Large Graph Mining

Recent Development, Challenges and Potential Solutions

SABRI SKHIRI / RESEARCH DIRECTOR EURA NOVA
THE SPEAKER

PASSIONATE BY COMPUTER SCIENCE, TECHNOLOGY & RESEARCH

Research director @ EURA NOVA

Make the link between Research & Customer challenges
Supervising 3 PhD thesis, 6 Master thesis with 3 BEL Universities

Head of the EU R&D Architecture for a Telco equipment provider

Guiding the transition from Telco to Service provider with new technologies

Committer on open source projects launched @ EURA NOVA

RoQ-Messaging, NAIAD, Wazaabi
Before starting
Ramp-up test to wake-up the room after lunch a Friday afternoon …

I will use persons to illustrate the topic in this tutorial
Can you give me their names?

Leonard Sheldon Moss Lary Page

Looks ready to start to learn about Graph Processing!
AGENDA

1 / Introduction

2 / Focus on two graph mining algorithms

3 / Introduction of Distributed Processing Framework

4 / Graph Data warehouse – an emerging challenge

5 / Conclusion
AGENDA

1 / Introduction

2 / Focus on two graph mining algorithms

3 / Introduction of Distributed Processing Framework

4 / Graph Data warehouse – an emerging challenge

5 / Conclusion
EXECUTIVE SUMMARY

Graph Mining needs another approach

Data Mining
Mature, algorithmic, libraries & products

New Needs
Linked data & reasoning on relationships

What do we need?
Is traditional data mining still applicable?

Graph Data Warehouse
Is traditional data warehouse still applicable?

Flat data, relational data, multi-dimensional data
No Linked data

Biology
Chemistry
Social Networks
Internet - Networks

Graph-based similarity
Algorithm re-design for graphs
Scalability for storage & processing

Conceptual modeling
Query
Processing Stack & materialization
Storage
LET’S START WITH DATA MINING

Process of discovering patterns or models of data. Those patterns often consist in previously unknown and implicit information and knowledge embedded within a data set [1]

Data mining techniques have been developed over the last 20 years. These techniques involve the process of analyzing data from different perspectives and summarizing it into useful information.

**Classification**
We position data in a pre-determined group.

**Pattern recognition**
We mine data to retrieve pre-determined patterns.

**Clustering**
Data are grouped within partitions according to criteria.

**Feature extraction**
We transform the input data into a set of features (data set reduction).

**Association**
Enables to link data between each other.

**Summarization**
Ranking such as page rank.
DATA MINING

Manages & processes data as a collection of independent instances

The Mining usually does not consider the global relations between the objects

Almost all clustering algorithms compute the similarity between all the pair of objects in the data set
Why the relationship matters?
Taking into account the relation between data in mining

Imagine to cluster people from their profiles
Why the relationship matters?
Taking into account the relation between data in mining

Imagine to cluster people not only from their profiles but also … by their social interactions

- More complete Data structure
- Greater expressive power
- Better model or real-life cases

New emergent industrial needs lead to deal with this kind of structured data
New Industry requirements
Need to structure and mine structured & linked data
1. Biochemical Networks

The metabolic pathways

http://biocyc.org
1. Biochemical Networks

Genetic regulation signal

Taking a systemic approach we end-up with a huge interaction graph.
1. Biochemical Networks

A biochemical network definition

\[ G(V,E) \]

\[ \begin{align*}
V &= V_G \cup V_R \cup V_P \cup V_C \\
E &= E_{\text{Reg}} \cup E_{\text{Trans}} \cup E_{\text{React}} \\
E_{\text{Reg}} &= \{V_P \times V_R\} \\
E_{\text{Trans}} &= \{V_G \times V_R\} \\
E_{\text{React}} &= \{V_C \times V_R\}
\end{align*} \]
1. Biochemical Networks

New emergent industrial needs

- What happens if I drop a compound in the system?  
  Drug simulation in drug design

- Find which genes are involved in the fat reduction pathway?  
  Genetic therapy

- Predict a metabolic pathway given a metabolic network and seed reactions  
  Subgraph extraction

- Predict a metabolic network from a genetic signature given a protein interaction graph & a regulation network
2. Chemical Databases

New emergent industrial needs

Database specifically designed to store chemical information.

