)\right),$$

where $\mathrm{erf}\left(\frac{a}{\sqrt{2}b}\right) = 1 - \mathrm{erfc}\left(\frac{a}{\sqrt{2}b}\right)$.

Inequality $2\left(1+\frac{2}{\lambda^2}\right)\mathrm{E}(\overline{T_1}) \le \frac{2}{a^2}\left(\frac{4}{3}\frac{\mathrm{E}\left(\left(X_1^+\right)^3\right)}{a}+b^2\mathrm{E}(\overline{T_1})\right)$ is equivalent to:

$$\mathrm{E}(\overline{T_1}) \le \frac{4}{3}\frac{\mathrm{E}((X_1^+)^3)}{a(b^2+a^2)}$$

$$= \frac{4}{3}\frac{(2+\lambda^2)}{\sqrt{2\pi}\lambda(1+\lambda^2)}\exp\left(-\frac{\lambda^2}{2}\right)+\frac{2(3+\lambda^2)}{3(1+\lambda^2)}\left(1+\mathrm{erf}\left(\frac{\lambda}{\sqrt{2}}\right)\right)$$

First we show that for $0 \le \lambda \le \frac{1}{2}$,

$$\mathrm{E}(\overline{T_1}) \le \left(\prod_{\eta=1}^{6}\left(1-\exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{12}\right)}\right)\right)\right)^{-\frac{1}{12}}$$

$$\le (1+\lambda^2)^{\frac{1}{2}}$$

$$\le \frac{4}{3}\frac{(2+\lambda^2)}{\sqrt{2\pi}\lambda(1+\lambda^2)}\exp\left(-\frac{\lambda^2}{2}\right)+\frac{2}{3}\left(1+\frac{2}{1+\lambda^2}\right)$$

$$\le \frac{4}{3}\frac{(2+\lambda^2)}{\sqrt{2\pi}\lambda(1+\lambda^2)}\exp\left(-\frac{\lambda^2}{2}\right)+\frac{2(3+\lambda^2)}{3(1+\lambda^2)}\left(1+\mathrm{erf}\left(\frac{\lambda}{\sqrt{2}}\right)\right).$$

Second we show that for $\lambda \ge \frac{1}{2}$,

$$\mathrm{E}(\overline{T_1}) \le \left(\prod_{\eta=1}^{6}\left(1-\exp\left(\frac{-\lambda^2}{2\sin^2\left(\frac{\eta\pi}{12}\right)}\right)\right)\right)^{-\frac{1}{12}}$$

$$\le \frac{4}{3}\frac{(2+\lambda^2)}{\sqrt{2\pi}\lambda(1+\lambda^2)}\exp\left(-\frac{\lambda^2}{2}\right)+\frac{2(3+\lambda^2)}{3(1+\lambda^2)}\left(1+\mathrm{erf}\left(\frac{\lambda}{\sqrt{2}}\right)\right).$$

The proof is trivial, and we will not list the procedures. ■

CHAPTER V

PARALLEL ALGORITHM ON CLUSTER SYSTEMS FOR COMPUTING MAX

In this chapter we present the overall parallel algorithm implemented on cluster systems with subsequence-hosting processors employing the linear-time sequential algorithm computing MAX [RT99]. Experiments are performed on the cluster of High Performance Computing Center at Oklahoma State University.

5.1 Linear-Time Sequential Algorithm to Compute MAX

The following MAX_Sequential implements the linear-time sequential algorithm by Ruzzo and Tompa.

Algorithm 4: MAX_Sequential

Input: A length-*n* real-valued sequence X.

Output: The sequence of all successive minimal maximum subsequences (that is, all maximal monotone subsequences) of X occupying the low-order subarray of an array $M[1..\lceil n/2 \rceil]$.

Data Structures: On consumed input X' (prefix of X):

1. Array $M[1..(i-1)]$: the canonical list $(Y_1, Y_2, \ldots, Y_{i-1})$ of $\mathrm{MAX}(X')$;
2. Stack *St*: the longest sublist $(Y_{(1)}, Y_{(2)}, \ldots)$ (which is not necessarily contiguous) of $\mathrm{MAX}(X')$ such that

$\min\{ps_{\eta-1}(X')|\eta \in \gamma_+(X')\} = ps_{\alpha(Y_{(1)})-1}(X') < ps_{\alpha(Y_{(2)})-1}(X') < \ldots,$ and

$\max\{ps_{\eta}(X')|\eta \in \gamma_+(X')\} = ps_{\beta(Y_{(1)})}(X') \geq ps_{\beta(Y_{(2)})}(X') \geq \ldots;$

3. Subsequence Y_i: the incremental input of the longest (contiguous) subsequence of positive terms of X immediately succeeding X'.

Begin

1. Initialize:

   $M[0] := \emptyset$; $\alpha(Y_{(0)}) := 0$; $\beta(Y_{(0)}) := 0$; $St := \emptyset$; $i := 1$;

2. Compute $\gamma(Y_i)$, if non-empty:

   $\alpha(Y_i) := \min\{\eta \in \gamma_+(X') \mid \eta > \beta(Y_{i-1})\}$;

   $\beta(Y_i) := \min\{\eta \in \gamma_+(X') \mid \eta > \beta(Y_{i-1})$ and $\eta + 1 \notin \gamma_+(X')\}$;

   if $\alpha(Y_i)$ does not exist, then

      Output $M[1..(i-1)]$;

      Stop;

   end if;

3. Absorb Y_i, and, if necessary, update i, $M[1..(i-1)]$, and *St*:

   3.1. while $i > 1$ and $St \neq \emptyset$

      $j :=$ Top(*St*);

      if $ps_{\alpha(Y_j)-1} \geq ps_{\alpha(Y_i)-1}$ then Pop(*St*);

      else if $ps_{\beta(Y_j)} < ps_{\beta(Y_i)}$ then

         Pop(St); $\beta(Y_j) = \beta(Y_i)$; $i := j$;

      else goto Step 3.2;

      end if;

end while;

3.2. Push i to *St*; $M[i] := Y_i$; $i := i + 1$; goto Step 2;

End

Algorithm MAX_Sequential has been optimized from the original one described in section 2.5:

1. The contiguous subsequence of positive terms of *X* is monotone, and it can be used to check the maximality directly. In section 2.5, each positive term is treated as a subsequence.

2. A stack is used to preserve the relative positions of the monotone subsequences, and it can avoid the redundant comparisons of prefix sums.

Figure 5.1 illustrate an example input for the algorithm MAX_Sequential. Subsequence Y_1 is monotone, the index 1 is pushed on top of *St*. In the second iteration, Y_2 is found, and it is compared to the subsequence Y_1 corresponding to index 1 at the top of *St*. Because $ps_{\alpha(Y_1)-1} \geq ps_{\alpha(Y_2)-1}$, index 1 is popped off from *St*. The stack *St* is empty now, which indicates that Y_1 is the maximal monotone subsequence. Index 2 is pushed on top of *St*. In the third iteration, Y_3 is found. Because $ps_{\alpha(Y_2)-1} < ps_{\alpha(Y_3)-1}$ and $ps_{\beta(Y_2)} > ps_{\beta(Y_3)}$, index 3 is pushed on top of *St*. Now *St* contains index 2 and 3. In the fourth iteration, Y_4 is found. $ps_{\alpha(Y_3)-1} \geq ps_{\alpha(Y_4)-1}$, index 3 is popped off from *St*. The stack *St* is not empty yet, so Y_3 could be maximal monotone or be merged into a maximal monotone subsequence. Then Y_4 will compare with Y_2 that is corresponding to index 2 at the top of *St*. Because $ps_{\alpha(Y_2)-1} < ps_{\alpha(Y_4)-1}$ and $ps_{\beta(Y_2)} < ps_{\beta(Y_4)}$, Y_2, Y_3, and Y_4 are merged into Y_2.

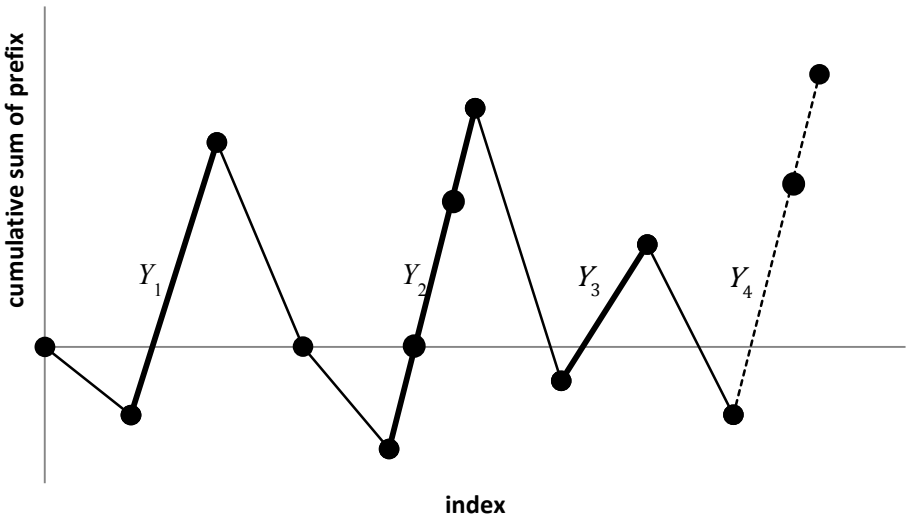

Figure 5.1: An example input of MAX_Sequential, where Y_1, Y_2, and Y_3 are in MAX(X'), the next Y_4 will be merged with Y_2, and Y_3.

Step 2 is linear. In Step 3, we see that for each Y_i, there are at most one Push and one Pop operation associated with it, so the algorithm MAX_Sequential runs in $O(n)$ time.

