Efficient Algorithms for Maximum-Density Segment Problem

Xue Wang, Fasheng Qiu

Sushil K. Prasad, Guantao Chen

Georgia State University, Atlanta, GA 30303

April 21, 2010

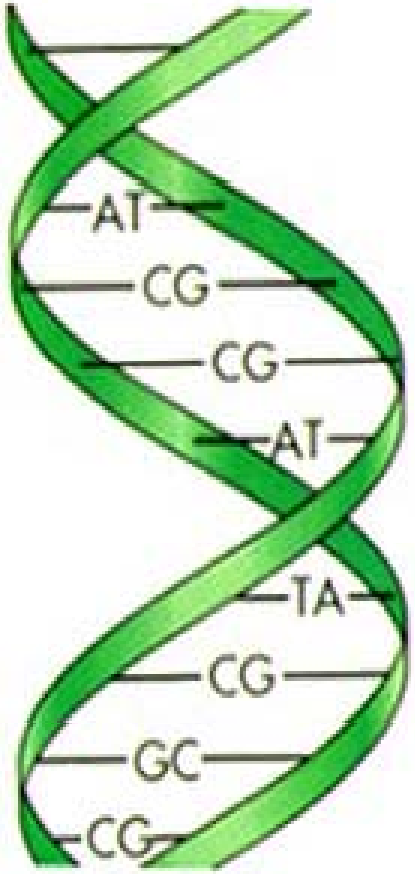
Outline

- Introduction
  - Origin of the maximum-density problem
  - Description of the maximum-density problem
  - Current progress
  - Why more efficient algorithm is needed
  - Our contribution
- Our Approach
  - Basic concepts
  - Overall idea
  - Find right-skew pointers efficiently
  - Three steps to find maximum-density segment.
- Implementation and Experimental Results
  - Comparison with naïve algorithm
  - Some additional words
- Ongoing and Future Work

Origin

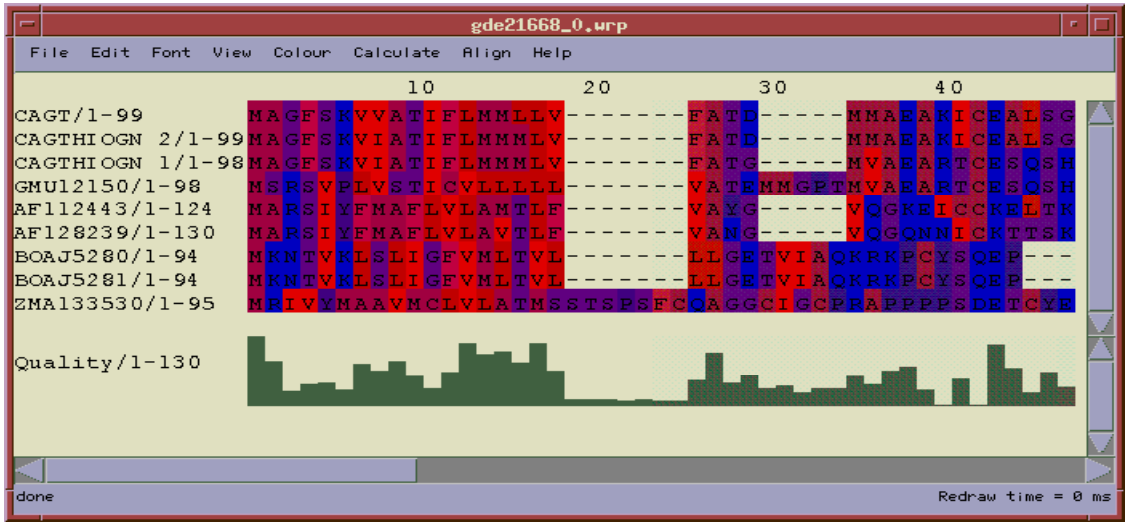
- In DNA sequence, GC-content is positively correlated with gene length, gene density, patterns of codon usage, recombination rate within chromosomes, …

- Find a segment where the density of C and G is maximum



http://www.topnews.in/health/files/dna.jpg

Origin



http://home.cc.umanitoba.ca/~psgendb/bioLegato/multalign/defensin.jalview.gif

- Sequence alignment
- Each aligned column has an alignment score
- Identify segment that is aligned best from output sequence

Problem Definition

S = $s_1 \ldots s_n$ =(a_1,w_1), (a_2,w_2), …, (a_n,w_n). w_i is the width of s_i

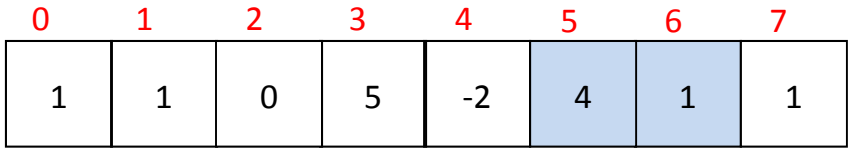
0	1	2	3	4	5	6	7
1	1	0	5	-2	4	1	1

- $\mu(i,j) = \frac{\sum_{i \le k \le j} (a_k)}{\sum_{i \le k \le j} (w_k)} = \frac{S[i,j]}{w(i,j)}$ μ(i,j) is the density of S[i,j].