Graphs are the natural representation for chemical compounds, most of the mining algorithms focus on mining chemical graphs.
2. Chemical Databases

New emergent industrial needs

A typical request: Structural similarity search

\( G_d = (V_d, E_d) \)

\( V_d = (\theta_1, \ldots, \theta_n) \)

Gd is the graph query

The objective is to maximize the probability that the ith \( \theta_i = \alpha \) knowing the measure a, b.

\[
\max \forall_i (P(\theta_i = \alpha \mid a, b)) \text{ with } \{\alpha \in V\}
\]
2. Chemical Databases

New emergent industrial needs

- **Structural indexing**
  Indexing the structural properties of the molecules

- **Structural similarity search**
  Similar molecules will have similar effects

- **3D molecule conformation**
  Based on similar molecule conformations

- **Structure-Activity-Relationship**
  How to modify the Structure for changing its activity
2. Chemical Databases

New emergent industrial needs

**Structure-Activity-Relationship**
Example of the sucralose where 3 hydroxyl groups have been replaced with Chloride (Cl)

Sugar $C_{12}H_{22}O_{11}$

Diet Sugar $C_{12}H_{19}Cl_3O_8$

http://en.wikipedia.org/wiki/Sucralose
3. Social network analytics

New emergent industrial needs

The Social Graph models the (direct or indirect) Social interactions between users.
3. Social network analytics

Example of Trust from a bipartite Graph

The Goal is to infer trust connections between actors in set A only connected through Item I

3. Social network analytics

Example of Trust from a bi-partite Graph

The Goal is to infer trust connections between actors in set A only connected through Item I

\[ J(u, v) = \frac{|N_u \cap N_v|}{|N_u \cup N_v|} \]

\[ D(i) = \left( \frac{2}{1 + e^{-deg(i)\sigma + 2\sigma}} - 1 \right) \]

\[ Trust(u, v) = \alpha + \beta J(u, v) + \gamma \left( 1 - \frac{\sum_{i \in S_I} D(i)}{|S_I|} \right) \]

Measure to compare similarity and diversity

Highly connected shared item will have higher distance values

3. Social network analytics

Example of Trust from a bi-partite Graph

3. Social network analytics

New emergent industrial needs

- People you may know
  Structural similarity based

- Trust computation on structural properties
  Used for accurate recommendation

- Collaborative filtering
  Tends to like what your friends like

- Influence management
  Used in marketing models
3. Social network analytics
Marketing model to influence users

SOCIAL KNOWLEDGE

TRADITIONAL MARKETING MODELS
Bolton 1998
Bolton & Lemon 1999

SOCIAL MODELS
Nitan & Libai 2011 / Singer 2012

INFLUENCE NETWORK
Able to predict much more accurately

> How to influence influencer to reach objectives

Viral marketing maven
Accurate churners

Product (content, services, etc.) adoption
Loyal user to reward to optimize the subscriber base

Decrease acquisition costs
3. Social network analytics

Building an interaction-based model for INFLUENCE

Vertex similarity distance
Edge weight computing
Betweenness centrality computation
Temporal analysis and version at vertex/edge

When all social interaction variables are considered within the same model we end up with a very powerful Social Profile model.
LET’S USE GRAPHS
Can I use the traditional data mining approaches?
Problem Statement

What changes with graphs?

**Similarity & Distances**
Must be graph-based

**Structural nature of the data model**
Makes mining algorithm more challenging to implement

**Scalability issue**
Most of the graph mining problems include significant graphs

Most of the existing graph mining algorithms deal with data in the main memory -> not possible anymore
Problem Statement

Let's position this tutorial

BSP approach
Using fully distributed approach
Google Pregel, Apache HAMA

Graph DB
Focus on storage & graph traversal
Neo4J, Dex, OrientDB

In-memory/MPI/HPC
Use multi-processors implementations
SNAP
Problem Statement
Let's position this tutorial

BSP approach
Using fully distributed approach
*Google Pregel, Apache HAMA*

Given a set of data mining algorithms, how can we adapt them to fully leverage the distributed processing approach?
Using the distributed way

The base data model is not the same anymore

The algorithm implementation will depend on the underlying distributed processing paradigm

- Mining algorithm
- (Distributed) graph processing
- Graph Model
- (Distributed) Storage
AGENDA

1 / Introduction

2 / Focus on two graph mining algorithms

3 / Introduction of Distributed Processing Framework

4 / Graph Data warehouse – an emerging challenge

5 / Conclusion
Graph Mining algorithms
Let’s see what a graph mining algorithm looks like
Page Rank
A ranking algorithm

Compute a ranking on every web page based only on the linkage structure

The web is a network of web pages
In addition to the page content, the page linkage represents a useful source of knowledge and information

Page Rank

Basic concepts

Authority: approximate by the number & the importance of pages pointing to the considered page
Either,

1. The surfer chooses an outgoing link of the current vertex uniformly at random, and follows that link to the destination vertex, or

2. it “teleports” to a completely random Web page, independent of the links out of the current vertex.

