Survey of

Distributed Mining of Information Network

Luke Leslie, Mayank Pundir, Yosub Shin

CS512 Spring 2015 Survey Presentation
Agendas

- Centralized HIN Data Mining algorithms
- MapReduce based Graph Mining
- Distributed Graph Processing Frameworks
- Distributed Graph Dataflow Frameworks (GraphX)
- Partitioning Distributed Graphs
Motivation

• Today’s information networks can have > 1M vertices, > 1B edges
  • DBLP >1.8M papers
  • Facebook > 100M users
  • World Wide Web > 3B nodes

• Scalable computation has been studied only for homogeneous information networks

• Scalable computation on HIN is necessary
Background

- (Distributed) Homogeneous Information Networks Algorithms
  - e.g. PageRank, K-means, SimRank

- (Centralized) Heterogeneous Information Networks Algorithms
  - e.g. RankClus [Sun et. al. EDBT 2009], GNetMine [Ji et. al. ECMLPKDD 2010], RankClass [Ji et. al. SIGKDD 2011], TruthFinder [Yin et. al. TKDE 2008]
MapReduce based Graph Mining

• Examples of MapReduce graph mining algorithms:

  • PageRank on MapReduce
    [Dean et al. OSDI 2004]

  • Personalized PageRank on MapReduce
    [Bahamani et al. SIGMOD 2011]

  • Graph Algorithms using Filtering with MapReduce
    [Lattanzi et al. SPAA 2011]

  • K-means Algorithm on MapReduce
    [Zhao et al. CLOUD 2009]

  • Triangle Count on MapReduce
    [Suri et al. WWW 2011]
What is MapReduce?

- Programming model for processing large data in distributed manner
- Map: Takes scattered data, generate key-value pairs
- Reduce: Takes values grouped by key, and aggregates data

PageRank implemented in MapReduce

```python
def Map(key, val):
    p ← key.PageRank/|key.OutNeighbors|
    Emit (key, val)
    for v' ∈ key.OutNeighbors do
        Emit (v', p)
    end

def Reduce(key, values):
    v ← ∅, s ← 0
    for p ∈ values do
        if IsVertex(p) then
            v ← p
        else
            s ← s + p
        end
    end
    v.PageRank ← s
    Emit (key, v)
```
Limitations of MapReduce based Graph Mining

• Limited to homogeneous networks

• MapReduce not good for graph processing:
  • Some graph processing algorithm impossible to implement in MapReduce. e.g. gradient descent
  • Shuffle phase incurs unnecessary overhead
  • Implementing graph algorithms unintuitive vs. Gather-Apply-Scatter
Distributed Graph Processing Frameworks

1. Gather - Each vertex collects val. from incoming neighbors
2. Apply - Each vertex processes val. received in ‘Gather’ phase
3. Scatter - Each vertex propagates output of ‘Apply’ to neighbors

Pregel [Malewicz et al. SIGMOD 2010]
Graph Applications using GAS

- Examples of GAS algorithms:
  - PageRank
  - SimRank
  - Personalized PageRank
  - K-means
  - Triangle Count

PageRank implemented in GAS

[Becchetti et al. TKDD 2010] [Gonzalez et al. OSDI 2014]
Distributed Graph Processing Frameworks

Graph Processing Framework Classification

• Out-neighbor maintaining frameworks
  • Pregel [Malewicz et al. SIGMOD 2010], Giraph [giraph.apache.org], Hama [Seo et al. CloudCom 2010]

• In-neighbor maintaining frameworks
  • LFGraph [Hoque et al. TRIOS 2013]

• In- and out- neighbor maintaining frameworks
  • PowerGraph [Gonzalez et al. OSDI 2012], Distributed GraphLab [Low et al. VLDB 2012], GraphLab [Low et al. UAI 2010]

Distributed Graph Processing Frameworks

Limitations of Distributed Graph Processing Algorithms

- Assumes homogeneous vertices and edges
- Unable to exploit link restrictions b/w vertices of diff. types
- Only simple graph algorithms are shipped with the frameworks, and no complex algorithms available
  - e.g. SCAN\[Xu et al. SIGKDD 2007\], Modularity-Based Clustering\[Clauset et al. Physical Review E 2004\]
Distributed Graph Dataflow Framework

- Having separate systems for each view is difficult to use and inefficient

GraphX [Gonzalez et al. OSDI 2014]

- Built on top of Spark (data analytics framework)
- Great flexibility in constructing / processing graph
- A graph is defined as: \( G = \{V, E, P_V, P_E\} \)
- Property graph represented in RDD (Resilient Distributed Datasets)
- RDD: In-memory, read-only collection of objects constructed from disk, or other RDD
  [Zaharia et. al. NSDI 2012]
  - Lineage: fault tolerance, extremely efficient
Distributed Graph Dataflow Framework

Distributed Graph Representation in GraphX

- Edges partitioned in partitioning func.,
- Vertices partitioned in vertex id (Routing table copartitioned)
- Subgraphs are expressed by bit masks

GraphX Algorithms

**Data:** Initialized graph

```python
Pregel(pagerankGraph, initialMessage)(
  vertexProgram, sendMessage, messageCombiner)
```

**def vertexProgram(id, attr, msgSum):**

- **Data:** vertex id, vertex property, and the aggregated message
- **Result:** Updated vertex property according to the Apply phase of GAS

```python
(oldPR, lastDelta) ← attr
val newPR ← oldPR + 0.85 * msgSum
return (newPR, newPR - oldPR)
```

