Parallel Algorithms

Jongseo Lee (KAIST)

2023-01-10

Note

Goals & Non-Goals

- Goals
  - Explore about two recent models of parallelism
- Non-Goals
  - Understand everything in the textbook

Massively Parallel Computing Model (MPC)

Definition

- **Massively Parallel Computing Model (MPC)**
  - What does the MPC consist of?
  - How does the input given to the MPC?
  - How does the MPC output the answer?
  - How does the MPC perform computation?
  - Which parameters of the MPC are important?

Massively Parallel Computing Model (MPC)

Definition

- The **Massively Parallel Computing Model (MPC)** consists of:
  - The input data size N
  - The number of machines M (machines will communicate with each other)
  - The memory size, s words, each machine can hold
- We are mostly interested if $M \cdot s = \Theta(N)$, $M \cdot s = O(N \operatorname{polylog} N)$, etc.
  - Note that $M \cdot s$ is the amount of the available resources

Massively Parallel Computing Model (MPC)

Definition

- Input
  - The input data is split across the M machines arbitrarily
- Output
  - Some machines will be the **output machines**.
  - The answer will be in the output machines.

Massively Parallel Computing Model (MPC)

Definition

Example - An integer array $A[]$ is given in the form of (i, a_i) across the machines

M_0	(0, 2)	(2, 2)	(3, 3)	(8, 1)	(10, 3)	

M_1	(4, 0)	(6, 1)	(1, 0)			

M_2	(7, 0)	(11, 4)	(5, 1)			

Massively Parallel Computing Model (MPC)

Definition

Example - The output data, prefix sums of the given array are stored across the machines

M_0	(0, 2)	(1, 2)	(2, 4)	(3, 7)		

M_1	(4, 7)	(5, 8)	(6, 9)	(7, 9)		

M_2	(8, 10)	(9, 12)	(10, 15)	(11, 19)		

Massively Parallel Computing Model (MPC)

Definition

- In each round of the computation,
  1. each machine performs some computation on the local data
  2. each machine sends/receives messages to any other machine
     - Note: A machine can send/receive at most s words per round

Massively Parallel Computing Model (MPC)

Definition

- Main Parameters
  - The number of machines, M (mostly omitted)
  - The number of communication rounds an algorithm
  - The size of local memory s
    - Problems are easier to solve with larger s. What if $s \gg N$?
    - Typically, $s = O(N^{\varepsilon})$ for $\varepsilon < 1$.

Massively Parallel Computing Model (MPC)

Definition

Definition.

An MPC algorithm is **sublinear** if $s \ll N$. Formally, $s = N^{\varepsilon}$ for some constant $\varepsilon < 1$.

Example.

An MPC algorithm with $s = n^{1+\delta}$ for some $\delta < 1$ on dense graphs is considered to be sublinear.

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

- Input: n pair of integers $(0, a_0), (1, a_1), \cdots, (N-1, a_{N-1})$
- Output: $(i, \sum_{j=0}^{i} a_j)$ for $0 \leq i < N$.
- In this example, we assume $M, s = \Theta(N^{0.5})$.

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

- Input: n pair of integers $(0, a_0), (1, a_1), \cdots, (N-1, a_{N-1})$
- Output: $(i, \sum_{j=0}^{i} a_j)$ for $0 \leq i < N$.

2	0	2	3	0	1	1	0	1	2	3	4
2	2	4	7	7	8	9	9	10	12	15	19

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

Initial State

M_0	(0, 2)	(2, 2)	(3, 3)	(8, 1)	(10, 3)	

M_1	(4, 0)	(6, 1)	(1, 0)			

M_2	(7, 0)	(11, 4)	(5, 1)			

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

Communication Round 1: Send (i, A_i) to machine $\left\lfloor \frac{Mi}{N} \right\rfloor$

M_0	(0, 2)	(1, 0)	(2, 2)	(3, 3)		

