Parallel Graph Algorithms

Richard Peng
Georgia Tech
OUTLINE

• Model and problems
• Graph decompositions
• Randomized clusterings
• Interface with optimization
THE MODEL

The `scale up’ approach:
- Have a number of processors
- Shared memory

‘Typcial’ spec:
- 256 cores
- 1TB RAM

[Madduri-Bader-Berry-Crobak`07]: shortest path
THE MODEL

Better upper bounds under dubious(?) assumptions:
- Concurrent read/write, PRAM
- Unit cost memory

Split computation up across processors which then process them in parallel

Goal: prove that $k$ processors $\rightarrow$ factor $k$ speedups
MEASURING COST

Multi-processor:

Taking time time $t(n)$ with $p(n)$ processors

Work/depth:

• Work: total operations performed
• Depth: max chain of dependencies

Work efficient: work should be close to sequential algorithms

• $d(n) = t(n)$, $w(n) \leq t(n) \times p(n)$,
• Many other models for parallelism are inter-reducible with polylog($n$) overhead
GRAPH PROBLEMS

- Connectivity / Reachability
- Shortest paths
- Optimization: flows, matchings

P-complete under polylog depth reductions

Open: on k processors, provably obtain factor $O(k)$ speedups of directed $s \rightarrow t$ reachability over DFS

Formally: $O(m \log^{O(1)} n)$ work, $O(m \log^{O(1)} n/k)$ depth
In an undirected graph $G$, is $s$ connected to $t$?

Repeatedly:
- Each vertex picks a neighbor
- Contract edges

# of vertices halves per round
$O(\log n)$ rounds

(omitting details on contract): $O(\log^2 n)$ depth, $O(m\log n)$ work

- More powerful operation: contract graph
- Communication no longer on original edge, closer to CLIQUE than CONGEST
SHORTEST PATH VIA. MATRICES

Graph, undirected or directed
Find shortest path from s to t

Min-plus matrix multiplication:
\[ d(u, v) = \min_w d(u, w) + d(w, v) \]

Depth: \( \log^{O(1)} n \)
Work: \( O(n^3 \log n) \)

Open: better work-depth tradeoffs: e.g. \((1 + \varepsilon)\)-approximation in \( O(m \log^2 n \varepsilon^{-2}) \) work, \( O(n^{0.7}) \) depth
A COMPLETE CALL STACK

Transshipment: match sources to sinks, minimize total distance of paths (no capacity constraints)

[Sherman `16]
[Becker-Karrenbauer -Krinninger-Lenzen `16]: O(m^{1+a}) work, O(m^a) depth via:
- Gradient descent
- Divide-and-conquer
- Graph clustering/embedding

Outermost: optimization loops

Inner loops: layered partitions of graphs

Bottom level: clustering schemes
OUTLINE

• Model and problems
• **Graph decompositions**
• Randomized clusterings
• Interface with optimization
Hopset: add `short cut’ edges so that shortest path lengths are approximated by ones with few hops

[Klein-Subramanian `93][???: scaling methods give exact h-hop distances in $O(h)$ depth, $O(m \log n)$ work

HOPSETS
MATRIX SQUARING AS A HOPSET

Min-plus matrix multiplication:
\[ d(u, v) = \min_w d(u, w) + d(w, v) \]

\[ \text{Dist}^k(u, v) \]
- Length of shortest k-hop u-v path
- Can view as a u-v shortcut
DIFFICULTIES IN FINDING GOOD HOPSETS

Highly connected, O(1)-hop graph is dense, expensive

Long paths / tree, need many hop edges

Challenge: avoid paying $O(n)$ steps, each taking $O(n)$
EXISTENCE OF GOOD HOPSETS

[Miller-Xu][folklore?] connect $n$ random pairs by exact shortest path distances between them

$n^{1/2}$ vertices

$n^{1/2} \times n^{1/2}$ pairs, one hop expected

$n^{1/2}$ vertices

Implies $O(m \log n)$ work, $O(n^{1/2} \log^2 n)$ depth shortest path algorithm
CONSTRUCTING HOPSETS

- $\varepsilon$-net like: partition into clusters, connect centers
- (In undirected case) recurse on smaller clusters
## SOME PREVIOUS WORKS ON HOPSETS

Open:
- exact hopsets in nearly-linear work
- Polylog hopcount + size + work

<table>
<thead>
<tr>
<th>Hop count $H$</th>
<th>Size</th>
<th>Work</th>
<th>Depth</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{O}(n^{1/2})$</td>
<td>$\tilde{O}(n)$</td>
<td>$\tilde{O}(mn^{1/2})$</td>
<td>$\tilde{O}(H)$</td>
<td>[UY91, KS97] (directed)</td>
</tr>
<tr>
<td>$O(\text{poly log } n)$</td>
<td>$O(n^{1+\alpha})$</td>
<td>$\tilde{O}(mn^\alpha)$</td>
<td>$\tilde{O}(H)$</td>
<td>[Coh00]</td>
</tr>
<tr>
<td>$(\log n)^{O((\log \log n)^2)}$</td>
<td>$O(n^{1+O\left(\frac{1}{\log \log n}\right)})$</td>
<td>$\tilde{O}(mn^{O\left(\frac{1}{\log \log n}\right)})$</td>
<td>$\tilde{O}(H)$</td>
<td>[Coh00]</td>
</tr>
<tr>
<td>$O(n^{4+\alpha})$</td>
<td>$O(n)$</td>
<td>$O(m \log^{3+\alpha} n)$</td>
<td>$\tilde{O}(H)$</td>
<td>[MPVX`15]</td>
</tr>
<tr>
<td>$O(1)$</td>
<td>$O(n^{1+1/H})$</td>
<td>$O(m^2)$</td>
<td>polylog(n)</td>
<td>[EN `16]</td>
</tr>
</tbody>
</table>
OUTLINE

• Model and problems
• Graph decompositions
• **Randomized clusterings**
• Interface with optimization
KEY TOOL IN DECOMPOSING GRPAHS

Low Diameter Decompositions
• Partition of V into clusters $S_1, S_2, \ldots, S_k$ s.t.
• The diameter of each $S_i$ is at most $d$.
• $\beta m$ edges between clusters.

