A Practical Parallel Algorithm for Diameter Approximation of Massive Weighted Graphs

Matteo Ceccarello

Joint work with
Andrea Pietracaprina, Geppino Pucci, and Eli Upfal

1 Università di Padova 2 Brown University
Problem definition

- Undirected, weighted graphs
- Find the **weighted diameter** of $G(V, E, w)$
  \[ \Phi(G) = \max_{u \in V, v \in V, \text{dist}(u, v) < \infty} \{ \text{dist}(u, v) \} \]

- The diameter is a fundamental topological parameter of a graph with important practical applications

Goal
Develop a parallel approximation algorithm that requires
- a number of rounds sublinear in the *unweighted* diameter
- linear space
Computing the diameter exactly

Solve *All Pairs Shortest Paths*

- Dijkstra’s algorithm from every node: $O(n \cdot (m + n \log n))$
- Floyd-Warshall’s dynamic programming algorithm: $O(n^3)$
- Repeated fast matrix multiplication: $O(n^\omega \log n)$

$\Omega(n^2)$ space/time is required
Previous work: approximation algorithms

\(\Delta\)-stepping

- [Meyer, Sanders JoA’03]
- Parallel Single Source Shortest Path algorithm
- Provides a 2-approximation to the diameter
- Linear space
- Running time lower bounded by unweighted diameter

MapReduce approximation algorithm for unweighted graphs

- [Ceccarello, Pietracaprina, Pucci, Upfal SPAA’15]
- Linear space
- Running time \textit{sublinear} in the unweighted diameter
Our Result

MapReduce algorithm for diameter approximation

- $O(\log^3 n)$ approximation factor
- Linear space
- Number of rounds sublinear in the unweighted diameter
MapReduce: a model of computation for Big Data

Characteristics

▶ Distributed computation on clusters of commodity machines
▶ Practical algorithms can only afford linear space

Model [Pietracaprina, Pucci, Riondato, Silvestri, Upfal ACM-ICS’12]

▶ An algorithm is expressed as a sequence of rounds, each transforming datasets of key-value pairs
▶ There are memory limits:
  ▶ $M_L$ memory available for local computation: sub-linear in the data size
  ▶ $M_A$ sum of local memories
A MapReduce round:

Data shuffle:

- $(k_1, v_1)$
- $(k_2, v_2)$
- $(k_3, v_3)$
- $(k_1, v_4)$
- $(k_1, v_5)$
- $(k_2, v_6)$
- $(k_3, v_7)$
- $(k_1, v_8)$

Aggregate space: $M_A$

Reduce $k_1$ + map (space $M_L$)
- $(k_1, w_1)$
- $(k_1, w_1)$

Reduce $k_2$ + map (space $M_L$)
- $(k_2, w_2)$
- $(k_2, w_3)$

Reduce $k_3$ + map (space $M_L$)
- $(k_3, w_4)$
- $(k_3, w_5)$
- $(k_2, w_6)$
- $(k_5, w_7)$
Idea

- Derive from the input graph a smaller auxiliary graph whose diameter can be computed on a single reducer
- Relate the diameter of the auxiliary graph to the one of the input graph
Diameter approximation: high-level strategy

\[ G = (V, E): \text{undirected graph, polynomial weights. } n = |V|, m = |E|, \text{weighted diameter } \Phi(G). \]

1. Compute a decomposition \( C \) of \( G \) into clusters of small radius centered at random nodes

2. Estimate diameter \( \Phi(G) \) from diameter \( \Phi(G_C) \), with \( G_C \) a suitable quotient graph derived from \( C \)

Remarks

- cluster granularity chosen so that \( G_C \) fits into local memory
- small radius \( \rightarrow \) low round complexity, better approximation
Random sampling does not work well

High-diameter, skinny regions may be hard to cover $\rightarrow$ large radius
Challenges

In order to attain small cluster radius we must

1. Build more clusters in *remote regions* of the graph

2. Avoid *heavyweight* edges in building the clusters
Algorithm $\text{CLUSTER}(\tau)$

Progressive clustering strategy [Ceccarello, Pietracaprina, Pucci, Upfal-SPAA’15]

1. Select new batch of $\tau$ cluster centers from uncovered nodes
2. Grow both old and new clusters until covering half of the uncovered nodes
3. Repeat steps 1-2 until complete coverage