M_1	(4, 0)	(5, 1)	(7, 0)	(6, 1)		

M_2	(9, 2)	(11, 4)	(8, 1)	(10, 3)		

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

Computation Round 1: **Sort** and compute prefix sum in each machine

M_0	(0, 2)	(1, 0)	(2, 2)	(3, 3)		

M_1	(4, 0)	(5, 1)	(6, 1)	(7, 0)		

M_2	(8, 1)	(9, 2)	(10, 3)	(11, 4)		

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

Computation Round 1: Sort and **compute prefix sum** in each machine

M_0	(0, 2)	(1, 2)	(2, 4)	(3, 7)		

M_1	(4, 0)	(5, 1)	(6, 2)	(7, 2)		

M_2	(8, 1)	(9, 3)	(10, 6)	(11, 10)		

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

Communication Round 2: Send total sum of machine i to machine $j (j > i)$

M_0	(0, 2)	(1, 2)	(2, 4)	(3, 7)		
M_1	(4, 0)	(5, 1)	(6, 2)	(7, 2)	**7**	
M_2	(8, 1)	(9, 3)	(10, 6)	(11, 10)	**2**	**7**

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

Computation Round 2: Add the numbers received into prefix sums

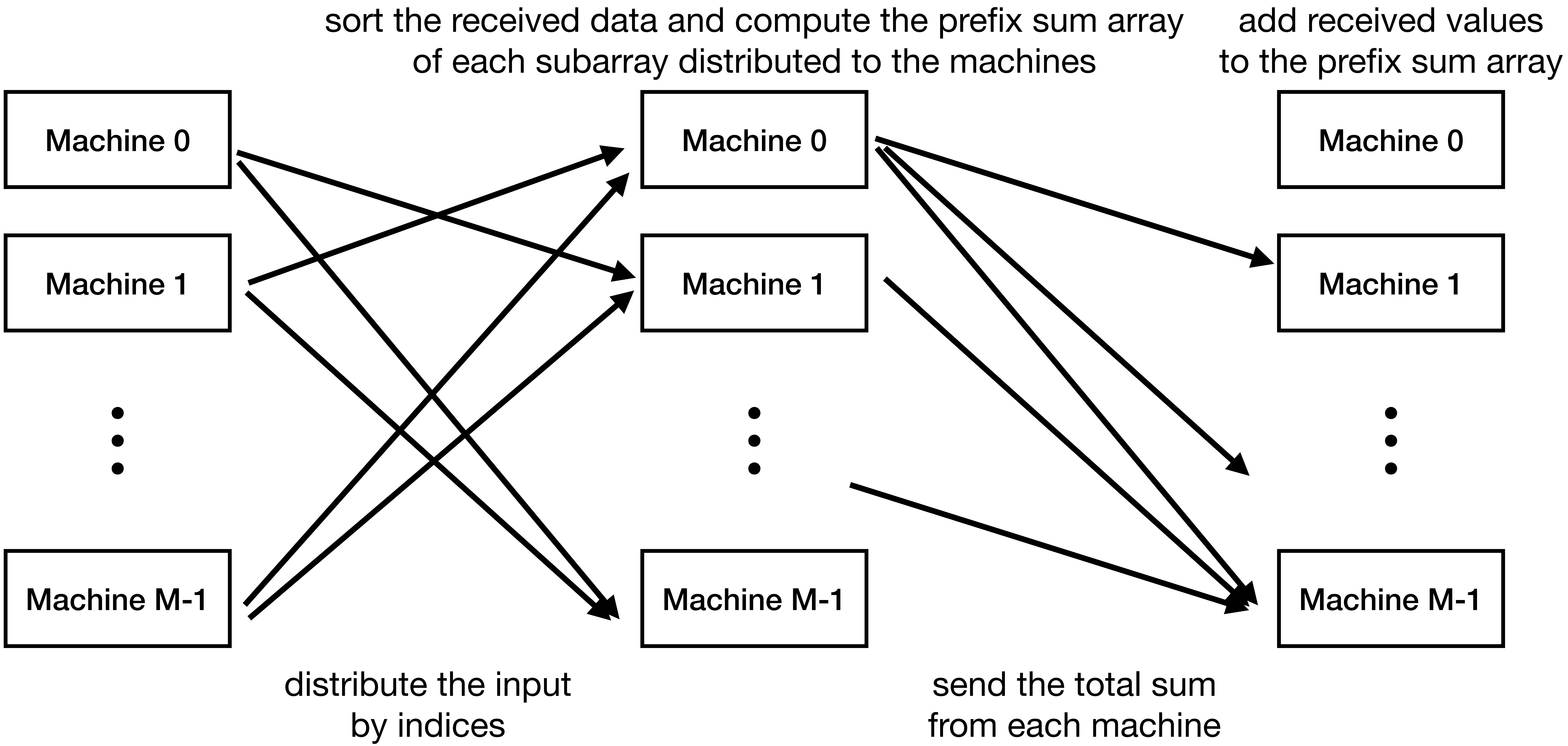
M_0	(0, 2)	(1, 2)	(2, 4)	(3, 7)		