When sequence X is partitioned into $(X_1, X_2, \ldots, X_m)$ on m processors, each processor can run MAX_Sequential independently to compute $\text{MAX}(X) = \cup_{\eta=1}^{m} \text{MAX}(X_\eta)$ if the partition satisfies rm-closure condition. If the partition does not satisfy rm-closure condition, then we implement the PRAM-algorithm in [DS06] with MPI. The PRAM-algorithm will call the parallel algorithm for all nearest smaller values (ANSV) developed in [HH01]. But it is not necessary to find the nearest right match or left match for all the indices in order to compute MAX(X). We only need to solve the right match or left match for the starting index of every monotone subsequence. In the following section we create the optimized ANSV sequential algorithm to minimize the computations in the parallel ANSV algorithm.

5.2 All Nearest Smaller Values Sequential Algorithm

In Chapter 3, we defined the $\mathrm{rm}_X(i)$ and $\mathrm{lm}_X(i)$ respectively for the positive terms x_i, $i \in \gamma_+(X)$ of input sequence $X = (x_\eta)_{\eta=1}^{n}$:

$$\mathrm{rm}_X(i) = \begin{cases} \min\{\eta \in [i+1, \beta(X)] \mid ps_{i-1} \geq ps_\eta\} & \text{if the minimum exists,} \\ \beta(X) + 1 \ (= n+1) & \text{otherwise;} \end{cases}$$

and ,

$$\mathrm{lm}_X(i) = \begin{cases} \max\{\eta \in [\alpha(X), i-1] \mid ps_{i-1} > ps_{\eta-1}\} & \text{if the maxmum exists,} \\ \alpha(X) - 1 \ (= 0) & \text{otherwise.} \end{cases}$$

Every positive term of X is contained in a unique maximal monotone subsequence, thus between two contiguous maximal monotone subsequences Y_i and Y_{i+1} $(\alpha(Y_{i+1}) > \beta(Y_i))$ is a subsequence $Y_{i,i+1}$ with index subrange $[\beta(Y_i)+1, \alpha(Y_{i+1})-1]$. Subsequence $Y_{i,i+1}$ only contains non-positive terms. If we replace every $Y_{i,i+1}$ with a single non-positive term $x_{i,i+1} = ps_{\alpha(Y_{i+1})-1} - ps_{\beta(Y_i)}$ to transform X into X', the MAX(X) should be equal to MAX(X') (simple task can be applied to each maximal monotone subsequence of MAX(X') to translate its index subrange in the context of X). If we assume $\mathrm{rm}_X(\alpha(Y_i)) \in [\beta(Y_{j-1})+1, \alpha(Y_j)-1]$, where $j > i$, then $\mathrm{rm}_{X'}(\alpha(Y_i)) = \alpha(Y_j) - 1$. Notice that the first term of Y_j must be positive, therefore, we can redefine rm_X to be:

$$\mathrm{rm}_X(i) = \begin{cases} \min\{\eta \in \gamma_+(X) \mid \eta \geq i+1,\ ps_{i-1} \geq ps_{\eta-1}\} & \text{if the minimum exists,} \\ \beta(X) & \text{else if } ps_{i-1} \geq ps_{\beta(X)}, \\ \beta(X) + 1 \ (= n+1) & \text{otherwise.} \end{cases}$$

Similarly we denote $x_0 = 0$, and redefine lm_X to be:

$$\mathrm{lm}_X(i) = \begin{cases} \max\{\eta \in \gamma_+(X) \mid \eta < i,\ x_{\eta-1} \leq 0,\ ps_{\eta-1} < ps_{i-1}\} & \text{if the minimum exists,} \\ \alpha(X) - 1 \ (= 0) & \text{otherwise.} \end{cases}$$

The above definitions allow us only to compute right match or left match among the starting indexes of the monotone subsequences. Algorithm MAX_Sequential actually implements the above left match function. We use the redefined rm_X to implement the PRAM-algorithm in [DS06].

We can find $\mathrm{rm}_X(i)$ for a single $i \in \gamma_+(X)$ in linear time. But the brute force implementation to find all $\mathrm{rm}_X(i)$ for every $i \in \gamma_+(X)$ may take $O(n^2)$ time. There is a linear time algorithm to find all the right matches for each $i \in \gamma_+(X)$ using a stack. Similar approach can find all the left matches for each $i \in \gamma_+(X)$. Actually the algorithm to find all right matches and left matches for the starting indexes of maximal monotone subsequences can be combined with algorithm MAX_Sequential to re-use the same stack. The following algorithm is the modification of algorithm MAX_Sequential:

Algorithm 5: MAX_ANSV_Sequential

Input: A length-*n* real-valued sequence *X*.

Output: The sequence of all successive minimal maximum subsequences (that is, all maximal monotone subsequences) of X occupying the low-order subarray of an array $M[1..\lceil n/2 \rceil]$, and the related right match and left match for the starting index of each maximal monotone subsequence.

Data Structures: On consumed input X' (prefix of *X*):

1. Array $M[1..(i-1)]$: the canonical list $(Y_1, Y_2, \ldots, Y_{i-1})$ of $\mathrm{MAX}(X')$;
2. Array rm$[1..(i-1)]$: the right matches $\mathrm{rm}_{X'}(\alpha(Y_1))$, $\mathrm{rm}_{X'}(\alpha(Y_2))$, . . . , $\mathrm{rm}_{X'}(\alpha(Y_{i-1}))$;
3. Array lm$[1..(i-1)]$: the left matches $\mathrm{lm}_{X'}(\alpha(Y_1))$, $\mathrm{lm}_{X'}(\alpha(Y_2))$, . . . , $\mathrm{lm}_{X'}(\alpha(Y_{i-1}))$;
4. Stack *St*: the longest sublist $(Y_{(1)}, Y_{(2)}, \ldots)$ (which is not necessarily contiguous) of $\mathrm{MAX}(\mathrm{X}')$ such that

$\min\{ps_{\eta-1}(X')|\eta \in \gamma_+(X')\} = ps_{\alpha(Y_{(1)})-1}(X') < ps_{\alpha(Y_{(2)})-1}(X') < \ldots,$ and

$\max\{ps_{\eta}(X')|\eta \in \gamma_+(X')\} = ps_{\beta(Y_{(1)})}(X') \geq ps_{\beta(Y_{(2)})}(X') \geq \ldots;$

5. Subsequence Y_i: the incremental input of the longest (contiguous) subsequence of positive terms of X immediately succeeding X'.

Begin

1. Initialize:

   $\mathrm{M}[0] := \emptyset$; $\alpha(Y_{(0)}) := 0$; $\beta(Y_{(0)}) := 0$; $St := \emptyset$; $i := 1$; $\mathrm{rm}[i] := n+1$; $\mathrm{lm}[i] := 0$;

2. Compute $\gamma(Y_i)$, if non-empty:

   $\alpha(Y_i) := \min\{\eta \in \gamma_+(X') \mid \eta > \beta(Y_{i-1})\}$;

   $\beta(Y_i) := \min\{\eta \in \gamma_+(X') \mid \eta > \beta(Y_{i-1})$ and $\eta + 1 \notin \gamma_+(X')\}$;

   if $\alpha(Y_i)$ does not exist, then

   Output $M[1..(i-1)]$;

   while $St \neq \emptyset$

   $j :=$ Top(St);

   if $ps_{\alpha(Y_j)-1} \geq ps_{\beta(X)}$ then

   $\mathrm{rm}[j] := \beta(X)$;

   Pop(St);

   end if;

   end while;

   Stop;

   end if;

3. Absorb Y_i, update $\mathrm{rm}[1..(i-1)]$, $\mathrm{lm}[1..i]$, and, if necessary, update i, $M[1..(i-1)]$, and St:

3.1. while $i > 1$ and $St \neq \emptyset$

$j :=$ Top(*St*);

if $ps_{\alpha(Y_j)-1} \geq ps_{\alpha(Y_i)-1}$ then

rm$[j] := \alpha(Y_i)$;

Pop(*St*);

else if $ps_{\beta(Y_j)} < ps_{\beta(Y_i)}$ then

Pop(*St*); $\beta(Y_j) = \beta(Y_i)$; $i := j$;

else

lm$[i] := \alpha(Y_j)$;

goto Step 3.2;

end if;

end while;

3.2. Push i to *St*; $M[i] := Y_i$; $i := i + 1$; rm$[i] := n + 1$; lm$[i] := 0$; goto Step 2;

End

The stack *St* is used to store the Y_i that its right match is not found yet. In step 2, when the end of the sequence X is reached, the algorithm checks if the right match can be obtained at the end of X for Y_i on the stack. This is to verify if sequence X satisfies rm-closure condition.

The algorithm MAX_ANSV_Sequential processes input sequence X sequentially to find maximal subsequence Y_i, and for each Y_i, there are at most one Push and one Pop operation associated with it, so the algorithm MAX_ANSV_Sequential runs in $O(n)$ time.

5.3 Range Minima Query

Lemma 3 of chapter 2 shows that the ending index of the maximal monotone subsequence of X constrained with the starting index $i \in \gamma_+(X)$ is the index of the unique maximum prefix sum on $[i, \mathrm{rm}_X(i) - 1]$. Assume sequence X is partitioned into $(X_1, X_2, \ldots, X_m)$ on m processors, if a partition X_j for $1 \leq j < m$ is not rm_localized, then there is at least one index $i \in \gamma_+(X_j)$ that $\mathrm{rm}_X(i)$ cannot be found on hosting processor P_j. We assume $\mathrm{rm}_X(i)$ exists on some other processor P_k, where $1 \leq j < k \leq m$, we need to find the index of the maximum prefix sum efficiently on $[i, \mathrm{rm}_X(i) - 1]$. Notice that it is the largest one among three index subranges: $[j, \beta(X_j)]$, $[\alpha(X_{j+1}), \beta(X_{k-1})]$, and $[\alpha(X_k), \mathrm{rm}_X(i) - 1]$.