- C_{min} is the width constraint

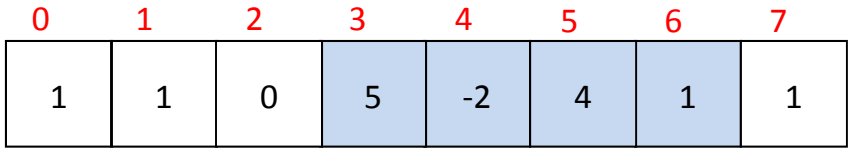
- **Definition: for given S and C_{min} , find *i, j* such that C_{min} <= *w(i,j)* and *μ(i,j)* is maximized.**

An Example



- $w_i = 1, 0 \leq i \leq 7$
- a1 = 1, a2 = 1, …., a7 = 1
- $C_{min} = 2$

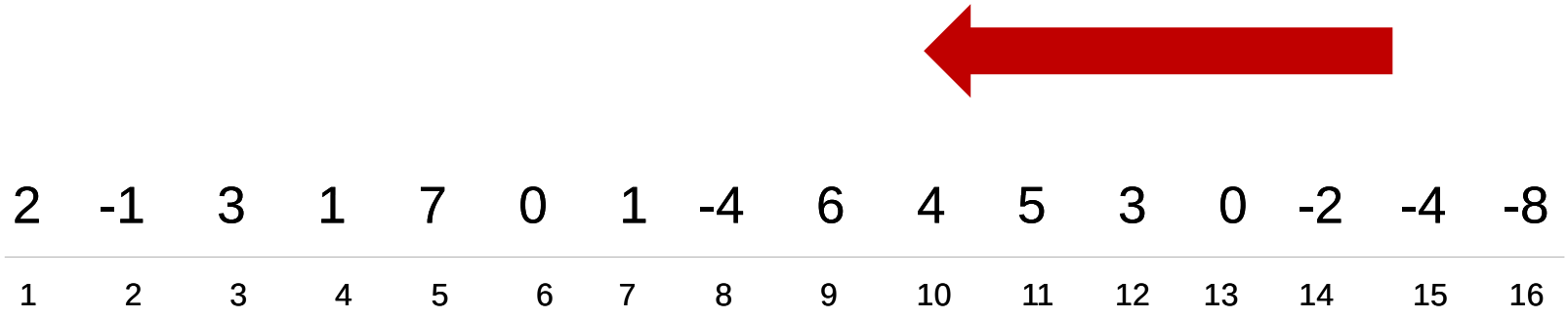
An Example



- $w_i = 1, 0 \leq i \leq 7$
- a1 = 1, a2 = 1, …., a7 = 1
- $C_{min} = 3$

Current Status

- O(n) algorithm for fixed width.
- O(nlogC_{min}) algorithm proposed in 2002.
- Two optimal sequential algorithms (O(n)) proposed in 2005.

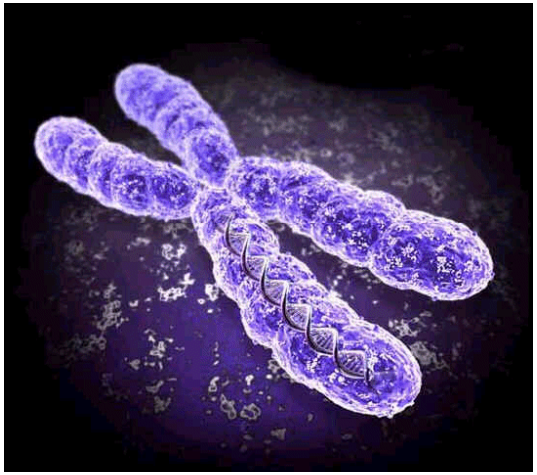


Treatment of current position depends on previous position

- No non-trivial parallel algorithm has been known.

Why more efficient algorithms are needed

- Super-large sequences:
  - The longest chromosome from human has approximately 2,220,000,000 base pairs.



http://www.odec.ca/projects/2005/anna5m0/public_html/images/chromosome.gif

- Output maximum-density segments starting from **each** index.
  - Current best sequential algorithm is $O(n\log C_{min})$

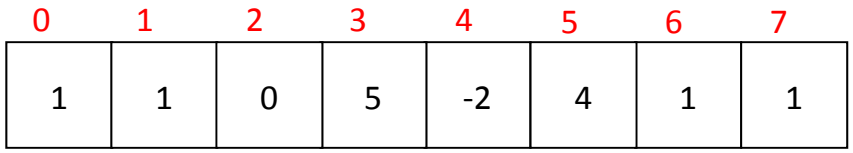
0	1	2	3	4	5	6	7
1	1	0	5	-2	4	1	1

Our Contribution

- Proved a “Combination Lemma”.
- Proposed n-processor $O(\log^2 n)$ algorithm.
- Implemented the proposed algorithm on GPU using CUDA.

Some Thoughts

- About the C_{min}
- n processor, O(n) time complexity
- n^2 processor, O(logn) time complexity
- cost is = $\Omega(n^2)$



width $w_i = 1$

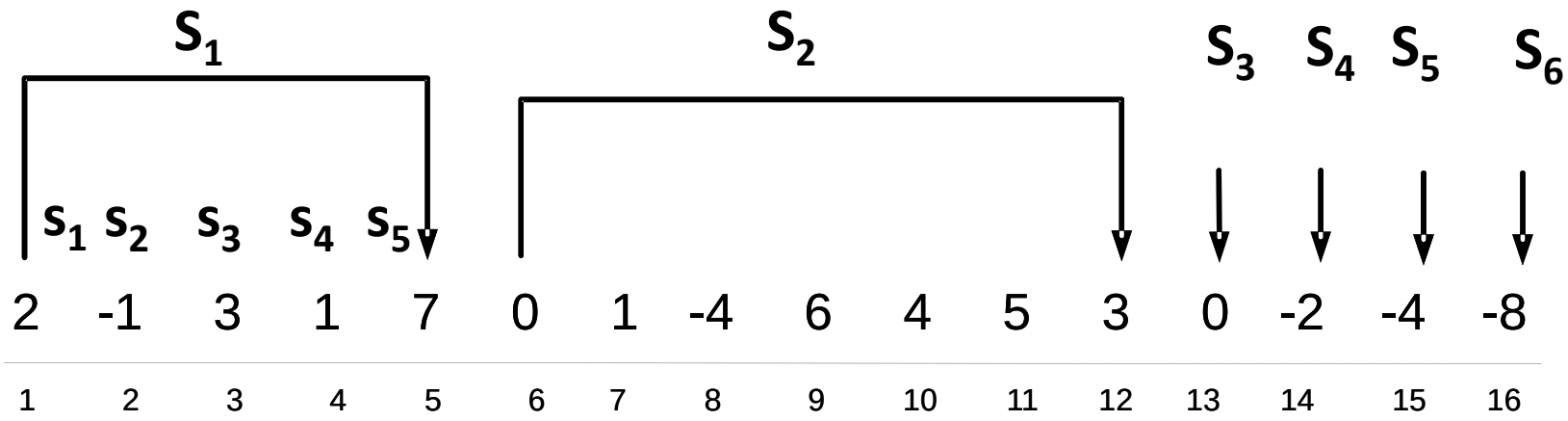
- Is there a better way?