M_1	(4, 7)	(5, 8)	(6, 9)	(7, 9)		

M_2	(8, 10)	(9, 12)	(10, 15)	(11, 19)		

Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum



Massively Parallel Computing Model (MPC)

Example: Computing Prefix Sum

- By taking $M = N^{0.5}, s = 2 \times N^{0.5}$, we used
  - $O(N^{0.5})$ machines each of them having $O(N^{0.5})$ words of memory
  - $O(1)$ rounds of communications, with total $O(N)$ words

Massively Parallel Computing Model (MPC)

Algorithms - Maximal Matching

Maximal Matching

- Instance

  A graph $G = (V, E)$

- Question

  Return a matching $M \subseteq E$ with every edge in $E \backslash M$ having an endpoint in M

Massively Parallel Computing Model (MPC)

Algorithms - Maximal Matching

Overview

- Repeat until the number of remaining edges fit into a single machine:
  - Put a subset of edges into a single machine
  - Compute a maximal matching
  - remove matched vertices

Massively Parallel Computing Model (MPC)

Algorithms - Maximal Matching

The algorithm (Note: $s = n^{1+\varepsilon}$)

- Let $G_0 := G$
- For round $r = 0,1,\cdots,R-1$ (Note: $E(G_r) = \emptyset$)
  - Each machine marks each of its edges independently with probability $p = \frac{n^{1+\varepsilon}}{2m}$
  - Each machine sends marked edges to machine 0
  - Machine 0 computes a maximal matching M_r and announce matched vertices
  - Remove edges having a matched endpoint and set G_{r+1} as a current graph

Massively Parallel Computing Model (MPC)

Algorithms - Maximal Matching

Fix a round and let m be the number of edges at the start of the round.

Lemma.

With high probability, the number of marked edges fit into a single machine

Lemma.

With high probability, the number of remaining edges is at most $\frac{10m}{n^{\varepsilon}}$

Massively Parallel Computing Model (MPC)

Algorithms - Maximal Matching

Theorem.

The algorithm terminates in $R = O(1/\varepsilon)$ rounds

Proof.

There are at most n^2 edges initially

After $R \leq \log_{n^\varepsilon} n^2 = O(1/\varepsilon)$ rounds, due to the second Lemma, there are at most $s = n^{1+\varepsilon}$ edges left.

Massively Parallel Computing Model (MPC)

Algorithms - Connected Components

Connected Components (ConnComp)

- Instance

  An undirected graph $G = (V, E)$

- Question

  Which vertices are in the same connected component?