First for index subrange $[j, \beta(X_j)]$: Notice that $\mathrm{rm}_X(i)$ is on a different processor, i.e., $\mathrm{rm}_{X_j}(i) = \beta(X_j) + 1$, so the maximum prefix sum on $[i, \beta(X_j)]$ is the one at the ending index of the maximal monotone subsequence in MAX(X_j) with starting index i.

Second for index subrange $[\alpha(X_k), \mathrm{rm}_X(i) - 1]$: We can use an array to store the maximum prefix sum between $[\alpha(X_k), \eta]$ for any $\eta \in [\alpha(X_k), \beta(X_k)]$. The creation of the array is in linear time, and the time to find the maximum prefix sum between $[\alpha(X_k), \eta]$ is the constant time to query the array.

Third for $[\alpha(X_{j+1}), \beta(X_{k-1})]$: This can be reduced to range minima query (RMQ) problem. Given an array $MaxP[1..m]$ of the maximum prefix sum of each partition X_η for $\eta \in [1, m]$, processor index j and k for $1 \leq j \leq k \leq m$, RMQ(j, k) outputs the maximum prefix sum in subarray $MaxP[j..k]$. Since the algorithm of RMQ will be invoked by every maximal monotone subsequence that could not find the right match of its starting index locally on the hosting processor, we expect the constant query time of the algorithm.

RMQ has a native solution in $O(m^2)$ time to build a table storing answers to all of the m^2 possible queries. We can improve the time complexity by using dynamic programming. The following algorithm Build_ST and algorithm Query_ST implement the sparse table algorithm in Bender's paper [BF00].

Algorithm 7: Build_ST

Input: An array $MaxP[1..m]$ containing the maximum prefix sums of m partitions of the input sequence X.

Output: An sparse table $ST[1..m][1..\lfloor \log_2 m \rfloor]$. For $i \in [1..m]$ and $j \in [1..\lfloor \log_2 m \rfloor]$, $ST[i][j]$ find the maximum element in the subarray $MaxP[i..i+2^j-1]$ if $i+2^j-1 \leq m$. The query of subarray $MaxP[i..i+2^j-1]$ is to be solved by finding the larger one between two recursive small queries on the two halves of the subarray.

Begin

1. Initialize ST starting at $i \in [1..m]$ with length 1:

   for $i := 1$ to m

   $ST[i][0] := i$;

   end for;

2. Create ST table by using dynamic programming. The maximum element in subarray $MaxP[i..i+2^j-1]$ is the larger one between two maxima of subarray $MaxP[i..i+2^{j-1}-1]$ and $MaxP[i+2^{j-1}..i+2^j-1]$.

   for $j := 1$ to $\lfloor \log_2 m \rfloor$

   for $i := 1$ to $m - 2^j + 1$

   if $MaxP\left[ST[i][j-1]\right] \geq MaxP\left[ST\left[i+2^{j-1}\right][j-1]\right]$ then

   $ST[i][j] := ST[i][j-1]$;

   else

$ST[i][j] := ST\left[i + 2^{j-1}\right][j-1];$

end if;

end for;

end for;

End

Algorithm 8: Query_ST

Input: An array $MaxP[1..m]$ containing the maximum prefix sums of m partitions of the input sequence X, index i and j such that $1 \leq i \leq j \leq m$. A sparse table $ST[1..m][1..\lfloor \log_2 m \rfloor]$ built by algorithm Build_ST.

Output: The index of the maximum element in subarray $MaxP[i..j]$.

Begin

1. Query subarray with length 1:

   if $i == j$ then

   return i;

   end if;

2. Select two overlapping subarrays that cover the subarray $MaxP[i..j]$. The maximum elements of the two subarrays can be queried from ST table.

   $k := \lfloor \log_2(j-i) \rfloor$;

   if $MaxP\left[ST[i][k]\right] \geq MaxP\left[ST[j-2^k+1][k]\right]$ then

   return $ST[i][k]$;

   else

   return $ST[j-2^k+1][k]$;

   end if;

End

The algorithm Build_ST runs in $O(m \log m)$ time, and Query_ST runs in constant time. The algorithms are appropriate if $m \log m$ is much less than the size of the partition of input sequence X. Otherwise the algorithm that builds the Cartesian Tree of $MaxP$ in linear time and queries in constant time [BF00] should be used.

5.4 Parallel Algorithm to Find MAX on Cluster Systems

Based on the above stated sequential algorithms, we build the parallel algorithm to find MAX on cluster systems with MPI.

Algorithm 9: MAX_Parallel

Input: A length-*n* real-valued sequence *X* (which is a random sample satisfying the assumptions in Theorem 13: part 2) and a prescribed probability threshold δ (Remark 8: Chebyshev's inequality).

Output: The sequence of all successive minimal maximum subsequences (that is, all maximal monotone subsequences) of *X*.

Begin

1. Construct sequential partition $\mathcal{P}(X) = \left(X_1, X_{1,2}, X_2, X_{2,3}, X_3, \ldots, X_{m-1}, X_{m-1,m}, X_m\right)$ of X such that: (1) for all $i \in \{1, 2, \ldots, m\}$, processor P_i hosts the subsequence $X_{i,i-1}X_iX_{i,i+1}$ in length-balanced manner except possibly for the last processor P_m, and (2) for all $i \in \{1, 2, \ldots, m\}$, $\left|X_{i,i+1}\right|$is the least upper bound of $\left\lceil \mathrm{E}(T) + \delta\sqrt{\mathrm{Var}(T)} \right\rceil$ computed via to Theorem 13: part 2;
2. Decide if $\mathcal{P}(X)$ is an rm-localized partition:

   2.1. for all $i \in \{1, 2, \ldots, m-1\}$, processor P_i computes:

$$is\_\mathrm{rm}Localized_i \coloneqq \left(\gamma_+\left(X_{i,i-1}X_i\right) == \emptyset\right) \vee$$

$$\left(\mathrm{rm}_{X_{i,i-1}X_iX_{i,i+1}}(\beta_i^*) \in \left[\beta_i^* + 1, \beta(X_{i,i-1}X_iX_{i,i+1})\right]\right);$$

   end for;

   processor P_m computes:$is\_\mathrm{rm}Localized_m \coloneqq$ true;

2.2. Compute $is\_\mathrm{rm}Localized := \wedge_{\eta=1}^{m} is\_\mathrm{rm}Localized_{\eta}$ using MPI Allreduce (prefix sum) function;

2.3. for all $i \in \{1, 2, \ldots, m-1\}$ processor P_i updates:

$is\_\mathrm{rm}Localized_i := is\_\mathrm{rm}Localized$;

end for;

3. If $\mathcal{P}(X)$ is rm-localized, then compute MAX(X) via Theorem 7: determine $X'_{i,i+1}$ for all $i \in \{1, 2, \ldots, m-1\}$ and compute MAX($X''_{i-1,i} X_i X'_{i,i+1}$ for all $i \in \{1, 2, \ldots, m\}$:

for all $i \in \{1, 2, \ldots, m\}$ processor P_i decides:

if $is\_\mathrm{rm}Localized_i$ then

if $i < m$ then

processor P_i sends $\mathrm{rm}_{X_{i,i-1} X_i X_{i,i+1}}(\beta_i^*)$ to processor P_{i+1};

processor P_{i+1} receives $\mathrm{rm}_{X_{i,i-1} X_i X_{i,i+1}}(\beta_i^*)$;

end if;

Invokes MAX_ANSV_Sequential to compute MAX($X''_{i-1,i} X_i X'_{i,i+1}$);

else

goto Step 4;

end if;

end for;

4. Let $|X_{i,i+1}| = 0$ for $i \in [1, m-1]$. Invoke a parallel algorithm adapted from the MAX-computing PRAM-algorithm [DS06] in which two embedded problems are solved by the parallel algorithm for "all nearest smaller values" [HH01] and RMQ algorithm;

4.1. for all $i \in \{1, 2, \ldots, m\}$ processor P_i

Invokes MAX_ANSV_Sequential to compute MAX(X_i);

end for;

4.2. for all $i \in \{1, 2, \ldots, m\}$ processor P_i

Create array $MaxP[1..m]$ to store the maximum prefix sum of each processor by MPI Allgather;

Invokes Create_ST;

end for;

4.3. for all $i \in \{1, 2, \ldots, m\}$ processor P_i

Invokes parallel algorithm to locate all the right matches in X for every starting index η such that $rm_{X_i}(\eta) = \beta(X_i) + 1$ of the subsequences in MAX(X_i) in 4.1;

Find the index of the maximum prefix sum between above index η and $rm_X(\eta) - 1$;

end for

End;

The parallel ANSV algorithm in [HH01] runs in $O(n/m + m)$. The RMQ algorithm runs in $O(m \log m)$. The complexity of MPI Allreduce is $O(\log m)$, while that of Allgather is $O(m + \log m)$. Therefore the algorithm MAX_Parallel runs in $O(n/m + m \log m)$.

5.5 Experiments

We implement the MAX_Parallel algorithm on the Cowboy cluster of High Performance Computing Center at Oklahoma State University. The cluster has 252 standard compute nodes, each with dual Intel Xeon E5-2620 “Sandy Bridge” hex core 2.0 GHz CPUs. The implementation is in C language with OpenMPI 1.4.

The experiments are designed with the following objectives:

1. Verify the speedup and efficiency of the algorithm with different number of processors.
2. Verify the bound of the common subsequence is appropriate.
3. Verify the time complexity of the algorithm with different size of data.

