Simplified External Memory Algorithms for Planar DAGs

Lars Arge
Duke University

Laura Toma
Bowdoin College

July 2004
Graph Problems

- Graph $G = (V, E)$ with $V$ vertices and $E$ edges
  - **DAG**: directed acyclic graph
  - $G$ is **planar** if it can be drawn in the plane so no edges cross

- Some fundamental problems:
  - BFS, DFS
  - Single-source shortest path (SSSP)
  - Topological order of a DAG
    - A labeling of vertices such that if $(v,u)$ in $E$ then $\mu(v) < \mu(u)$
Massive graphs

- Massive planar graphs appear frequently in GIS
  - Terrains are stored as grids or triangulations
  - Example: modeling flow on terrain
    - Each point is assigned a flow direction such that the resulting graph is directed and acyclic
    - To trace the amount of flow must topologically sort this graph

- Massive graphs stored on disk
  - Assume edge-list representation stored on disk

- I/O can be severe bottleneck
I/O Model [AV’88]

- Parameters:
  \[ N = V+E \]
  \[ B = \text{disk block size} \]
  \[ M = \text{memory size} \]

- I/O-operation:
  - movement of one block of data from/to disk

- Fundamental bounds:
  - \textbf{Scanning:} \( \text{scan}(N) = O\left(\frac{N}{B}\right) \) I/Os
  - \textbf{Sorting:} \( \text{sort}(N) = O\left(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{B}\right) \) I/Os

- In practice \( B \) and \( M \) are big
  \[ \frac{N}{B} < \frac{N}{B} \log_{\frac{M}{B}} \frac{N}{B} \ll N \]
Previous Results

- **Lower bound:** $\Omega(\min\{V,\text{sort}(V)\})$ (practically sort($V$))
- Not matched for most **general graph** problems, e.g.
  - **General undirected graphs**
    - BFS: $O\left(\sqrt{\frac{VE}{B}} + \text{sort}(E)\right)$ [MM’02]
    - SSSP: $O\left(\sqrt{\frac{VE}{B}} \frac{\log W}{w} + \text{sort}(E)\right)$ [MZ’03]
    - DFS: $O\left(((V + \frac{E}{B}) \log V + \text{sort}(E))\right)$ [KS’96]
  
  - **General directed graphs**
    - BFS, DFS, SSSP, topological order (DAG)
      $O\left(((V + \frac{E}{B}) \log V + \text{sort}(E))\right)$ [BVWB’00]

- **Sparse graphs** $E = O(V)$
  - Directed BFS, DFS, SSSP: $O(V)$ I/Os
Previous Results

- Improved algorithms for special classes of (sparse) graphs
  - Planar undirected graphs solved using
    - $O(\text{sort}(N))$ reductions
  - $O(\text{sort}(N))$ multi-way separation algorithm [MZ’02]

- Generalized to planar directed graphs
  - BFS, SSSP: $O(\text{sort}(N))$ [ATZ’03]
  - DFS: $O(\text{sort}(N) \log N)$ [AZ’03]
  - Planar DAG topological sort: $O(\text{sort}(N))$ [ATZ’03]
    - Computed using a DED of the dual graph
Our Results

• Simplified algorithms for planar DAGs

\[ O(\text{scan}(N)) \text{ I/Os} \]
separation \( \Rightarrow \) top order, BFS, SSSP

• This does not improve the \( O(\text{sort}(N)) \) known upper bound since computing the separation takes \( O(\text{sort}(N)) \) I//Os

• Previous algorithms take \( O(\text{sort}(N)) \) even if separation is given
Planar graph separation: R-partition

- A partition of a planar graph using a set $S$ of separator vertices into $O\left(\frac{N}{R}\right)$ subgraphs (clusters) of at most $R$ vertices each such that:
  - $O\left(\frac{N}{\sqrt{R}}\right)$ separators vertices in total
  - Each cluster is adjacent to $O\left(\sqrt{R}\right)$ separator vertices

- In external memory choose $R = B^2$
  - $O(N/B)$ separator vertices
  - $O(N/B^2)$ clusters, $O(B^2)$ vertices each and $O(B)$ boundary vertices
  - Can be computed in $O(\text{sor}(N))$ I/Os [MZ’02]
Planar SSPP

1. Compute a $B^2$-partition of $G$

2. Construct a substitute graph $G^R$ on the separator vertices such that it preserves SP in $G$ between any $u,v$ in $S$
   - replace each subgraph $G_i$ with a complete graph on boundary of $G_i$
   - for any $u,v$ on boundary of $G_i$, the weight of edge $(u,v)$ is $\delta_{Gi}(u,v)$