Outline

- Introduction
  - Origin of the maximum-density problem
  - Description of the maximum-density problem
  - Current progress
  - Why more efficient algorithm is needed
  - Our contribution
- Our Approach
  - Basic concepts
  - Overall idea
  - Find right-skew pointers efficiently
  - Three steps to find maximum-density segment.
- Implementation and Experimental Results
  - Comparison with naïve algorithm
  - Some additional words
- Ongoing and Future Work

Basic Concepts/DRSP [Lin et al.]

- DRSP: Decreasing right skew partition
- Existence and uniqueness

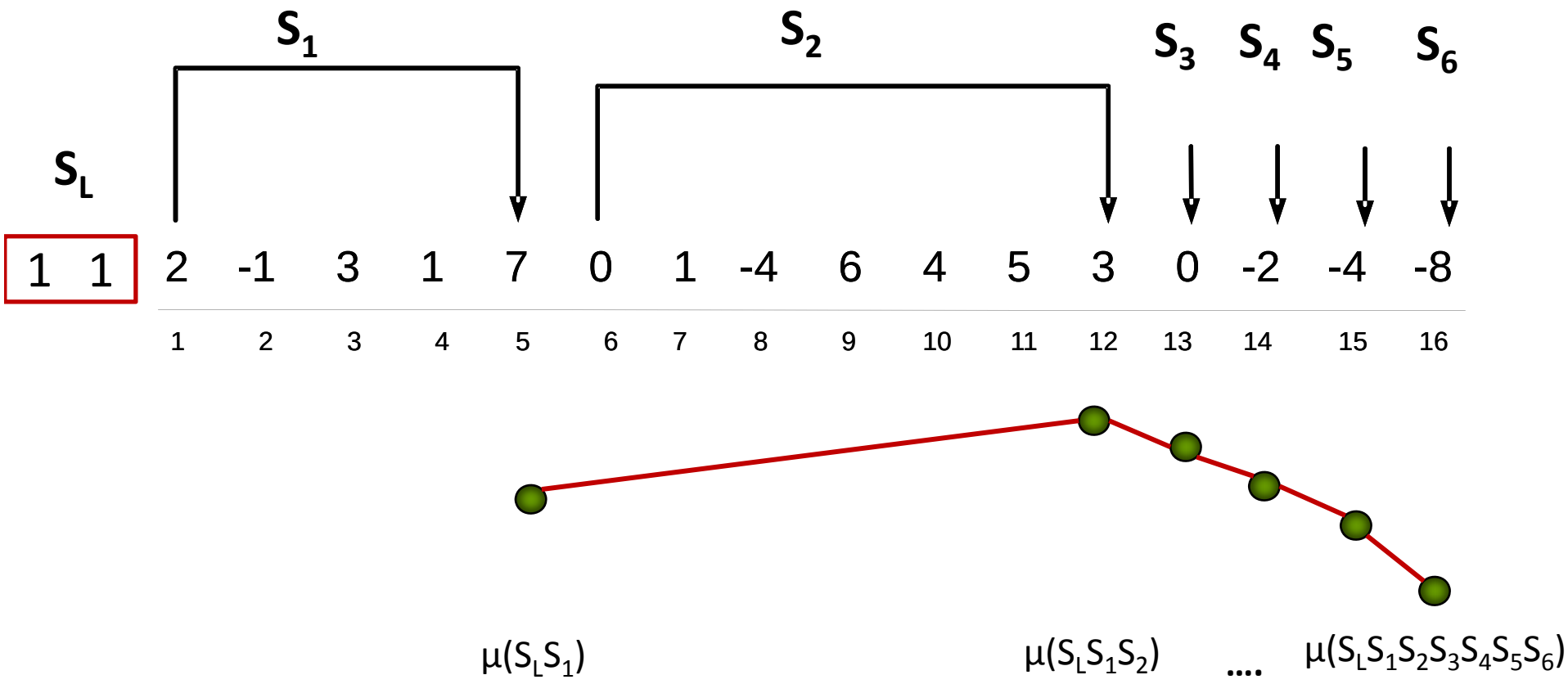


Decreasing: $\mu(S_1) > \mu(S_2) > \mu(S_3) > \mu(S_4) > \mu(S_5) > \mu(S_6)$

Right skew: the density of any prefix sequence is no more than the density of remaining suffix : $\mu(s_1) \leq \mu(s_2s_3s_4s_5)$, $\mu(s_1s_2) \leq \mu(s_3s_4s_5)$, …

Basic Concepts/DRSP [Lin et al.]