  A solution is a labeling of vertices s.t. $\ell(u) = \ell(v)$ iff there's a path from u to v

Massively Parallel Computing Model (MPC)

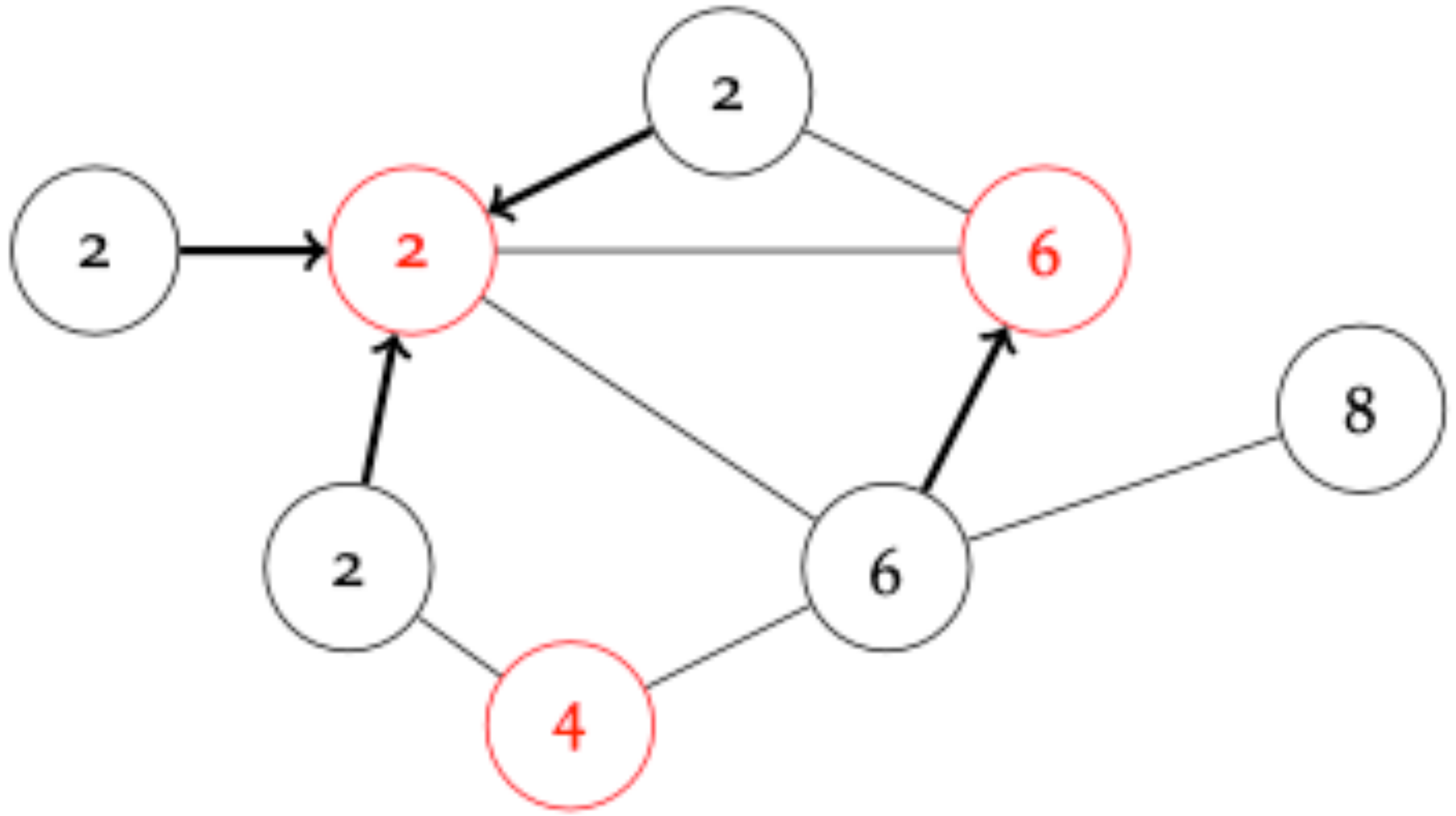
Algorithms - Connected Components

Idea for handling sparse graphs: Graph Exponentiation

- Operation: Connect every vertex with vertices of distance ≤ 2
- Repeating the operation $O(\log D)$ steps, each connected component becomes a clique
    - D: diameter of G, the length of the longest shortest path
- Problem: required memory of a single machine can be up to $\Omega(n^2)$