Intuitively, the random surfer traverses frequently “important” vertices with many vertices pointing to it.
Page Rank

Random surfer who browses the pages

Let $G = (V, E)$ be the web graph

The PageRank equation

$$PR(v) = \frac{(1 - p)}{|V|} + p \cdot \sum_{u \in d_{in}(v)} \frac{PR(u)}{d_{out}(u)}$$

- Number of incoming edges to vertex $V$
- Number of outgoing edges from vertex $u$
- The dumping factor (0.85)

We will see how to implement it in a distributed processing framework in the 2\textsuperscript{nd} part of this tutorial
Graph clustering

Introduction

Probably the most important topic studied in graph mining
Graph area: referred as community detection

Goal
Given a set of instances, grouping them into groups which share common characteristics based on similarity

Graph clustering

Example in targeting advertisement

Let us see 2 kind of clustering algorithms

(1) Generalization of K-Means & (2) divide algorithm that uses the structure
K-Means based clustering

The original algorithm concept

**Goal**: finding cluster by minimizing the sum of the distances between the data instances and the corresponding centroid

**The k Number of groups**

**A similarity measure**

\[ D : (o_i, o_j) \rightarrow \mathbb{R} \]

**Steps**

1. Select K instance as initial centroids
2. Each data instance is assigned to the nearest cluster
3. Each cluster **center** is recomputed as the average of the data instance in the cluster
4. Repeat step [2-3]
Adapting K-Means to Graph model

What do we need to change?

Extending K-Means to take advantage of the linkage information

A Graph-aware selection of the vertex center

A Graph-aware similarity measure

\[ D : (o_i, o_j) \rightarrow \mathbb{R} \]

Median Vertex

Minimizes the sum of distances to all other vertices

\[ v_m = \min_{v \in C} \sum_{u \in C} D(u, v) \]

The Simplest is the geodesic distance

Number of edges (hops)
Adapting K-Means to Graph model

What do we need to change?

Extending K-Means to take advantage of the linkage information

A Graph-aware selection of the vertex center

A Graph-aware similarity measure

\[ D : (o_i, o_j) \rightarrow \mathbb{R} \]

Closeness Centrality

A node is the more central the lower its total distance to all other nodes

\[ CC(v) = \frac{|V| - 1}{\sum_{v \neq u, v \in V} D(v, u)} \]

The Simplest is the geodesic distance

Number of edges (hops)

We usually take the shortest path as distance

Centrality-based clustering
A divide method

From the graph, iteratively cut specific edges
Progressively cut into smaller communities

[1] proposed to use the edge betweenness centrality to select the edges to be cut

The cutting strategy should select the edges connecting as much as possible communities

Edge betweenness centrality

Definition

Locates structurally the “well-connected” edges
If it is located on many shortest paths

$$BC(e) = \sum_{v, w \in V} \frac{b_{vw}(e)}{b_{vw}}$$

$$B_{vw}(e) = \text{the number of shortest paths from } V \text{ to } W \text{ through } e$$
$$B_{vw} = \text{the total number of shortest paths from } V \text{ to } W$$

Centrality-based clustering
Step by step description

Steps
1. Compute the betweenness of all existing edges
2. Remove the edge with the highest betweenness centrality
3. Repeat step [1,2] until the communities are suitably found

\[
BC(e) = \sum_{v,w \in V} \frac{b_{vw}(e)}{b_{vw}}
\]

Extremely useful for web & social graphs
Characterized by Small-World structure property

AGENDA

1 / Introduction

2 / Focus on two graph mining algorithms

3 / Introduction of Distributed Processing Framework

4 / Graph Data warehouse – an emerging challenge

5 / Conclusion
Scalability issues

Why do we need a distributed approach?

The graphs can reach a significant size ~ x100 millions nodes, x billion edges

Most of the Graph mining frameworks & libraries use in-memory graph data => we need another paradigm
(really) Short introduction to distributed computing
How to distribute a processing over a huge data set?

The ability to run simultaneously software in different processors in order to increase its performance while the distributed concept emphasizes the notion of loosely coupling between those processors.
Distributed architectures
From the resource sharing & the paradigm viewpoint

- Shared memory
- Shared Disks
- Share Nothing

Explicit parallel programming
Implicit parallel programming
Shared memory
Distributed architecture

Distributed systems that share a common memory space
Case of distributed machine, it can be a distributed cache

Pros
High speed transfer

Cons
The shared memory must manage the data consistency &
The access from different clients
Can be costly when adding a new memory nodes
Can be highly expensive
Shared disk
Distributed architecture

Distributed systems that share a common shared disk space
Typically through a LAN

Pros
Almost transparent for the applications
Less costly when adding new storage node

Cons
Access contention & data consistency issue when clients increase
Expensive
Distributed systems where each machine has its own memory space

Pros
Can be implemented on cheap or expensive server
With an adapted distributed processing framework the application does not need to deal with the distributed aspect
Highly elastic

Cons
Applications need to be re-designed
This kind of system needs to distribute the data

**Partitioning policy**

1 2 3 4 5

- This leads to the interesting concept of data locality
  - Executing a process where the data is located
Explicit parallel programming

Distributed architecture: programming model viewpoint

The developer will have to explicitly program the parallel aspects
Create tasks, synchronization, managing threads & processes, thread safe operation, etc.