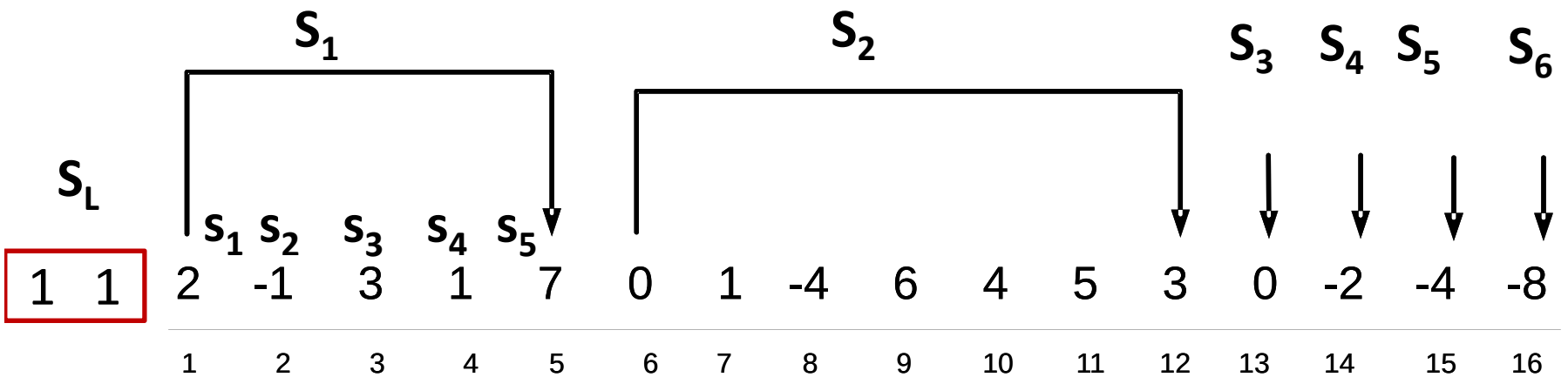
- Bitonic property



Basic Concepts/DRSP [Lin et al.]

- Bitonic property
- Atomic property:

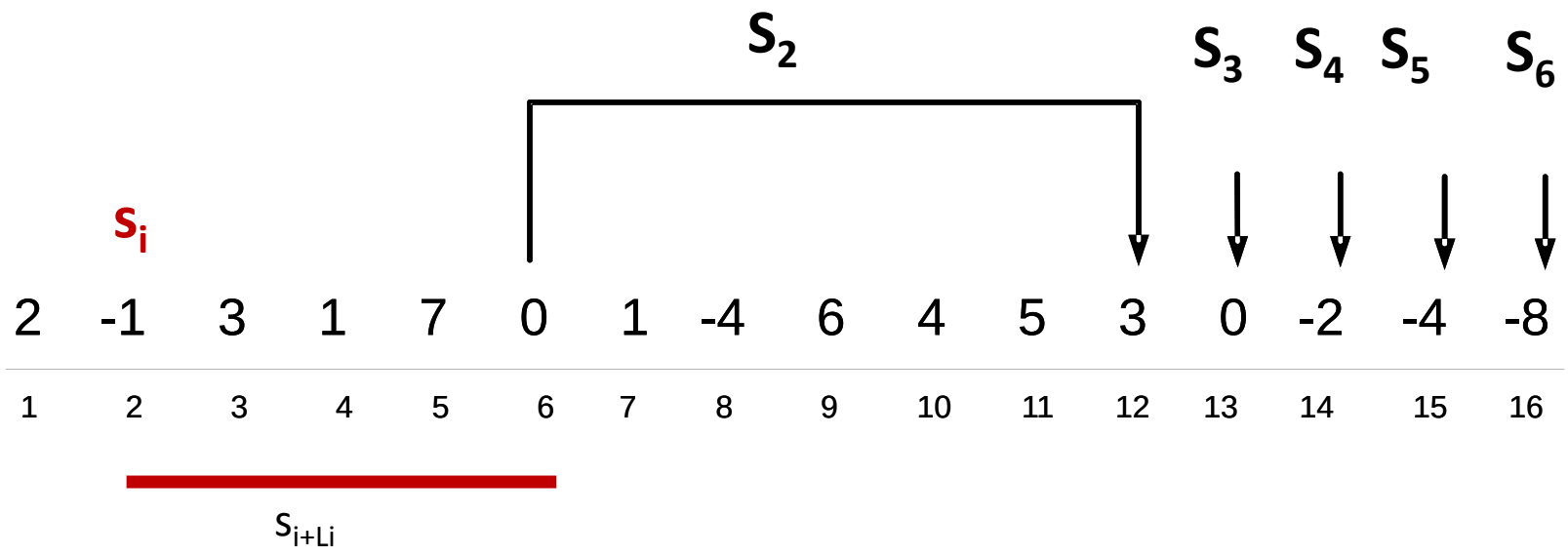
  if $\mu(S_L S_1) \geq \mu(S_L)$, then $\mu(S_L S_1) >= \mu(S_L s_1 \ldots s_i)$, $1 <= i <= 5$



$\mu(S_L S_1) = 14/7 = 2$, $\mu(S_L s_1) = 4/3$, $\mu(S_L s_1 s_2) = 3/4$, …

General Idea

- If for each i, 1<= i <=n, we know the DRSP of $s_{i+Li}...s_n$
- With one processor, the maximum-density segment beginning with i can be identified within O(log(n)) by binary search.
- With n processors, maximum density segment beginning with i can be identified in O(log(n)) in parallel



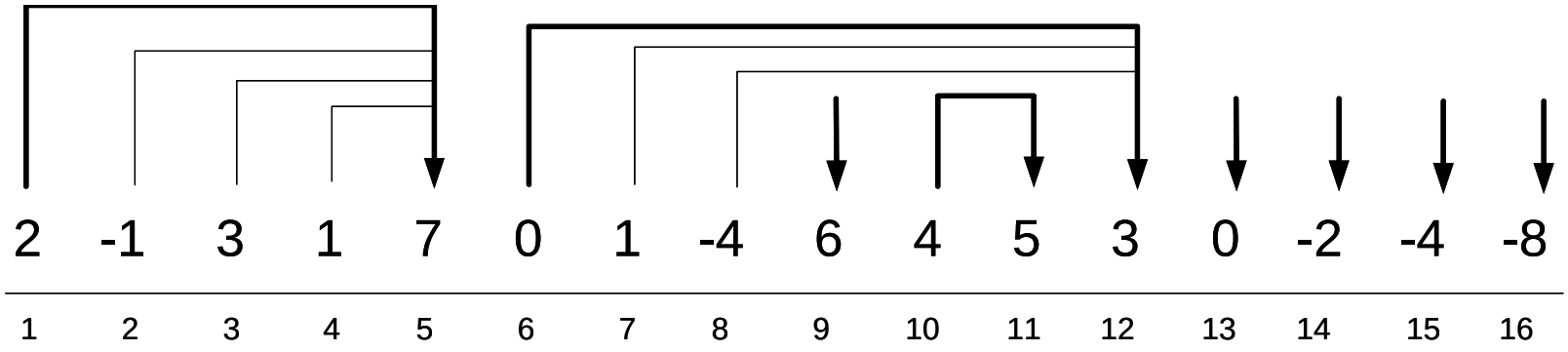
General Idea

Efficiently find the DRSP of each suffix of S?