Massively Parallel Computing Model (MPC)

Algorithms - Connected Components



Idea for handling dense graphs: Label Contraction

- Operation
  - Mark each vertex independently with prob. p
  - Relabel each unmarked vertex with a marked neighbor
  - The denser the graph, the smaller the value of p required for the contraction

image: http://people.csail.mit.edu/ghaffari/MPA19/Notes/MPA.pdf

Massively Parallel Computing Model (MPC)

Algorithms - Connected Components

The Algorithm: Putting them together (Rough Sketch)

- Repeat the following phase:
    1. Perform the graph exponentiation, collapse the vertices into a supernode
    2. Perform label contraction
- Choosing the appropriate p allows us to obtain a sublinear algorithm having $O(\log D \cdot \log \log_{m/n} n)$ round complexity
    - Each phase takes $O(\log D)$ rounds
    - $O(\log \log_{m/n} n)$ phases suffice

Massively Parallel Computing Model (MPC)

What the MPC can't do

2-Cycle Problem

- Instance: A graph that is either one cycle or the union of two cycles
- Question: Determine if the graph is one cycle or the union of two cycles

2-Cycle Conjecture

Any sublinear MPC algorithm for the 2-Cycle problem requires $\Omega(\log n)$ rounds

Furthermore, it is conjectured to hold even for the problem of distinguishing

- A cycle of length n
- Or the two cycles of length $n/2$

Massively Parallel Computing Model (MPC)

What the MPC can't do

Theorem. Assuming 2-Cycle conjecture, any sublinear MPC for undirected connectivity problem requires $\Omega(\log n)$ rounds

Proof. Reduction from 2-Cycle problem to undirected connectivity problem

- consisting of one cycle → connected
- consisting of two cycles → disconnected

Massively Parallel Computing Model (MPC)

What the MPC can't do

Component-stable MPC algorithms for a graph problem

- The output of each vertex depends only on the connected component that it belongs to

Massively Parallel Computing Model (MPC)

What the MPC can't do

Theorem. Assuming 2-Cycle conjecture, following holds for sublinear component-stable MPC algorithms

- Maximal matching requires $\Omega(\log \log n)$ rounds (including its constant approximation)
- For every constant c, c-coloring of a cycle requires $\Omega(\log \log^* n)$ rounds
- Any constant approximation for Vertex Cover requires $\Omega(\log \log n)$ rounds
- (Using more assumptions) $(\Delta + 1)$-coloring of a graph requires $\Omega(\sqrt{\log \log n})$ rounds

Adaptive Massively Parallel Model (AMPC)

Definition

The **Adaptive Massively Parallel Model (AMPC)** is basically the MPC model but with shared random access memory

- In the i-th round, each machine can read data from random access memory D_{i-1} and write to D_i
  - D_i s are common for all machines
- Each machine can perform $\leq S$ reads and $\leq S$ writes in a single round
  - Note: S is different from s

Clearly, the AMPC is at least as strong as the MPC.

Adaptive Massively Parallel Model (AMPC)

Example

- Suppose there's a function $g : X \to X$ and $\{(x, g(x)) : x \in X\} \subseteq D_{i-1}$.
- Then, given $y \in X$ and $k = O(S)$, a machine can compute $g^k(y)$.