4. Study the behaviors between different mean values of the common probability distribution.
5. Study the effects of the different bound sizes with different δ values.

We first test the base case of 5M random data from a normal distribution with mean -0.25 and variance 1.0, and $\delta = 3$. The size of the common subsequence is the upper bound on $E(T) + \delta\sqrt{\mathrm{Var}(T)}$ where T is the conditional first weak descending epoch. The independent and identically distributed random sample data with the common normal distribution are generated through Box-Muller transformation [BM58]. We use the following approximate formula by Bryc [Brc00] to calculate the tail probability:

$$\mathrm{prob}(Z > z) = \frac{z^2 + 5.575192695z + 12.77436324}{\sqrt{2\pi}z^3 + 14.38718147z^2 + 31.53531977z + 2 * 12.77436324} e^{-\frac{z^2}{2}}.$$

The base case has been run with $N = 100$ trial-sequences, and each one has 5M data and from the same normal distribution with mean -0.25 and variance 1.0. The performance measures in (absolute) speedup and efficiency of MAX_Parallel algorithm are collected in two sets of mean-statistics:

1. The set of conditional mean-statistics on "success" scenario (satisfiability of the rm-locality condition for the first $(m-1)$ processors) from *N* trial-sequences and the MAX-computing by (local) MAX_Sequential in Steps 1 – 3 of MAX_Parallel.

2. The set of unconditional ones for MAX_Parallel with all steps.

Based on the optimal sequential-time algorithm [RT99], the mean optimal sequential time for MAX-computation of a length-*n* sequence, $T^*(n)$, is approximately 0.155881 sec for the 5M synthetic random data (when averaged over $N = 100$ sequences).

Table 5.1 summarizes the above-stated two sets of mean-statistics of the running time $T_m(n)$, speedup $S_m(n) = \frac{T^*(n)}{T_m(n)}$, and efficiency $E_m(n) = \frac{T_1(n)}{mT_m(n)}$ of MAX_Parallel for $\delta = 3$ and m processors with $m \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. Figure 5.2 illustrates the conditional and unconditional speedups, and Figure 5.3 illustrates the conditional and unconditional efficiencies.

			Conditional mean-statistics over observed-N_s			Unconditional mean-statistics over N		
m	N_s	Observed N_s	$T_m(n)$	$S_m(n)$	$E_m(n)$	$T_m(n)$	$S_m(n)$	$E_m(n)$
1	100	100	0.159702	0.991785	1.000000	0.159704	0.991772	1.000000
2	88.88889	100	0.079643	1.988750	1.002612	0.079650	1.988575	1.002536
4	70.2332	97	0.041431	3.822983	0.963662	0.041747	3.794045	0.956380
8	43.84624	91	0.020898	7.579194	0.955247	0.021475	7.375553	0.929593
16	17.08882	88	0.010599	14.943863	0.941728	0.011089	14.283524	0.900126
32	2.595803	78	0.006784	23.347583	0.735656	0.007769	20.387437	0.642393
64	0.059895	67	0.002665	59.433396	0.936339	0.003670	43.158038	0.679939
128	3.19E-05	41	0.001864	84.973176	0.669352	0.005376	29.462426	0.232085

Table 5.1: Mean statistics for 5M data with N(-0.25,1) and $\delta = 3$

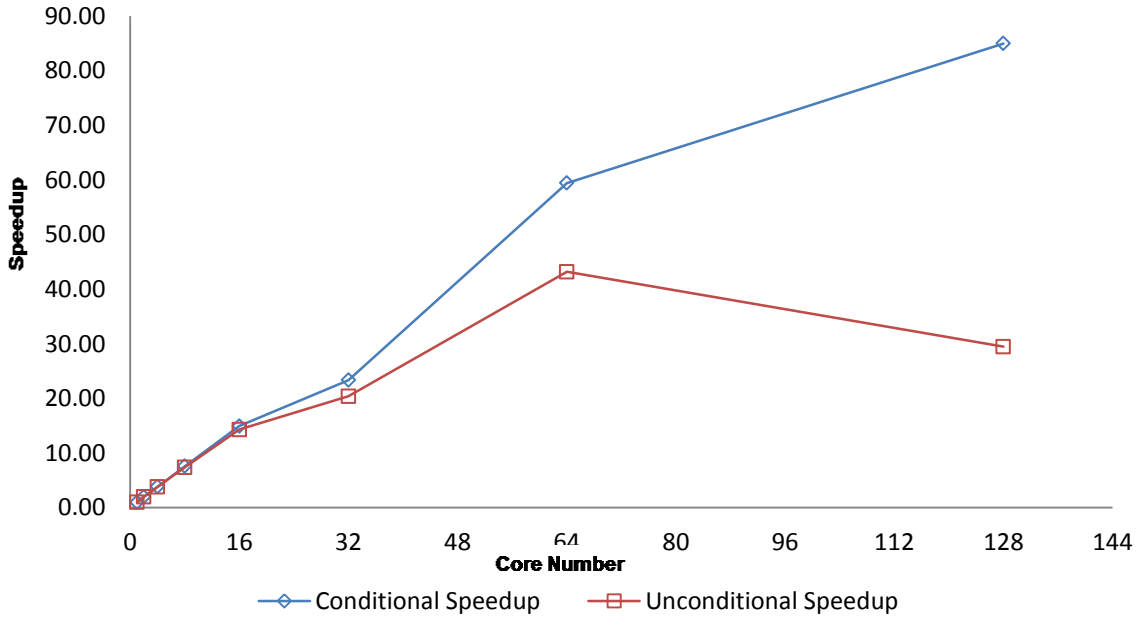

Figure 5.2: Speedup for 5M data with N(-0.25,1) and $\delta = 3$

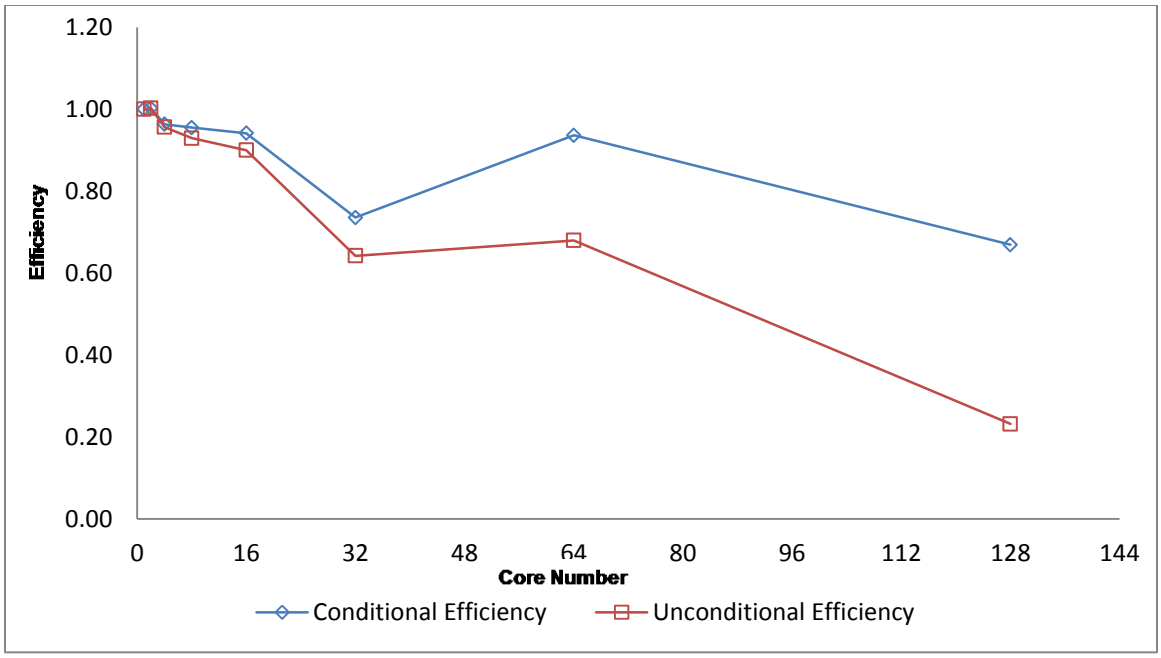

Figure 5.3: Efficiency for 5M data with N(-0.25,1) and $\delta = 3$

Since probability Prob(satisfiability of rm-locality for single processor)$\geq 1 - \frac{1}{\delta^2}$, the expected number N_s of "successes" from N trial-sequences is bounded below:

$$N_s \geq N\left(1 - \frac{1}{\delta^2}\right)^{m-1}.$$

The empirical and statistical results tabulated in the two columns: (expected) N_s and observed-N_s show that the constraints on $\mathrm{E}(T)$ and $\mathrm{Var}(T)$ in bounding $E(T) + 3\sqrt{\mathrm{Var}(T)}$ serves as a good lower-bound predictor for N_s.

The calculated upper bounds on $\mathrm{E}(T)$ and $\sqrt{\mathrm{Var}(T)}$ are 7.981282 and 22.891637 respectively, thus the size of the common subsequence is 76. The average weak descending epoch is 3.270820, and the average conditional weak descending epoch is 6.658881. The upper bound on $\mathrm{E}(T)$ matches the observation.

For the conditional statistics on "success" scenario, the speedup and efficiency are close to their theoretical bounds of m and 1 respectively, except for $m = 128$. For the unconditional ones, even for a small $\delta = 3$, the speedup and efficiency are about 2/3 of their theoretical bounds, except for $m = 128$. The conditional speedup and efficiency for $m = 128$ is quite off. The main reason is because the computation time is so small that the communication time and system overhead play bigger roles. We expect better speedup and efficiency at $m = 128$ for larger data size.