**def sendMessage(edge):**

- **Data:** An edge triplet
- **Result:** Check for the convergence, and send authority to destination vertex

```python
if edge.srcAttr..2 > tolerance then
  return
  Iterator((edge.dstId, edge.srcAttr..2*edge.attr))
else
  return Iterator.empty
end
```

**def messageCombiner(a, b):**

- **Data:** Two messages destined to same vertex
- **Result:** The aggregated value

```python
return a + b
```

PageRank implemented in GraphX [Gonzalez et al. OSDI 2014]
Distributed Graph Dataflow Framework

Limitations of Distributed Graph Dataflow Framework

- Heavyweight
- Only simple graph algorithms are shipped with the frameworks, and no complex algorithms available
- Implementing HIN algorithms is not trivial and can lead to inefficiency
Partitioning Distributed Graphs

Partitioning Distributed Graphs

• Partitioning in homogeneous networks
  • METIS [Karypis et al. IPPS 1996]

• Partitioning homo. networks in distributed systems
  • GraphLab, PowerGraph: Vertex cut, partitioning heuristics
  • LFGraph: Fast(non-intelligent) hash based partitioning

• Partitioning streaming homo. networks
  • Greedy heuristics (hashing, chunking, edge count, triangle count, modularity) [Stanton et al. KDD 2012],
    FENNEL greedy vertex assignment [Tsourakakis et al. WSDM 2014]
Partitioning Distributed Graphs

METIS

[Karypis et al. IPPS 1996]

How to coarsen a graph using a maximal matching

\[ G = (N, E) \]

- \( E_S \) is shown in red
- Edge weights shown in blue
- Node weights are all one

\[ G_c = (N_c, E_c) \]

- \( N_S \) is shown in red
- Edge weights shown in blue
- Node weights shown in black

- Coarsening Phase: Run maximal edge matching; Merge matched vertices
- Partitioning Phase: Partition with coarsened graph (cheaper)
- Uncoarsening Phase: Restore original graph

Partitioning Streaming Graph Algorithm

Average gain of each heuristic over Random partitioning

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>BFS</th>
<th>DFS</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avoid Big</td>
<td>AB</td>
<td>-27.3</td>
<td>-38.6</td>
</tr>
<tr>
<td>Balanced</td>
<td>B</td>
<td>-1.5</td>
<td>-1.3</td>
</tr>
<tr>
<td>Prefer Big</td>
<td>PB</td>
<td>-9.5</td>
<td>-18.6</td>
</tr>
<tr>
<td>Chunking</td>
<td>C</td>
<td>37.6</td>
<td>35.7</td>
</tr>
<tr>
<td>Deterministic Greedy</td>
<td>DG</td>
<td>57.7</td>
<td>54.7</td>
</tr>
<tr>
<td>Exp. Det. Greedy</td>
<td>EDG</td>
<td>59.4</td>
<td>56.2</td>
</tr>
<tr>
<td>Exp. Rand. Greedy</td>
<td>ERG</td>
<td>45.6</td>
<td>45.6</td>
</tr>
<tr>
<td>Exp. Triangles</td>
<td>ET</td>
<td>50.7</td>
<td>49.3</td>
</tr>
<tr>
<td>Greedy EvoCut</td>
<td>GE</td>
<td>60.3</td>
<td>58.6</td>
</tr>
<tr>
<td>Hashing</td>
<td>H</td>
<td>-1.9</td>
<td>-2.1</td>
</tr>
<tr>
<td><strong>Linear Det. Greedy</strong></td>
<td>LDG</td>
<td>76</td>
<td>73</td>
</tr>
<tr>
<td>Linear Rand. Greedy</td>
<td>LRG</td>
<td>46.4</td>
<td>44.9</td>
</tr>
<tr>
<td>Linear Triangles</td>
<td>LT</td>
<td>55.4</td>
<td>54.6</td>
</tr>
<tr>
<td>Randomized Greedy</td>
<td>RG</td>
<td>45.5</td>
<td>44.9</td>
</tr>
<tr>
<td>Balance Big</td>
<td>BB</td>
<td>67.8</td>
<td>68.5</td>
</tr>
<tr>
<td>Triangles</td>
<td>T</td>
<td>49.7</td>
<td>48.4</td>
</tr>
</tbody>
</table>

Heuristic performance of BFS over Watts-Strogatz Graphs

Streaming Graph Partitioning for Large Distributed Graphs [Stanton et al. KDD 2012]

Limitations of Existing Graph Partitioning Methods

- Don’t consider heterogeneous nature of networks
  - Placing certain vertex, edge types in same partition may benefit a mining algorithm
  - Splitting meta-paths using weak links may provide natural means to partition the data
Summary

• Centralized HIN Data Mining algorithms
  • RankClus, GNetMine, RankClass, TruthFinder

• MapReduce based Graph Mining
  • PageRank, Personalized PageRank, Filtering, K-means

• Distributed Graph Processing Frameworks
  • Gather-Apply-Scatter algorithm (Pregel)

• Distributed Dataflow Frameworks
  • GraphX: Built on Spark, RDD, Fault tolerance using Lineage

• Partitioning Distributed Graphs
  • METIS (homogeneous), Greedy heuristics, FENNEL (streaming homogeneous)
References (1/2)

References(2/2)

[42] X. Yin, J. Han, and P. S. Yu. Truth Discovery with Multiple Conflicting Information Providers on the Web. In IEEE Transactions on Knowledge and Data Engineering. IEEE, 2008.
Thank you!