Adaptive Massively Parallel Model (AMPC)

What the AMPC can do better than the MPC

For now, **2-Cycle problem** refers to the problem of distinguishing between:

- A single cycle on n vertices
- Two cycles on $n/2$ vertices each

We aim to derive an AMPC algorithm solving 2-Cycle problem in $O(1)$ rounds with high probability

Adaptive Massively Parallel Model (AMPC)

What the AMPC can do better than the MPC

The Algorithm: $s = O(n^{\varepsilon})$ (sublinear) and takes $O(1/\varepsilon) = O(1)$ rounds

ALGORITHM 1: SHRINK($G = (V, E), \delta, t$)

(1) For $i := 1, \ldots, t$:

(a) Sample each vertex independently with probability $n^{-\delta/2}$. Denote the set of sampled vertices by M. Randomly distribute the vertices of M to the machines.

(b) For each sampled vertex v traverse the cycle in each direction until a sampled vertex is reached. Let l_v and r_v be the sampled vertices, where the each of two traversals finishes.

(2) Return the graph $(M, \{xl_x \mid x \in M\} \cup \{xr_x \mid x \in M\})$.

ALGORITHM 2: 2-CYCLE($G = (V, E)$)

(1) $G' :=$ SHRINK($G, \epsilon, O(1/\epsilon)$)

(2) Solve the 2-CYCLE problem on G', which has size $O(n^{\epsilon})$ w.h.p. on a single machine.

Adaptive Massively Parallel Model (AMPC)

What the AMPC can do better than the MPC

Lemma.

Let G be a graph consisting of cycles and $n := |V(G)|$. Fix an iteration from $SHRINK(G, \varepsilon, O(1/\varepsilon))$.
If the size of a cycle was $\Omega(n^{\varepsilon})$ at the beginning of the iteration, its size shrinks by at least $\Omega(n^{\varepsilon/2})$ times after the iteration w.h.p.

Lemma.

After the shrink operation, the length of each cycle becomes $O(n^{\varepsilon})$ w.h.p.

Lemma.

Total communication of each machine is $O(n^{\varepsilon})$ in each round w.h.p.

Adaptive Massively Parallel Model (AMPC)

What the AMPC can do better than the MPC

Problem	AMPC	MPC
Connectivity	$O(\log\log_{m/n} n)$	$O(\log D + \log\log_{m/n} n)$ [10]
Minimum spanning tree	$O(\log\log_{m/n} n)$	$O(\log n)$
2-edge connectivity	$O(\log\log_{m/n} n)$	$O(\log D \cdot \log\log_{m/n} n)$ [2]
Maximal independent set	$O(1)$	$\widetilde{O}(\sqrt{\log n})$ [25]
2-Cycle	$O(1)$	$O(\log n)$
Forest Connectivity	$O(1)$	$O(\log D \cdot \log\log_{m/n} n)$ [2]

Fig. 1. Round complexities of our algorithms in the AMPC model compared to the state-of-the-art MPC algorithms. D denotes the diameter of the input graph. We consider the setting where the space per machine is sublinear in the number of vertices, that is $S = O(n^{\epsilon})$ for constant $\epsilon < 1$.

Adaptive Massively Parallel Model (AMPC)

What the AMPC can't do

Theorem. (Note: 2-Cycle problem refers to the original 2-Cycle problem)

- Any deterministic AMPC algorithm for 2-Cycle problem takes $\geq \frac{1}{2}\log_S(n/2)$ rounds
    - If $S = n^{\varepsilon}$, the number of rounds is $\Omega(1/\epsilon)$
- Any *good (informal)* randomized AMPC algorithm for 2-Cycle takes $\geq \frac{1}{2}\log_S(n/2)$ rounds
    - If $S = n^{\varepsilon}$, the number of rounds is $\Omega(1/\epsilon)$

References

- Computational Intractability: A Guide to Algorithmic Lower Bounds
  https://hardness.mit.edu/
- S. Behnezhad et al. Massively Parallel Computation via Remote Memory Access. *ACM Transactions on Parallel Computing, Vol. 8, No. 3*
- Massively Parallel Algorithms
  http://people.csail.mit.edu/ghaffari/MPA19/Notes/MPA.pdf