Typically parameters:
• $\beta = \log^{O(1)} n$,
• $d = O(\log n / \beta)$
Parallel variant of a clustering scheme in [Bartal `96]

- Each vertex \( u \) starts unit speed BFS at time \(-\text{Exp}(\beta)\)
- BFS stops at ‘owned’ \( v \), owns any ‘sleeping’ \( v \) reached.
EXP START TIME CLUSTERING ON GRID

\[ \beta = 0.002 \]

\[ \beta = 0.005 \]

\[ \beta = 0.01 \]

\[ \beta = 0.02 \]

\[ \beta = 0.05 \]

\[ \beta = 0.1 \]
ANALYSIS ON UNDIRECTED GRAPHS

More global view:
- Each vertex picks $\delta_u = -\text{Exp}(\beta)$
- $v$ assigned to $\arg\min_u \text{dist}(u) + \delta_u$

Diameter: w.h.p. $\min_u \delta_u \approx -O(\log n / \beta)$

$e = uv$ ‘cut’ only if first two BFSs reach $u$ within $O(1)$

‘Backward’ analysis, view from $u$:
- Only $\text{dist}(v, w)$ and $\delta_v$ affect the way things reach $u$
- Equivalent to star centered at $u$
ANALYSIS: VIEW GRAPH FROM MIDPOINT

First two BFSs reach $O(1)$ apart $\iff$ max and $2^{nd}$ max of $k$ copies of shifted $\text{Exp}(\beta)$ are within $O(1)$

$\text{Exp}(\beta)$ can be viewed as particle decay:
- Start from $2^{nd}$ last particle decayed
- Prob. of last one lasting $<O(1)$: $O(\beta)$

Difference $\sim \text{Exp}(\beta)$
OUTLINE

- Model and problems
- Graph decompositions
- Randomized clusterings
- Interface with optimization
ITERATIVE METHODS

Gradual convergence to solution

• Gradient descent
• Newton steps
• Mirror descent

Preconditioning: solve problem in $A$ by solving several problems in some $B \approx A$, ‘error removal’

[Sherman `13] [Kelner-Lee-Orecchia-Sidford `14]: given operator that $\alpha$-approximates maxflow for ANY demand $d$, can compute $(1 + \epsilon)$-approx maxflow in $O(\alpha^2 \log n \epsilon^{-2})$ calls

[Madry `10] [KLOS `13]: $\alpha = O(m^\theta)$ in $O(m^{1+\theta})$ time
Tree: unique s-t path, maxflow = demand / bottleneck edge

Multiple (exact) demands: flows along each edge determined via linear mapping, \(O(n)\) time

[Racke `01][Racke-Shah-Taubig `14]: ANY undirected graph has a tree that’s an \(O(\log^c n)\) approximator
RECENT: TRANSSHIPMENT PROBLEM

Generalization of shortest paths: match sources to sinks, minimize total distance of paths (no capacity constraints)

Open:
- O(m log^{O(1)} n) work in O(log^{O(1)} n) depth?
- Implications for shortest paths?

`precondition' using L_1 embeddings:
- [Sherman `16]: O(m^{1+a}) work, O(m^{a}) depth
- [Becker-Karrenbauer-Krinninger-Lenzen `16]: O(\epsilon^{-1} \text{polylog}(n)) distributed rounds
THE OTHER END: THE LAPLACIAN PARADIGM

Open: distributed Laplacian solvers?

Evidence in favor: [Ghaffari-Karrenbauer-Kuhn-Lenzen-Patt-Shamir`15] distributed undirected maxflow

Directly related:
Elliptic systems

Few iterations:
Eigenvalues, Eigenvectors,
Heat kernels

Many iterations / modify algorithm
Graph problems, Image processing
THE OTHER END: SPARSIFIED SQUARING

\[ I - A_1 \approx_\varepsilon I - A^2 \]
\[ I - A_2 \approx_\varepsilon I - A_1^2 \]
\[ \vdots \]
\[ I - A_i \approx_\varepsilon I - A_{i-1}^2 \]
\[ I - A_d \approx I \]

\approx : approximations of graphs

Open: more graph algorithms via sparsified squaring?

Algorithms involving repeated squaring

- NC algorithm for shortest path
- [Reingold `05] Logspace connectivity
- Multiscale methods
- [P-Spielman `14] Solving \( Lx = b \)
MAKING SQUARING FAST: SPARSIFICATION

Approximate a dense graph by a sparse one

[Koutis`14]: build and remove $O(\log n)$ spanners, repeat with random half of what’s left

• [Abraham-Durfee-Koutis-Krinninger-P`16]: can make dynamic
• [Miller-P-Vladu-Xu `15]: spanners via exp. start time clustering

Question: dynamic exponential time clustering and applications?
QUESTIONS

• Connections between PRAM and other models?
• Speeding up directed $s \rightarrow t$ reachability
• Tight(er) bounds on undirected hopsets?
• Translating PRAM algorithms to data structures?
• Faster transshipment?
• Distributed Laplacian solver / sparsified squaring?
• Numerical approach to more graph problems?