Laxman’s Comments on MPC Chapter

1. It might be nice to have some coverage of the *transitive closure bottleneck* (termed as such by Karp and Ramachandran in their survey). This is the phenomenon where a number of important problems, like directed reachability and SSSP and its many variants are in NC, but only due to using matrix squaring, which blows up the work (the processor-time product) to $O(n^3)$ using combinatorial matrix multiplication. The conjecture that many in this area believe is that these problems cannot be solved in NC work-efficiently (i.e., using $O(n + m)$ work).

BILL COMMENT: I have added this paragraph in the Further Directions section which is the last section:

An interesting parameter for parallel algorithms is work which is the product of Number-of-Processors and Time. Karp and Ramachandran [5] define the Transitive Closure Bottleneck to illustrate the issue. The bottleneck is that many important problems like directed reachability and Single-Source-Shortest-Path (and its many variants) are in NC (polynomial number of processors, logarithmic time) but only due to using matrix squaring which has work $O(n^3)$. This leads to a later conjecture that these problems require work $O(n + m)$ where $m$ is the number of edges.

2. Another model of interest in modern parallel computing is the binary-forking model. This was originally defined in CLRS in the chapter on parallel algorithms. A formal model was later given in BFGS’19 http://www.cs.cmu.edu/~guyb/paralgpapers/BlellochFinemanGuSun20.pdf and follow-up work has studied more low-depth algorithms in this model, as well as (relevant for this book), lower bounds on binary-forking programs:

https://epubs.siam.org/doi/pdf/10.1137/1.9781611976465.128

I put the following into the introduction:

A more recent, and more realistic, model of parallel computing is the binary-forking model. This was originally defined informally in the textbook by Cormen et al. [2] in the chapter on parallel algorithms. A formal model was later given developed by Blelloch et al. [1]. Goodrich et al. [3] did follow-up work on both (1) low-depth algorithms (2) lower bounds (especially of interest to readers of this book).

3. I’m very happy to chat more about these topics if you are interested in them,
or interested in adding something (a few sentences, some more pointers for the interested reader) to the book about them.

4. Page 426

For the description of the input/output format for MPC, the example makes it sound like there will be $n$ machines, each with space $s$, which would result in a total space of $ns$ (possibly much larger than $N$). Instead, how about just saying:

*At the end of the computation the answer will be stored there in a distributed manner. We give an example. If the problem is connected components then we want to assign to every vertex $v$ a number, $n_v$, such that two vertices are in the same component if and only if they are assigned the same number. In addition to the input machines there will be $n/s$ output machines, where for each vertex $v$ one of the machines stores the mapping $(v, n_v)$. (something like this). I think it’s also fine to be more vague here and say that at the end, the output is distributed across a set of machines, and one machine stores the fact that $v$’s number is $n_v$.*

BILL COMMENT: It seems like the real difference between our paragraph is that I said there is ONE output processor per node which is WRONG, and you either say there are $n/s$ output processors or are more vague about it. I think it would be a good place to REMIND the reader that the graph is MUCH bigger than the number of processors. How is this:

There will be some machines designated as **output machines**. At the end of the computation the answer will be stored at the output machines. We give an example. If the problem is connected components then we want to assign to every vertex $v$ a number $n_v$ such that two vertices are in the same component if and only if they are assigned the same number. We want to store all the pairs $(v, n_v)$. Since the number of nodes may be much larger than the number of processors the output machines will each store many (though that is bounded by the space $s$) $(v, n_v)$ pairs.

5. Page 427

The number of processors SHOULD BE The number of machines

BILL COMMENT: I found out I use ‘processor’ 9 times and ‘machine’ 83 times’ I have replaced ALL occurrences of ‘processor’ with ‘machine’
Is this definition of *strongly sublinear* (Definition 21.3) standard? The MPC notes (textbook) by Mohsen uses the terminology superlinear, near-linear, strongly sublinear

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

which is what I’m also used to reading about in the context of MPC graph algorithms. Why not just call this a sublinear MPC algorithm?

BILL COMMENT: I am sure that what I have is NOT standard- I am working off scribe notes from grad students who listened to Moh’s lecture. (Reminds me of the game of Telephone). ANYWAY, I am happy to do a change but I want to make sure its the right one. Are you suggesting replacing the term ‘strongly sublinear’ with ‘sublinear’?

When mentioning any PRAM algorithm in time $t$, is this true for any of the well-known PRAM models? I believe it is, but maybe a citation to the first simulation result would be useful here:

https://arxiv.org/abs/1101.1902

BILL COMMENT: I have been told that the simulation holds in all PRAM models and I believe it. I am not sure anyone has bothered to write it all down.