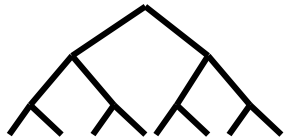
Right Skew Pointers [Goldwasser et al.]

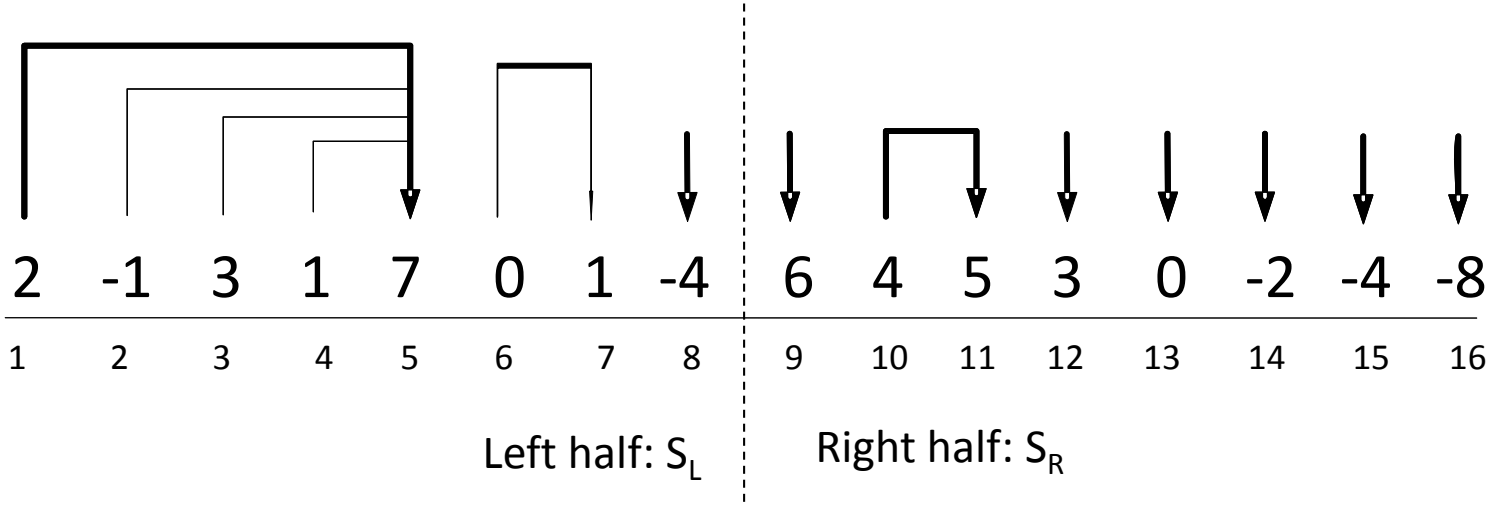
❖ The right skew pointer tells us the DRSP of $s_i \ldots s_n$ (**The use of right-skew pointers**)

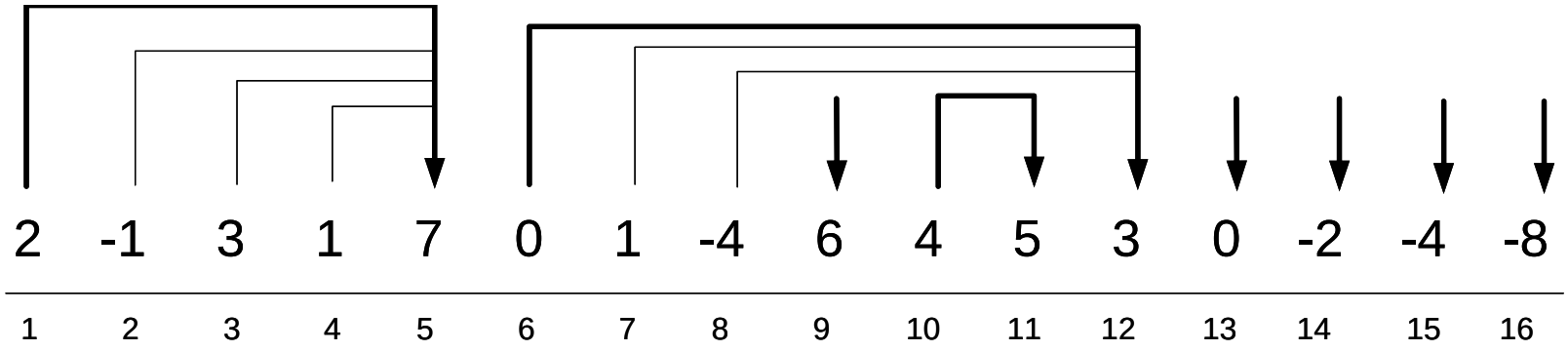
❖ For each s_i, find the longest right skew segment beginning with s_i (Lemma) (**How to find right skew pointers**)