$\Delta$-stepping (inspired by [Meyer,Sanders JoA’03])

- $\Delta \leftarrow$ guess on minimum cluster radius
- In each iteration of progressive clustering (Steps 1-2):
  - Use only light edges (weight $< \Delta$) and stop at radius $\Delta$
  - If desired coverage cannot be obtained then $\Delta \leftarrow 2\Delta$
Define

- $R_G(\tau)$: optimum radius for a $\tau$-clustering of $G$
- $\ell_X$: max number of edges in a min-weight path of weight $X$

Theorem

W.h.p. $\textsc{Cluster}(\tau)$ computes a decomposition $C$ of $G$ into $O(\tau \log^2 n)$ clusters

- max cluster radius: $O(R_G(\tau) \log n)$
- round complexity: $O(\ell_{R_G(\tau)} \log n)$
  on $\text{MR}(n^\epsilon, m)$ for any constant $\epsilon \in (0, 1)$. 
Graph $G$, weighted diameter $\Phi(G) = 16$
Diameter approximation: example

$\tau = 1$, $\Delta = 4$
Diameter approximation: example

$\tau = 1$, $\Delta = 4$
$\tau = 1, \Delta = 4$
\( \tau = 1, \Delta = 4 \)
$\tau = 1, \Delta = 4$
Diameter approximation: example

Quotient graph $G_C$

$\Phi(G_C) = 12$

$\Phi(G) \leq 12+4+2 = 18$ (vs 16)
Diameter approximation: main result

Theorem
For a given weighted graph $G$, w.h.p. we can compute an upper bound to $\Phi(G)$

- approximation ratio: $O(\log^3 n)$
- round complexity: $O(\ell_{RG}(\tau) \log n \log n)$
  on MR($n^\epsilon$, $m$), for any constant $\epsilon \in (0, 1)$.

Corollary: Graphs of bounded doubling dimension $b$

- $\Psi(G)$ is the unweighted diameter of $G$. The number of rounds is
  
  $O\left(\left\lceil \frac{\Psi(G)}{n^{\epsilon'/b}} \right\rceil \log^3 n\right)$

- Plain $\Delta$-stepping requires $\Omega(\Psi(G))$ rounds.
Diameter approximation: experiments

Experimental setup

- In-house cluster with 16 machines
- 18GB RAM / Intel i7 nehalem 4-core processor
- Spark MapReduce platform

Datasets

<table>
<thead>
<tr>
<th>Graph</th>
<th>$n$</th>
<th>$m$</th>
<th>$\Phi(G)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>roads-USA</td>
<td>23,947,347</td>
<td>29,166,673</td>
<td>55,859,820</td>
</tr>
<tr>
<td>roads-CAL</td>
<td>1,890,815</td>
<td>2,328,872</td>
<td>16,485,258</td>
</tr>
<tr>
<td>livejournal*</td>
<td>3,997,962</td>
<td>32,681,189</td>
<td>9.41</td>
</tr>
<tr>
<td>twitter*</td>
<td>41,652,230</td>
<td>1,468,365,182</td>
<td>9.07</td>
</tr>
<tr>
<td>mesh(S)*</td>
<td>$S^2$</td>
<td>$2S(S-1)$</td>
<td>†</td>
</tr>
<tr>
<td>R-MAT(S)*</td>
<td>$2^S$</td>
<td>$16 \cdot 2^S$</td>
<td>†</td>
</tr>
<tr>
<td>roads(S)</td>
<td>$\approx S \cdot 2.3 \cdot 10^7$</td>
<td>$\approx S \cdot 5.3 \cdot 10^7$</td>
<td>†</td>
</tr>
</tbody>
</table>

† the diameter depends on the size of the graph, controlled by $S > 1$. 

Scalability
We compare our algorithm (CLUSTER) with Δ-stepping.
Diameter approximation: experiments

Work

Approximation
Conclusions

- Approximation factor $O(\log^3 n)$: much better in practice
- Number of rounds sublinear in the unweighted diameter

Open problems

- Reduce the theoretical approximation factor
- Relate the approximation factor to the available resources
- Diameter approximation on directed graphs

Open source implementation

- crono.dei.unipd.it/gradias/
- github.com/Cecca/graphx-diameter

@mceccarello