Second, we run the base case with 5M, 10M, and 20M data respectively on m processors with $m \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. The common normal distribution and $\delta = 3$ are kept the same.

The following Figure 5.4 shows that for larger data size, the unconditional speedup is better. The same improvement can also be found for unconditional efficiency. For example, the efficiencies for 128 processors are 0.232085, 0.430357, and 0.569211 respectively for 5M, 10M, and 20M data.

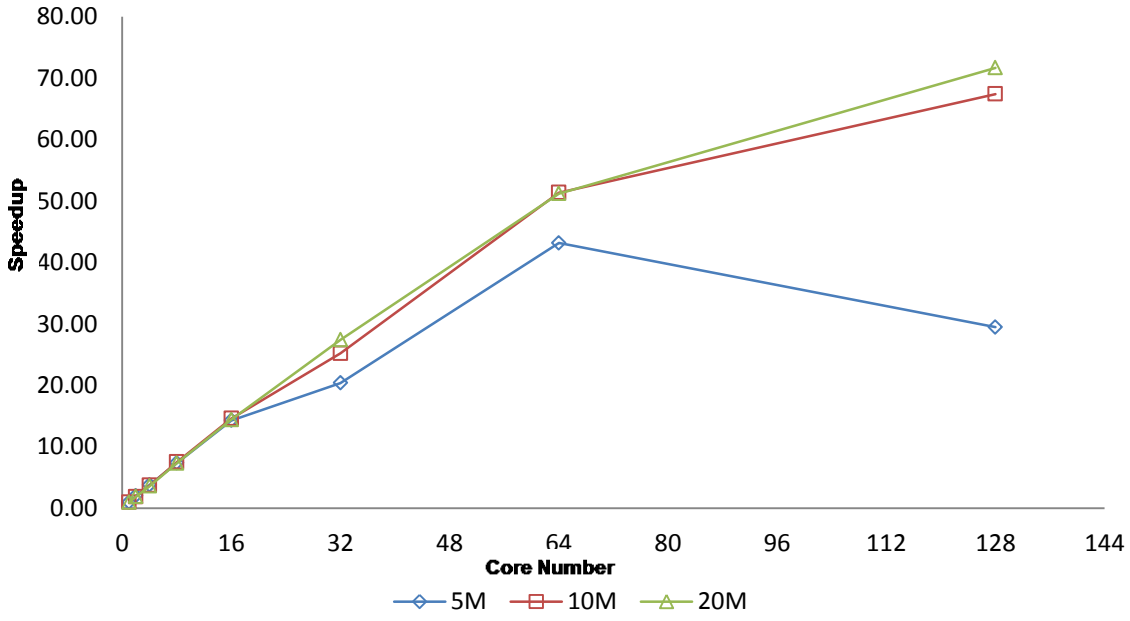

Figure 5.4: Unconditional speedups for data with N(-0.25,1) and $\delta = 3$

			mean-statistics over observed-N_s			mean-statistics over N		
m	N_s	observed-N_s	$T_m(n)$	$S_m(n)$	$E_m(n)$	$T_m(n)$	$S_m(n)$	$E_m(n)$
1	100	100	0.158392	0.9906372	1	0.158392	0.990637	1
2	93.75	99	0.080535	1.948333	0.983374	0.08073	1.943627	0.980998
4	82.39746	99	0.041374	3.7924542	0.957074	0.041468	3.783857	0.954905
8	63.65008	95	0.020995	7.4736366	0.943034	0.021292	7.369388	0.92988
16	37.98124	93	0.010616	14.780426	0.932508	0.010946	14.33483	0.904394
32	13.52414	94	0.005095	30.796663	0.971492	0.006581	23.84273	0.752127
64	1.714709	89	0.002878	54.520153	0.859929	0.003997	39.25669	0.619183
128	0.027565	75	0.002009	78.103036	0.615947	0.005283	29.70074	0.23423

Table 5.2: Mean statistics for 5M data with N(-0.25,1) and $\delta = 4$

			mean-statistics over observed-N_s			mean-statistics over N		
M	N_s	observed-N_s	$T_m(n)$	$S_m(n)$	$E_m(n)$	$T_m(n)$	$S_m(n)$	$E_m(n)$
1	100	100	0.180848	0.9550009	1	0.18085	0.95499	1
2	88.88889	100	0.08944	1.9310152	1.011002	0.089447	1.930864	1.010934
4	70.2332	95	0.045532	3.7931565	0.992972	0.046066	3.749186	0.981472
8	43.84624	84	0.023394	7.3826622	0.966316	0.024304	7.106238	0.930145
16	17.08882	61	0.011933	14.473309	0.947205	0.013276	13.00919	0.851395
32	2.595803	51	0.005866	29.44255	0.963433	0.007274	23.74347	0.776954
64	0.059895	19	0.002904	59.47314	0.973054	0.004905	35.21101	0.576102
128	3.19E-05	8	0.001485	116.30303	0.951431	0.006665	25.91298	0.211987

Table 5.3: Mean statistics for 5M data with N(-0.125,1) and $\delta = 3$

			mean-statistics over observed-N_s			mean-statistics over N		
M	N_s	observed-N_s	$T_m(n)$	$S_m(n)$	$E_m(n)$	$T_m(n)$	$S_m(n)$	$E_m(n)$
1	100	100	0.17962	0.960383	1	0.179622	0.960372	1
2	93.75	99	0.091237	1.8907242	0.984359	0.09141	1.887146	0.982507
4	82.39746	97	0.045639	3.7797498	0.983917	0.045954	3.753841	0.977184
8	63.65008	95	0.0235	7.3405957	0.955426	0.023988	7.191262	0.935999
16	37.98124	91	0.011923	14.468171	0.941563	0.012352	13.96567	0.908871
32	13.52414	69	0.00644	26.786335	0.871603	0.007541	22.87548	0.744356
64	1.714709	52	0.002943	58.615019	0.95364	0.004177	41.29854	0.671916
128	0.027565	32	0.001483	116.32097	0.946245	0.00507	34.02446	0.276784

Table 5.4: Mean statistics for 5M data with N(-0.125,1) and $\delta = 4$

Then, we run 5M random data with mean -0.25 and -0.125, also $\delta = 3$ and 4 respectively. The results are listed in above Table 5.1, 5.2, 5.3 and 5.4 respectively.

Figure 5.5 and 5.6 show the conditional speedup and unconditional speedup respectively for 5M data. For 5M data, the runtime and speedups change slightly when δ changes from 3 to 4, which may indicate that $\delta = 3$ is large enough to generate a good upper bounds on the size of the common subsequence. Notice that the observed-N_s is increasing when using larger δ, so the unconditional average run time should improve also. But on the other hand, larger δ means longer common subsequence, so the computation time may increase slightly.

Also there are not much changes for runtime and speedups when the mean of common normal distribution changes from -0.25 to -0.125 for 5M data. We also notice that the calculated bound on $\mathrm{E}(T)$ changes accordingly. For the case of $\delta = 3$, the calculated upper bound changes from 7.981282 to 15.281384, and the observed average conditional weak descending epoch changes from 6.658881 to 12.296764. The calculated bound on $\mathrm{E}(T)$ is appropriate.

Similar speedup results are found for 10M and 20M data when changing the δ and the mean of common normal distribution. Figure 5.7 and 5.8 are conditional and unconditional speedups for 10M data, while Figure 5.9 and 5.10 are those for 20M data. From the charts we notice the better speedups for larger data sets.

Except for 5M random data with mean -0.25 ($\delta = 3$ and 4), conditional efficiencies are over 0.9 for all other cases. The following Figure 5.11, 5.12, and 5.13 are unconditional efficiencies for 5M, 10M, and 20M data respectively. We only observed small changes of efficiencies between $\delta = 3$ and $\delta = 4$, or between mean -0.25 and -0.125 if other parameters are the same. The better efficiencies are displayed for larger data sets.