Efficiently Find Right Skew Pointers







As a whole

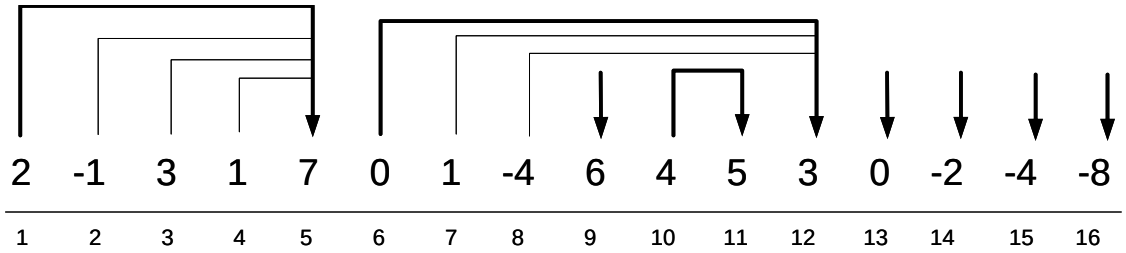
Efficiently Find Right Skew Pointers

Lemma 3.2: (Combination) Let DRSP(S_L)=$S_1, \ldots, S_t$ and DRSP(S_R)=$S_{t+1}, \ldots, S_{t+u}$. Let k be the greatest index $i \in [1, t+u]$ that maximizes $\mu(S_1 \ldots S_i)$, then the first right-skew segment of DRSP($S_L S_R$) is $S_1 \ldots S_k$. Moreover, k is either 1 or $\in [t+1, t+u]$.

Guarantee that the construction of right-skew pointers can be done in a parallel and binary fashion.

General Steps

❖ **Step 1:** Find right-skew pointers for S ($\log^2 n$)



❖ **Step 2:** Find for each i, i* such that $w(i, i^*) \geq C_{min}$ and $\mu(i, i^*)$ reaches maximum. (logn)

2	-1	3	1	7	0	1	-4	6	4	5	3	0	-2	-4	-8
1				1*											

❖ **Step 3:** Identify the maximum-density segment from all S[i, i*]. (logn)

Efficiently Find Right Skew Pointers

```
1  Compute the prefix sums of values and widths of S;
2  PDRSP(S, n);
```

procedure PDRSP(S, n) // find right-skew pointers of S

```
1   for i = 1 to n do in parallel   // initialization
2       p[i] ← i;
3   end for
4   if n is 1 then return;   // base case
5   do in parallel
6       PDRSP(S[1,⌈n/2⌉], ⌈n/2⌉);   // divide-and-conquer
7       PDRSP(S[⌈n/2⌉+1,n], n−⌈n/2⌉);
8   COMBINE(S[1,⌈n/2⌉], S[⌈n/2⌉+1,n]);
```

procedure COMBINE(S_L, S_R) // find right-skew pointers of $S_L S_R$ using known right-skew pointer of S_L and S_R

```
1   l_L ← length of S_L, l_R ← length of S_R;
2   construct T_PJ of S_R;   // T_PJ is the pointer-jumping table
3   for each s_i ∈ S_L do in parallel
4       S_L,i ← S_L[i, l_L];
5       p[i] ← SUFFIX-FIND (S_L,i, S_R, i);
6   end for
```

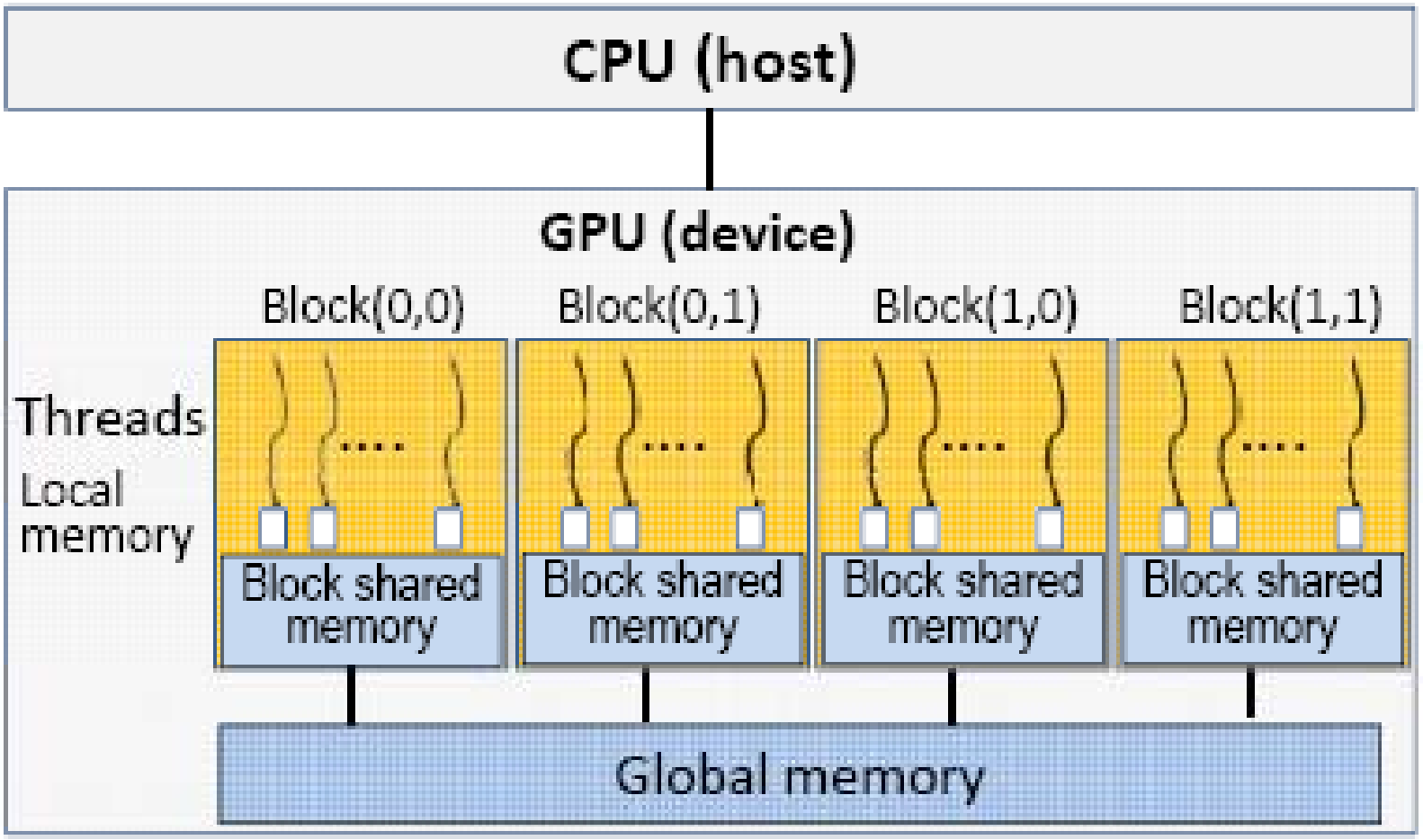
procedure SUFFIX-FIND$(S_{L,i}, S_R, i)$ // find the right-skew pointer $p[i]$ in $S_{L,i}S_R$

```
1    j ← 0;
2    for k ← ⌈lg l_R⌉ to 0 do   // binary search
3        if j ≥ l or μ(S_L,i S_R[1, j]) > μ(S_L,i S_R[1, p[j+1]]) then
4            if μ(S_L[i, p[i]] > μ(S_L,i S_R[1, j]) then return p[i];
5            else return l_L + j;
6        r ← T_PJ(j+1, k);
7        if r < l_R and μ(S_L,i S_R[1, r]) ≤ μ(S_L,i S_R[1, p[r+1]]) then j ← p[r+1];
8   end for
9   if j < l_R and μ(S_L,i S_R[1, j]) ≤ μ(S_L,i S_R[1, p[j+1]]) then   // final check
10      if μ(S_L[i, p[i]]) > μ(S_L,i S_R[1, p[j+1]]) then return p[i];
11      else return l_L + p[j+1];
12  return l_L + j;
```