I have added

See Goodrich et al. [4] for a simulations of one of the PRAM models.

6. Page 429

Theorem 21.8 should be w.h.p. for the depth, and in expectation for the work.

BILL COMMENT: I put replaced this with:

There exists an algorithm in the PRAM model which, given a graph $G$ on $n$ vertices and $m$ edges, will, with high probability, solve MAXIMAL IND. SET in $O(\log n)$ depth and $O(m)$ work. The algorithm will always solve the problem, though it may (quite rarely) take more time or more work than specified.

BILL QUESTION: I just noticed that I am using the concept of WORK here. I do not want to do that- its the only place I do, and I want to only bring that up at the end (your point 1 above). SO- how many machines does Luby’s PRAM algorithm use? I THINK ITS $O(n)$.

7. Page 430

For Theorem 21.9 and 21.3.3, could a bit more be added about the high-level techniques or ideas? I’m more familiar with MIS, so I would wonder about the
pattern matching paper. If a similar level of detail to the connected components algorithm could be given, I think that would be very useful for the reader.

BILL COMMENT: Would be a good idea if the (1) the book wasn’t already 527 pages, (2) I don’t really know the MPC stuff, and (3) I am tired.

8. Page 431
unconditional lower bound SHOULD BE unconditional lower bounds
BILL COMMENT- DONE, THANKS!

9. Page 432
For each port: Was port defined? I was unsure here whether the number of input ports to each machine is equal to the number of output ports. Later, item (1) says that each machine has an input from each machine in a round before it. A figure illustrating a few layers with different widths might help here.

Can any $s$-Shuffle instance with width $M$ and using $R$ rounds also be simulated in MPC with $M$ machines with space $s$ and $R$ rounds?
SEE NEXT COMMENT.

10. Page 433
I had some trouble understanding this section without looking at the Roughgarden-Wang paper. Maybe some short high-level idea of the proof would be useful before the definitions?

BILL COMMENT- I looked at the RW paper. Given my (lack of) understanding of the material do not think I can do it justice so I think I will cut out the definition of shuffle and most of the proofs, and just SAY that you can

Can go from MPC to Shuffle to Polynomials.
and then do stuff with polynomials.

11. In Exercise 21.18, part 3, should $d$ be $n$?
BILL COMMENT- YES, I HAVE MADE THE CORRECTION.

12. Page 435
Before starting 21.5, maybe some discussion about the previous unconditional lower bounds could be added. In particular, commenting on the fact that the bounds are $\Omega(O(1))$ for the realistic setting of $s = O(N^\epsilon)$. Since we cannot
seem to make progress on super-constant unconditional lower bounds, work has been on conditional lower-bounds, which is covered next.

BILL COMMENT-
I added to the last theorem in the UNCOND LB section the case of $s = N^\epsilon$
I now begin the Cond LB section as follows:
The unconditional lower bound in Theorem ??3 was that if $s = N^\epsilon$ (which is a realistic value of $s$) then the number of rounds is $\Omega(1)$. This is a weak lower bound. Hence we now turn to conditional lower bounds.

BILL COMMENT: I say its a weak lower bound; however, what DO we think the lower bond on graph connectivity or monotone graph properties should be?
BILL COMMENT: Graph conn is an odd problem to look at for cond vs uncond since for cond we ASSUME that graph conn is hard via the 1 vs 2 cycle question. Not sure what to do about that...

13. Page 436
Perhaps a bit more about what a component-stable MPC algorithm is could be added? Isn’t all that is required is for the outputs of nodes in different connected components of the graph to be independent? I found the remark about viewing the MPC algorithm as a graph hard to parse here. Also, there is an update to the component-stable framework by Czumaj et al. from 2021 extends the original framework to be more general, and also describes some component-unstable MPC algorithms:

BILL COMMENT: We can probably take care of this with a brief discussion to get the wording just right.

14. Page 441
then $G(x)$ is the 1 cycles SHOULD BE then $G(x)$ is the 1 cycle
BILL COMMENT- FIXED.

15. Page 442
For future directions, regarding directed $s$-$t$ connectivity, the paper by Nanongkai/Scquizzato
lays out a very nice picture of our knowledge of the hardness of different problems in MPC under suitably defined reductions. The results IIRC mostly hold under $NC^1$ reductions.

BILL COMMENT

I have added the following to the first paragraph of Further Directions.

For an up-to-date survey of what is known about lower bounds on the MPC model see the paper of Nanongkai & Scquizzato [6].

References