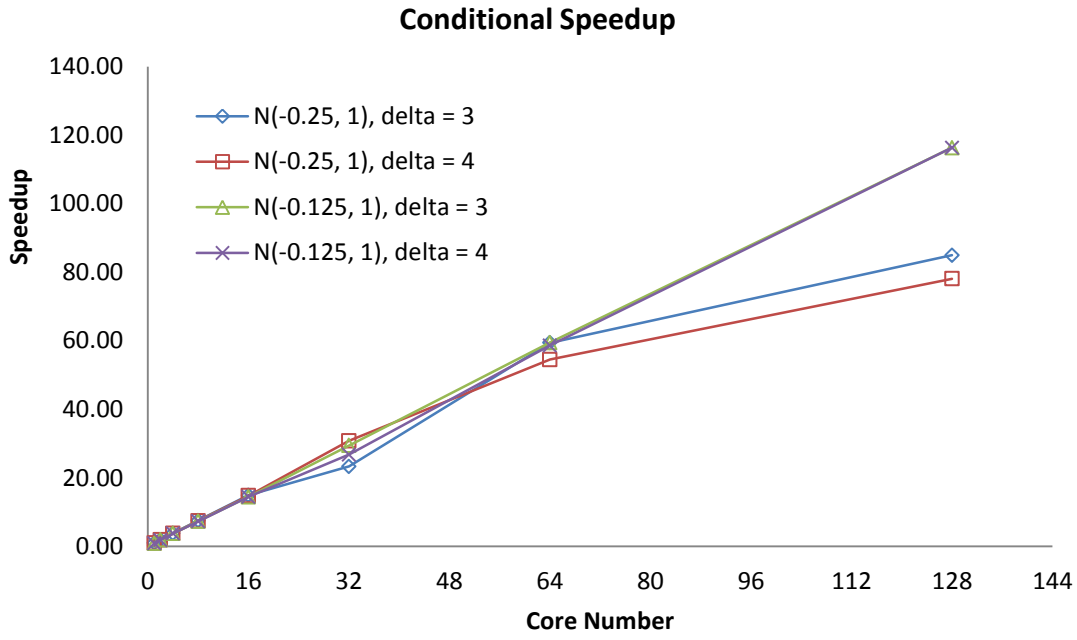

Figure 5.5: Conditional speedups for 5M data

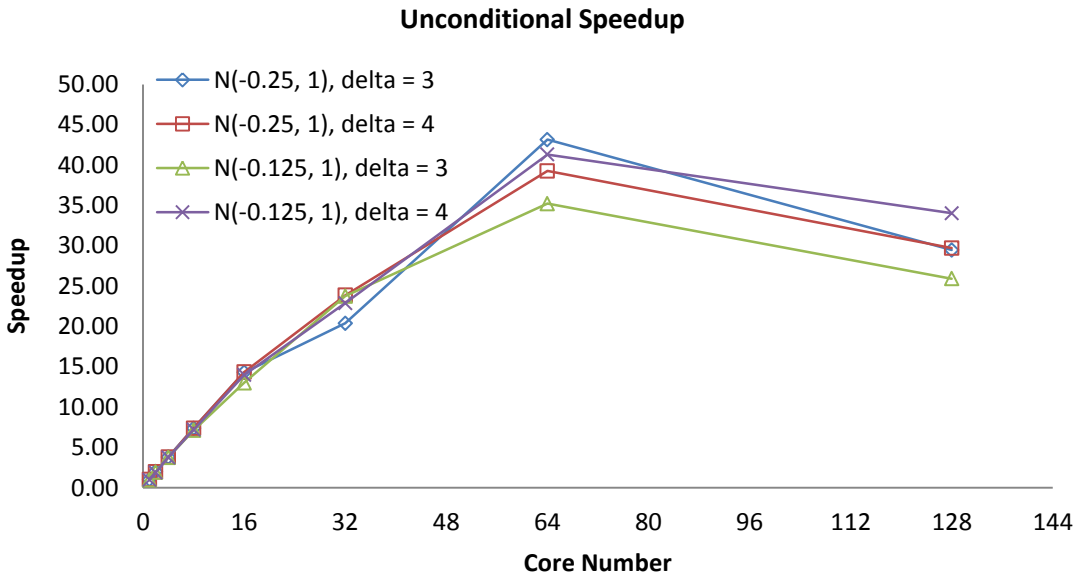

Figure 5.6: Unconditional speedups for 5M data

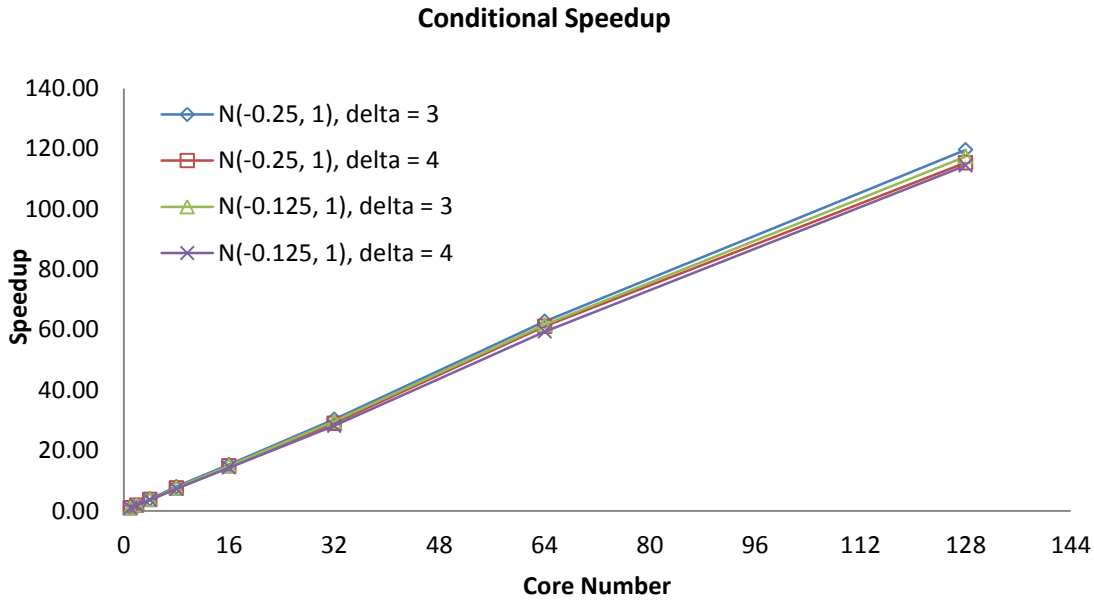

Figure 5.7: Conditional speedups for 10M data

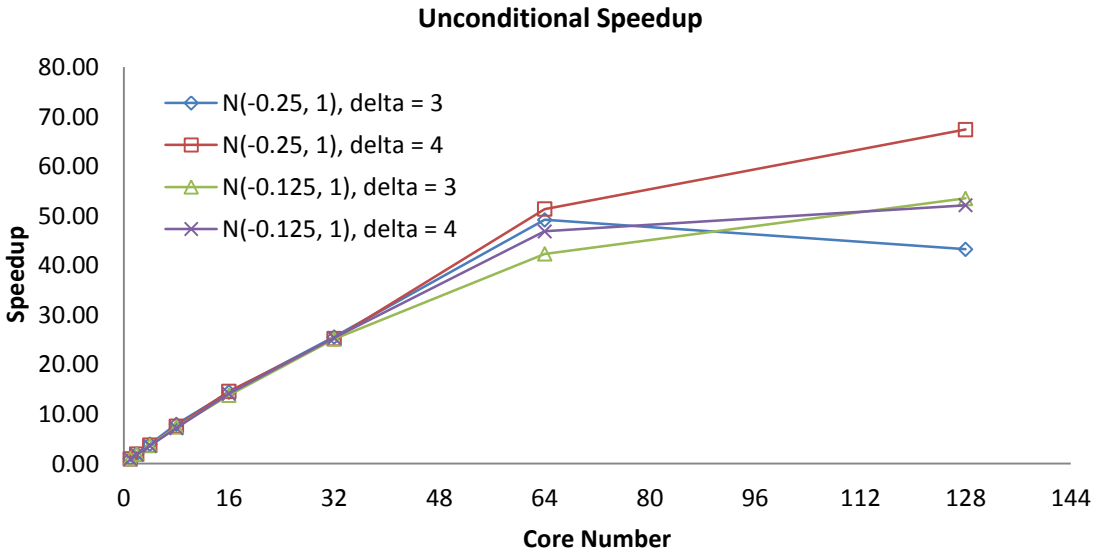

Figure 5.8: Unconditional speedups for 10M data

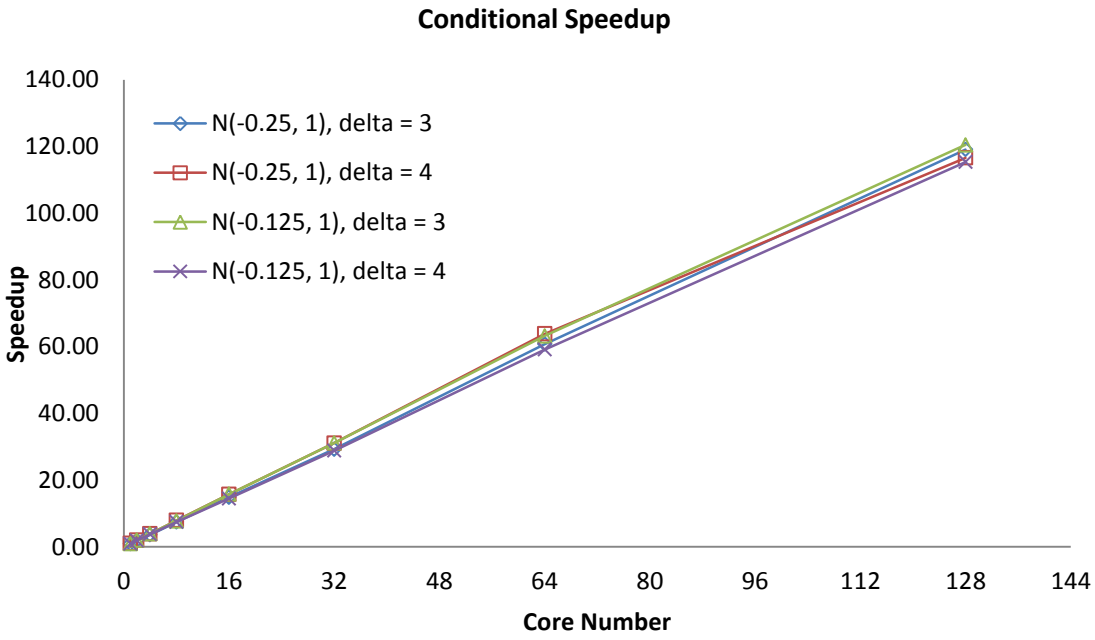

Figure 5.9: Conditional speedups for 20M data

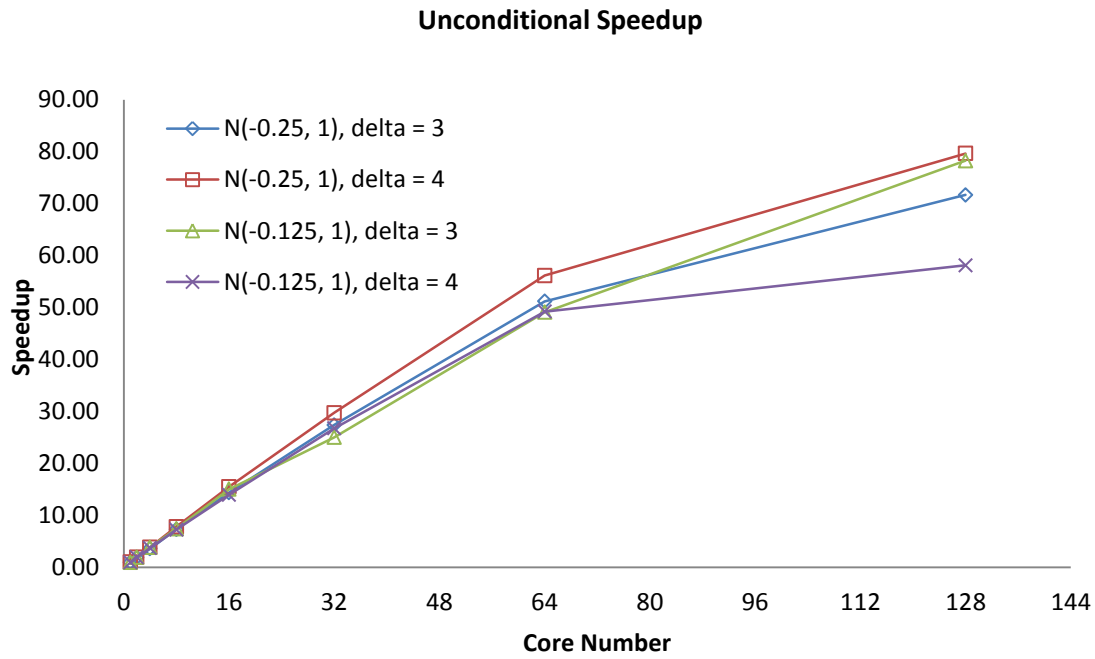

Figure 5.10: Unconditional speedups for 20M data

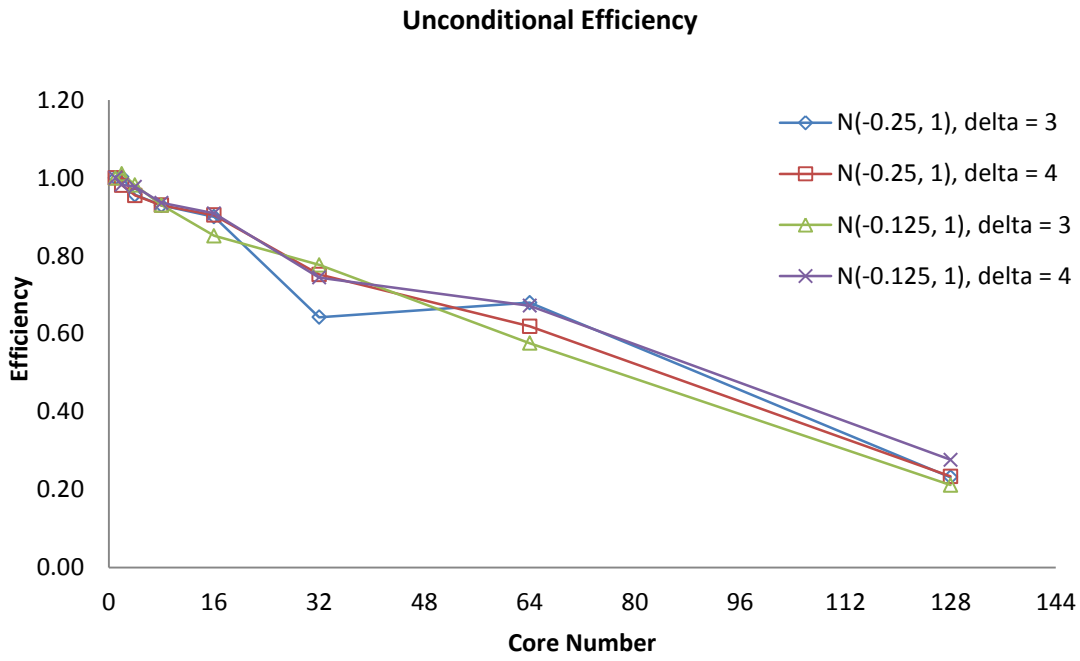

Figure 5.11: Unconditional efficiencies for 5M data

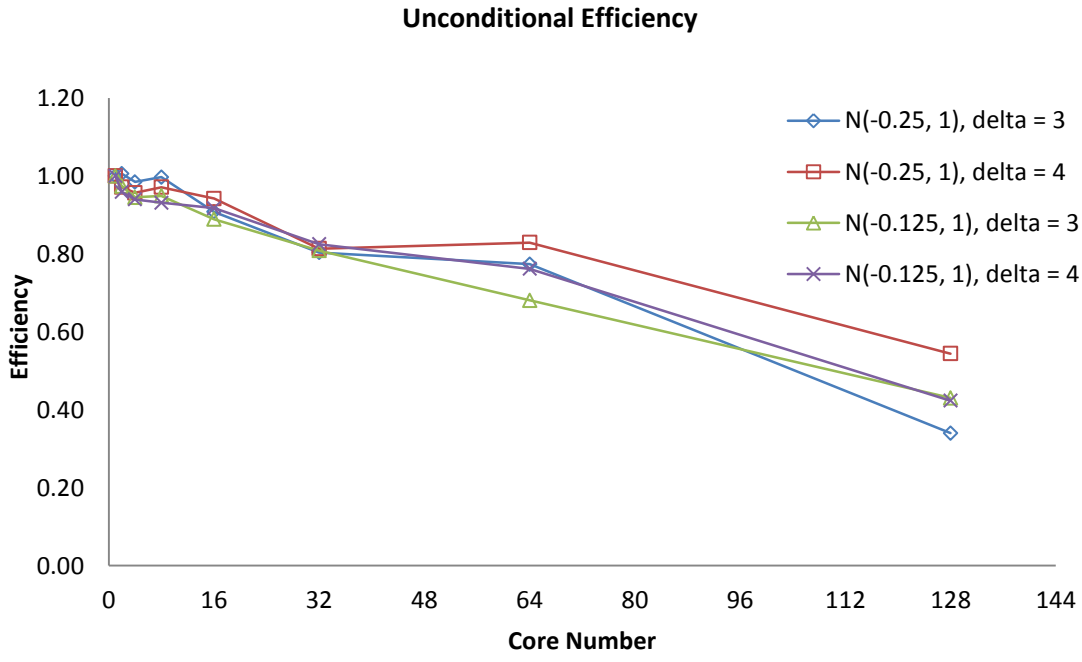

Figure 5.12: Unconditional efficiencies for 10M data

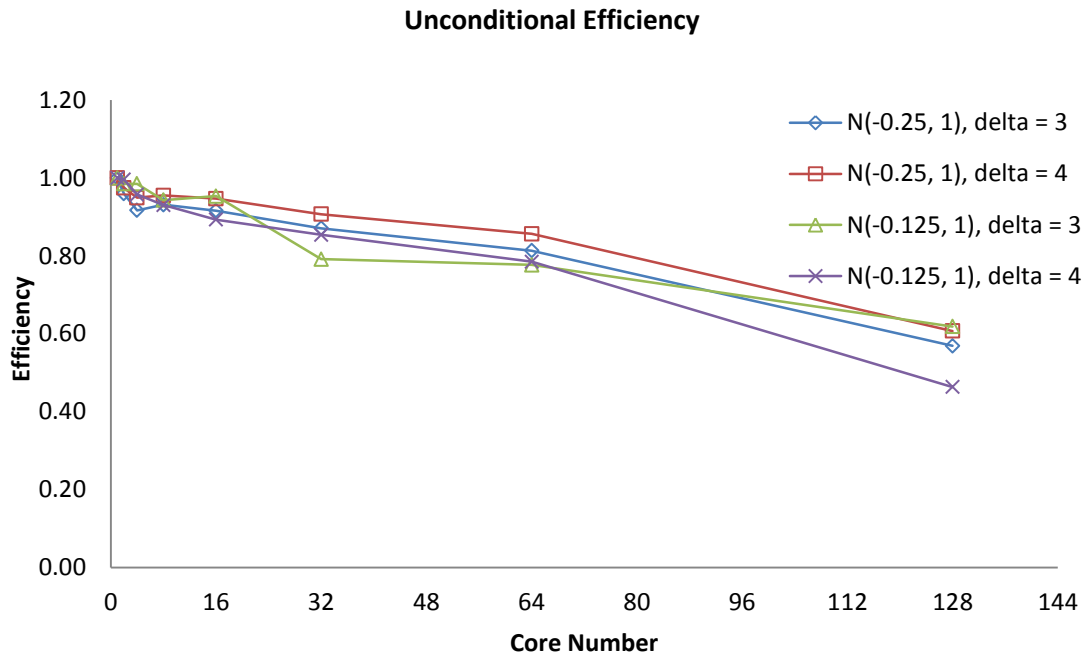

Figure 5.13: Unconditional efficiencies for 20M data

CHAPTER VI

CONCLUSIONS

6.1. Conclusions

The problem of computing the set of all minimal maximum subsequences of a real-valued sequence has major applications such as in bioinformatics, pattern matching, and data mining. The MAX-computation has real practical importance as it appears as a subroutine in biological sequence analysis. Hence there is a natural need for computing MAX in parallel and its implementation on practical parallel systems.

Ruzzo and Tompa presented linear-time sequential algorithm for computing MAX [RT99]. The main purpose of our research is to develop the parallel algorithm on cluster systems with subsequence-hosting processors employing the optimal sequential algorithm computing MAX. We prove that if the structural decomposition of a non-empty real-valued sequence satisfies the rm-locality condition, then the decomposition is MAX-independent. We propose a length-balanced and MAX-independent sequential partition into multiple processors on cluster systems with the common subsequence hosted by every two successive processors.