Outline

- Introduction
  - Origin of the maximum-density problem
  - Description of the maximum-density problem
  - Current progress
  - Why more efficient algorithm is needed
  - Our contribution
- Our Approach
  - Basic concepts
  - Overall idea
  - Find right-skew pointers efficiently
  - Three steps to find maximum-density segment.
- Implementation and Experimental Results
  - Comparison with naïve algorithm
  - Some additional words
- Ongoing and Future Work

CUDA Architecture



nVIDIA GTX280

Finding good partner with CUDA

n elements, block size *t,* # blocks is *n/t.*

$t \leq 512$ due to the required shared memory.

***t = 512* for maximal use of shared memory.**

- Find right skew pointers. *Seq. is divided up to equal-size sub sequences (except the last one).*

  -Find right skew pointers for each block (subsequence)

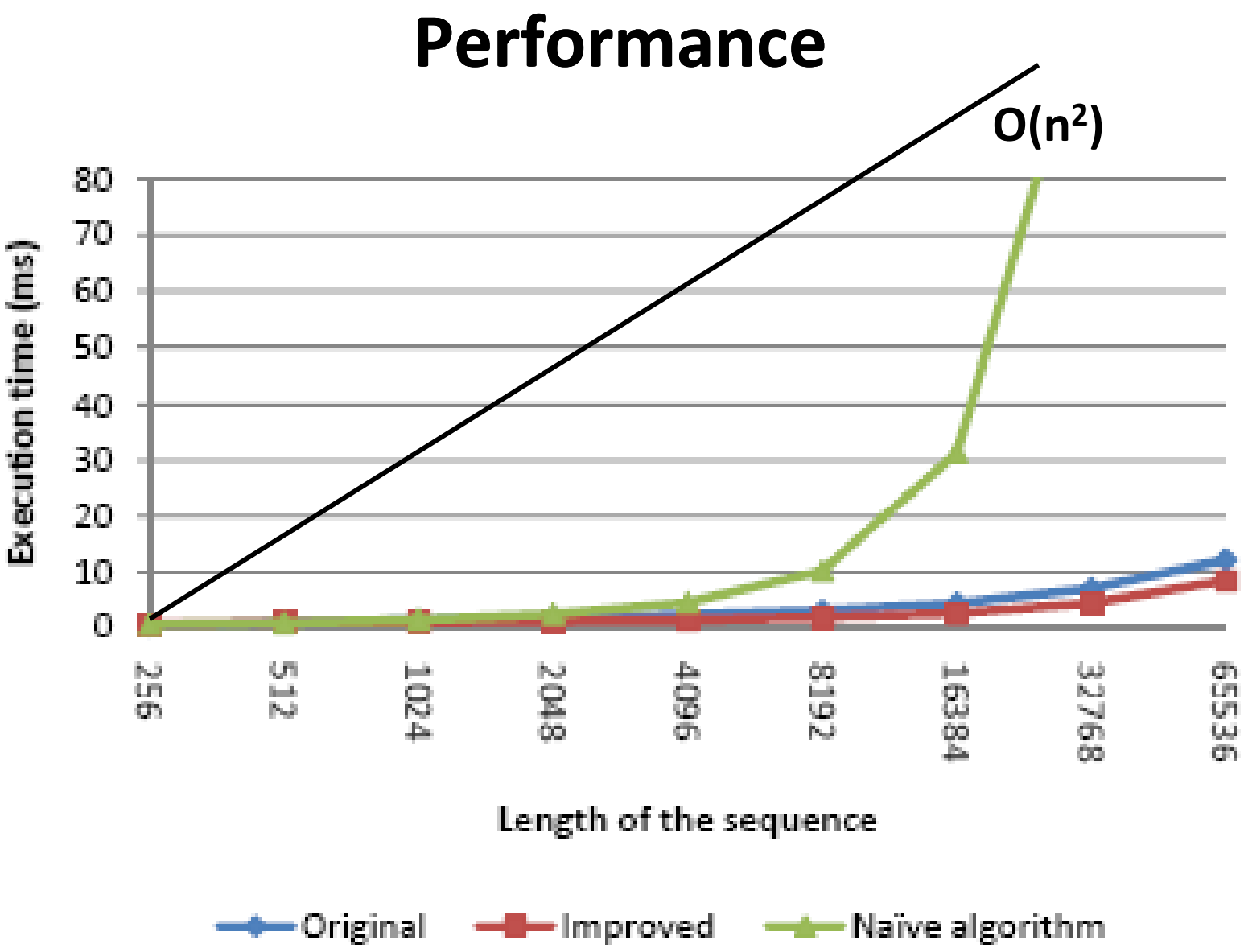
  intra-block communication (through shared memory, less expensive). Each block processes a subsequence of length t.