3. Solve SSPP on $G^R$

4. Find SSPP to vertices inside clusters

Computed efficiently using

- $G^R$ has $O(N/B)$ vertices and $O(N)$ edges
- Properties of the $B^2$-partition
A Topological Sort Algorithm

- Compute indegree of each vertex
- Maintain list Z of indegree-zero vertices
- Repeatedly
  - Number an indegree-zero vertex v
  - Consider all edges \((v, u)\)
    and decrement indegree of u
  - If indegree(u) becomes 0
    insert u in list Z

- \(O(1)\) I/O per edge \(\implies O(N)\) I/Os
Topological Sort using $B^2$-partition

1. Construct a substitute graph $G^R$ using $B^2$-partition
   - edge from $v$ to $u$ on boundary of $G_i$
     iff exists path from $v$ to $u$ in $G_i$

   • Lemma: for any separator vertices $u, v$
     if $u$ is reachable from $v$ in $G$, then $u$ is reachable from $v$ in $G^R$

2. Topologically sort $G^R$ (separator vertices in $G$)

   • Lemma: A topological order on $G^R$ is a topological order on $G$.

3. Compute topological order inside clusters
Topological Sort using $B^2$-partition

• Problem:
  – Not clear how to incorporate removed vertices from $G$ in topological order of separator vertices ($G^R$)

• Solution (assuming only one in-degree zero vertex $s$ for simplicity):
  – Longest-path-from-$s$ order is a topological order
  – Longest paths to removed vertices locally computable from longest-paths to boundary vertices
Topological Sort using $B^2$-partition

1. Construct substitute graph $G^R$
   - Weight of edge between $v$ and $u$ on boundary of $G_i$ equal to length of longest path from $v$ to $u$ in $G_i$
2. Topologically sort $G^R$
3. Compute longest path to each vertex in $G^R$ (same as in $G$):
   - Maintain list $L$ of longest paths seen to each vertex
   - Repeatedly:
     • Obtain longest path for the next vertex $v$
       in topological order
     • Consider all edges $(v,u)$ and
       update longest path to $u$ in $L$
4. Find longest path to vertices inside clusters
Longest path to vertices inside a cluster

- must cross the boundary of the cluster

\[ \text{LP}(v) = \max_{u \in \text{Bnd}(G_i)} \{ \text{LP}(u) + \text{LP}_{G_i}(u, v) \} \]
Analysis

Topologically sort $G^R$

- Maintain a list $L$ with the indegree of each vertex
- Maintain list $Z$ of indegree-zero vertices
- Repeatedly
  - Number an indegree-zero vertex $v$ from $Z$
  - Consider all edges $(v,u)$ and decrement indegree of $u$ in $L$
  - If indegree($u$) becomes 0 insert $u$ in list $Z$

- $G^R$ has $O(N/B)$ vertices and $O(N/B^2 \times B^2) = O(N)$ edges
- Each vertex: access its adjacency list $\implies O(N/B)$ I/Os
- Each edge $(v,u)$: update $L$ $\implies O(N)$ I/Os
  - Can be reduced to $O(N/B)$ using boundary set property
Analysis

• Boundary set in the $B^2$-partition
  *(Maximal) set of separator vertices adjacent to the same clusters*
  - A boundary set is of size $O(B)$
  - There are $O(N/B^2)$ boundary sets [F’87]
    *(ass. bounded degree)*

• We store $L$ so that vertices in the same boundary set are consecutive
  - Vertices in same boundary set have same $O(B)$ neighbors in $G^R$
  - Each boundary set is accessed once by each neighbor in $G^R$
  - Each boundary set has size $O(B)$

$\Rightarrow O(N/B^2) \times O(B) = O(N/B)$ I/Os
Conclusion and Open Problems

- Given a $B^2$-partition topological order can be computed in $O(\text{scan}N))$ I/Os

- Longest path idea can also be used to compute SSSP and BFS in the same bound

- Open problems:
  - Improved DFS on DAGs? (exploiting acyclicity)
    - Planar directed DFS $O(\text{sort}(N) \log N)$
Analysis

Compute longest path to each vertex in $G^R$ (same as in $G$):
- Maintain list $L'$ of longest paths seen to each vertex
- Repeatedly:
  - Obtain longest path for next vertex $v$ in topological order
  - Consider all edges $(v, u)$ and update longest path to $u$

- We store $L'$ so that vertices in the same boundary set are consecutive
  - Vertices in same boundary set have same $O(B)$ neighbors in $G^R$
  - Each boundary set is accessed once by each neighbor in $G^R$
  - Each boundary set has size $O(B)$

$\Rightarrow O(N/B^2) \times O(B) = O(N/B)$ I/Os