We analyze the length bound of the common subsequences probabilistically for random sequences of independent and identically distributed random variables with common normal

distribution via the theory of random walk. The upper bounds on the expectation and variance of the conditional first weak descending ladder epoch are derived, which produces the upper bound on the length of the common subsequence in accordance with Chebyshev's inequality.

We design and implement the domain-decomposed parallel algorithm to find MAX on the cluster with MPI. Depends on if the partition is rm-localized or not, linear-time sequential algorithm or parallel algorithm adapted from PRAM MAX-computing algorithm [DS06] will be invoked. Several optimizations have been applied to our parallel algorithm to minimize the local computations and the communications among processors.

We have done an empirical study of the speedup and efficiency achieved by the parallel algorithm with synthetic random data on the cluster at Oklahoma State University. Multiple data sets are created from independent and identically distributed random variables with common normal distribution. The test results from different numbers of processors, different data sizes, different mean values of normal distribution, and different δ values show the good overall runtime speedup and efficiency.

6.2. Future Work

There are two directions for general theoretical developments. First, the length bound of the common subsequences (to capture the rm-locality) is achieved via explicit bounds on the mean and variance of the first ladder epoch in the underlying random walk with normal distribution. This leads to a deserving study for general probability distribution. Second, there are other notions of (minimal) maximality for ranking subsequences of a real-valued sequence [BH06], developing efficient parallel algorithms for their computation is interesting.

REFERENCES

[ACS03] C. E. R. Alves, E. N. Caceres, and S. W. Song, "Computing Maximum Subsequence in Parallel," in *Proceedings II Brazilian Workshop on Bioinformatics - WOB 2003*, Macaé, RJ, Brazil, Dec. 3-5, 2003, pp. 80-87, 2003.

[ACS13] C. E. R. Alves, E. N. Caceres, S. W. Song, "Finding All Maximal Contiguous Subsequences of a Sequence of Numbers in $O(1)$ Communication Rounds," *IEEE Transactions on Parallel .& Distributed Systems*, vol. 24, no. 4, pp. 724-733, April 2013.

[AG91] S. G. Akl and G. R. Guenther, "Application of broadcasting with selective reduction to the maximal sum subsegment problem," *International Journal of High Speed Computing*, vol. 3, no. 2, pp. 107-119, 1991.

[Ale06] A. K. Aleškevičienė. "On Calculation of Moments of Ladder Heights," *Lithuanian Mathematical Journal*, vol. 46, no. 2, pp.12-145, 2006.

[Asm03] S. Asmussen. *Applied Probability and Queues*, *Second Edition*, New York, Springer, 2003.

[Bae07] S. E. Bae, "*Sequential and parallel algorithms for the generalized maximum subarray problem*," Ph.D. Dissertation, University of Canterbury, 2007.

[BC06] J. L. Bates and R. L. Constable, "Proofs as programs," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 1, pp. 113-136, January 1985.

[BF00] M. A. Bender and M. Farach-Colton, "The LCA problem revisited," in *LATIN 2000: Theoretical Informatics*, pp. 88-94, Springer Berlin Heidelberg, 2000.

[BBG+89] O. Berkman, D. Breslauer, Z. Galil, B. Schieber, and U. Vishkin, "Highly parallelizable problems," in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pp. 309–319, Association for Computing Machinery, 1989.

[BBN+92] V. Brendel, P. Bucher, I. R. Nourbakhsh, B. E. Blaisdell, and S. Karlin, "Methods and algorithms for statistical analysis of protein sequences," in *Proceedings of the National Academy of Sciences of USA*, vol. 89, no. 6, pp. 2002-2006, 1992.

[Ben00] J. Bentley, *Programming Pearls*, *Second Edition*. Addison-Wesley, 2000.

[BH06] T. Bernholt and T. Hofmeister, "An algorithm for a generalized maximum subsequence problem," in *LATIN 2006*: *the 7th Latin American Theoretical Informatics Symposium*, Valdivia, Chile, March 20-24, 2006, pp. 178–189, Springer-Verlag, Berlin Heidelberg, 2006.

[BM58] G. E. P. Box and M. E. Muller, "A Note on the Generation of Random Normal Deviates," *Ann. Math. Stat.* no. 29, pp. 610-611, 1958.

[Brc02] W. Bryc, "A uniform approximation to the right normal tail integral," *Applied mathematics and computation*, vol. 127, no. 2, pp. 365-374, 2002

[CD02] M. Chiani and D. Dardari, "Improved exponential bounds and approximation for the *Q*-function with application to average error probability computation," *Global Telecommunications Conference*, vol. 2, pp. 1399-1402, 2002.

[Che96] D. Z. Chen, "Efficient geometric algorithms on the EREW PRAM," *IEEE Transactions on Parallel and Distributed Systems*," vol. 6, no. 1, pp. 41–47, 1995.

[Cra91] J. W. Craig, "A new, simple, and exact result for calculating the probability of error for two-dimensional signal constellations," In *Proceedings of the 1991 IEEE Military Communications Conference*, vol. 2, pp. 571–575. IEEE, October 1991.

[DS06] H. K. Dai and H. C. Su, "A parallel algorithm for finding all successive minimal maximum subsequences," in *LATIN 2006*: *the 7th Latin American Theoretical Informatics Symposium*, Valdivia, Chile, March 20-24, 2006, pp. 337-348, Springer-Verlag, Berlin Heidelberg, 2006.

[Fel71] W. Feller, *An Introduction to Probability Theory and its Application*, *Vol. II; Second Edition*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1971.

[Gus97] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, New York, 1997.

[HH01] X. He and C. H. Huang, "Communication efficient BSP algorithm for all nearest smaller values problem," *Journal of Parallel and Distributed Computing*, vol. , no. 61, pp. 1425-1438, 2001.

[JáJ92] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, 1992.

[KA90] S. Karlin and S. F. Altschul, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes," in *Proceedings of the National Academy of Sciences U. S. A.*, vol. 87, no. 6, pp. 2264-2268, 1990.

[KA93] S. Karlin and S. F. Altschul, "Applications and statistics for multiple high-scoring segments in molecular sequences," in *Proceedings of the National Academy of Sciences U. S. A.*, vol. 90, no. 12, pp. 5873-5877, 1993.

[KBB91] S. Karlin, P. Bucher, V. Brendel. and S. F. Altschul, "Statistical-methods and insights for protein and DNA-sequences," *Annual Review of Biophysics and Biophysical Chemistry*, vol. 20, no. 1, pp. 175–203, 1991.

[KB92] S. Karlin and V. Brendel, "Chance and statistical significance in protein and DNA sequence analysis," *Science*, vol. 257, no. 5066, pp.39–49, 1992.

[KD92] S. Karlin and A. Dembo, "Limit distributions of maximal segmental score among Markov-dependent partial sums," *Advances in Applied Probability*, vol. 24, no. 1, pp. 113–140, 1992.

[LF80] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *Journal of the Association for Computing Machinery*, vol. 27, no. 4, pp. 831–838, 1980.

[Man89] U. Manber, *Introduction to Algorithms: A Creative Approach*, Addison-Wesley, 1989.

[PD95] K. Perumalla and N. Deo, "Parallel algorithms for maximum subsequence and maximum subarray," *Parallel Processing Letters*, vol. 5, no. 3, pp. 367–373, 1995.

[PR09] J. Pasternack and D. Roth, "Extracting article text from the web with maximum subsequence segmentation," in *Proceedings of the 18th international conference on World Wide Web*, pp. 971-980, ACM, 2009.

[QA99] K. Qiu and S. Akl, *Parallel maximum sum algorithms on interconnection networks*, Tech. Rep., pp. 99–431, Queen's University, Department of Computer and Information Science, 1999.

[RT99] W. L. Ruzzo and M. Tompa, "A linear time algorithm for finding all maximal scoring subsequences," in *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pp. 234-241, International Society for Computational Biology, 1990.

[Smi87] D. R. Smith, "Applications of a strategy for designing divide-and-conquer algorithms," *Sci. Comput. Program*, vol. 8, no. 3, pp. 213-229, 1987.

[Sug07] O. V. Sugakova, "The counting process and summation of a random number of random variables," *Theory of Probability and Mathematical Statistics,* vol. 74, pp. 181-189, 2007.

[TT98] H. Tamaki, and T. Tokuyama, "Algorithms for the maximum subarray problem based on matrix multiplication," In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 446–452, SIAM, 1998.

[TVP05] T. Takaoka, K. Voges, and N. Pope, "Algorithms for data mining," in *Business Applications and Computational Intelligence*, ed. K. Voges and N. Pope, pp. 291–315, Idea Group Publishing, 2005.

[Wen95] Z. Wen, "Fast parallel algorithms for the maximum sum problem," *Parallel Computing*, vol. 21, vol. 3, pp. 461-466, 1995.

VITA

Zhu Wang

Candidate for the Degree of

Doctor of Philosophy

Thesis: A PARALLEL ALGORITHM FOR FINDING ALL MINIMAL MAXIMUM SUBSEQUENCES VIA RANDOM-WALK THEORY

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2015.

Completed the requirements for the Master of Science in Information Systems at Dakota State University, Madison, South Dakota in May, 2004.

Completed the requirements for the Bachelor of Science in Industrial Foreign Trade at University of Electronic Science and Technology of China, Chengdu, Sichuan, China in July, 1993.

Experience:

Senior Software Engineer, Weatherford Inc., Katy, Texas, from 2008 - 2015.

Teaching Assistant, Computer Science Department, Oklahoma State University, 2005 - 2008.