  - Find right skew pointers for the entire sequence (by melding the RSP of all subsequences).

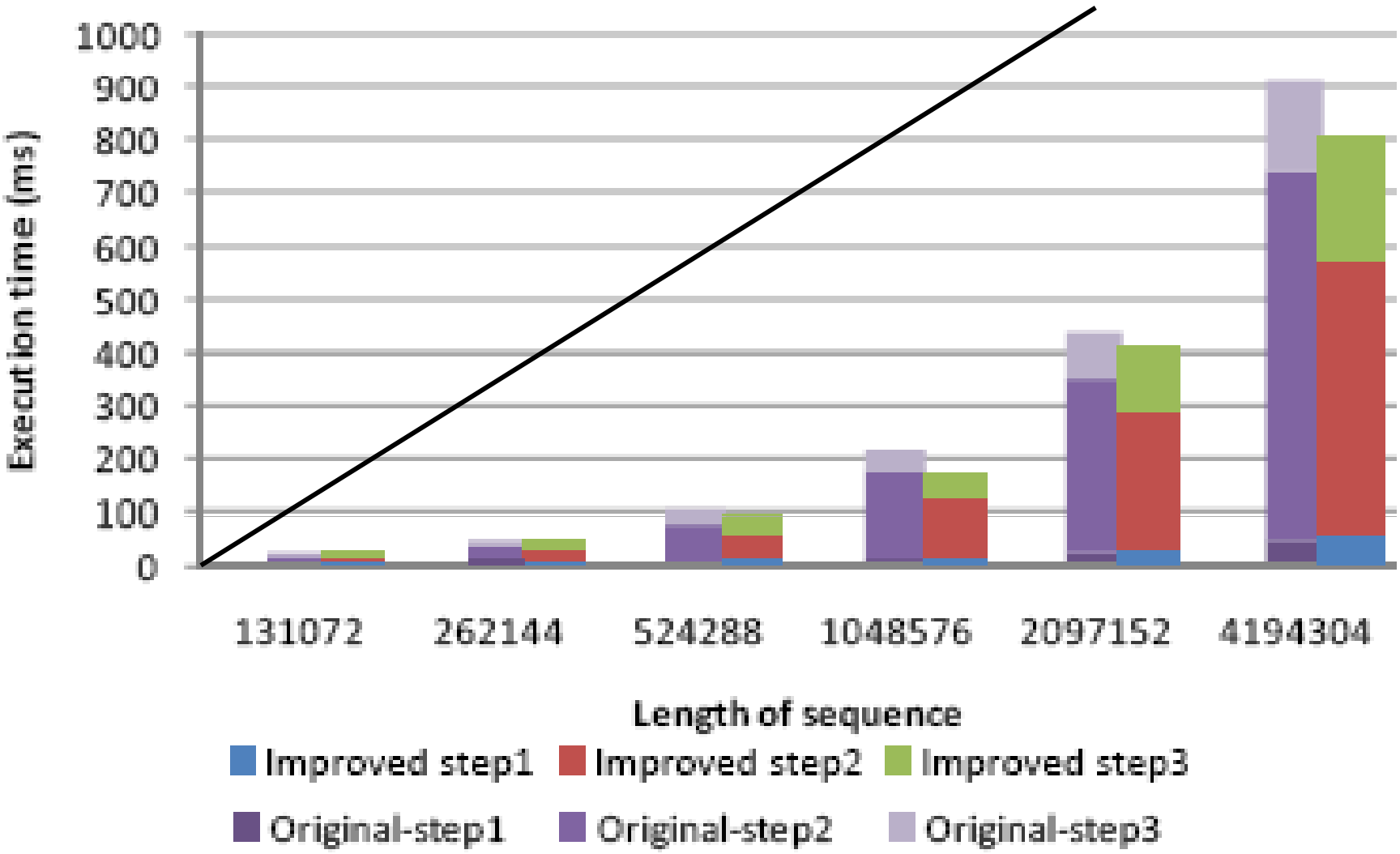
  inter-block communication (through global memory, much expensive).

- Find good partners

  Each thread finds the good partner for its corresponding element.


Performance
O(n²)
Execution time (ms)
80
70
60
50
40
30
20
10
0
256
512
1024
2048
4096
8192
16384
32768
65536
Length of the sequence
Original
Improved
Naïve algorithm

Performance



Some Additional Words

- We have implemented optimal sequential algorithm
- One a single CPU of GTX 280, out of memory above 3 million size
- Current parallel algorithm should be further optimized.

Ongoing and Future Work

- Algorithm for both lower and upper bound is available
- Optimize CUDA code and compare parallel algorithm with optimal sequential algorithm
- Apply to real sequences
- Reduce memory requirement
- Cost-optimal parallel algorithm

Thanks!

Questions?

References

Yaw-Ling Lin, Tao Jiang, and Kun-Mao Chao. Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis. Journal of Computer and System Sciences, 65(3):570 – 586, 2002.

Michael H. Goldwasser, Ming-Yang Kao, and Hsueh-I Lu. Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications. Journal of Computer and System Sciences, 70(2):128 – 144, 2005.

Kai min Chung and Hsueh-I Lu. An optimal algorithm for the maximum-density segment problem. SIAM J Comput, 34:373 – 387, 2005.

X. Huang. Local rates of recombination are positively correlated with gc content in the human genome. Mol Biol Evol, 10:219 – 225, 1994.