Pros
Richer expressivity, give very low level control over the distributed processing (main pain point in Hadoop MR)

Cons
Serious complexity
Error-prone

Not advised solution
Implicit parallel programming
Distributed architecture: programming model viewpoint

The developer will NOT have to take of those details
The compiler or the framework handles all aspects related to parallel execution
The code to run, the scheduling, the location of execution, etc

Pros
Much more easy – hidden complexity
Highly scalable

Cons
Much less control on the execution as it is completely handled by the framework

Most of the examples we present here are Implicit programming with share nothing data resources
Let’s talk about graph processing
How can I process a graph using implicit parallel programming and a share nothing processing?
Map Reduce

The well known framework from Google & Hadoop its open source version

Created by Google to index crawled web pages

The 3 main strengths of Hadoop [1]

Data Locality
Can schedule a process where the data is

Fault Tolerant
Automatic re-scheduling of failing tasks

Parallel processing
On different chunks of data

Map Reduce

Short introduction – 2 main phases Map & Reduce

Main concepts

Map Phase

- The problem is partitioned into a set of smaller sub-problems
- Distributed over the worker in the cluster & processed independently

Reduce Phase

- All answers to all sub-problems are gathered from the worker nodes and then merged

The developer only focus on the algorithm but

Is it really suited for Graph Processing & mining?

Gives a simple way to deal with large data sets in completely distributed way

However… not really suited for Graph processing

1. Does not manipulate a Graph model – makes complex the algorithm
2. Is not suited for iterative processing

1 iteration = 1 MR
Requiring a lot of I/O, data migration, unnecessary computation
Few works have been done in this direction


Despite the improvements these solutions lack for graph based model since they deal with multi-dimension data.
Then comes Google with Pregel

Methods for dealing with linked structures using Map reduce concept

Providing a distributed computing framework dedicated to graph processing

Bulk Synchronous Processing (BSP) for graph processing

In a BSP model an algorithm is executed as a sequence a Supersteps separated by a global synch. point until termination.

In 1 Superstep a processor can:
1. Perform computation on local data
2. Send or receive messages

Concep of superstep@Pregel
Learning distributed graph processing framework

The vertices of the graph execute the same user defined function (compute) in //
Modification of the state of a vertex or its outgoing edges
Read messages sent to the vertex from previous supersteps
Send messages to other vertices that will be received in the next supersteps
Modification of the Graph Topology

Concep of superstep@Pregel
Learning distributed graph processing framework

How do I stop the processing?
Use the “Vertex Voting”
Each node votes to halt -> become inactive unless it receives a non-empty message

Inactive vertices are not involved in processing anymore.
The processing stops when all vertices are inactive.
Open source implementation of Pregel
Methods for dealing with linked structures using Map reduce concept

Apache Giraph
From Google Pregel
BSP for distributed graph processing

Distributed Graph Processing
Processing
HDFS
Let’s play with Giraph
Implementing a single source shortest path (SSP)
Re-thinking the SSP for Giraph Processing

Thinking in term of supersteps & messages

**Definition of the vertex value**
The distance to reach the current vertex from the source

**Definition of the messages**
Vertex sends its current value + edge weight

1. Init vertex value to larger possible value for all vertices except the source
2. On each step
   1. The vertex reads the message from its neighbor
   2. Each message contains the distance between the source & current vertex through the last vertex
   3. We take the min value between the current value & the received value
   4. Send the message to all neighbor as min distance + weighted edge
Re-thinking the SSP for Giraph Processing

Thinking in term of supersteps & messages

Input : Messages: Set of received messages

if currentVertex is the source then
    minimumDistance ← 0;
else
    minimumDistance ← ∞;
end

foreach message m ∈ Messages do
    minimumDistance ← minimum(minimumDistance, valueOf(m));
end

if minimumDistance ≤ valueOf(currentVertex) then
    valueOf(currentVertex) ← minimumDistance;
    foreach outgoing edge e from currentVertex do
        sendMessageTo(targetOf(e), (minimumDistance + valueOf(e)));
    end
end

VoteTohalt();
SSP for Giraph Processing

Let’s dive into the supersteps

**Input**: Messages: Set of received messages

- If `currentVertex` is the source then
  - `minimumDistance ← 0;`
- Else
  - `minimumDistance ← ∞;`
- End

- For each `message m ∈ Messages` do
  - `minimumDistance ← minimum(minimumDistance, valueOf(m));`
- End

- If `minimumDistance ≤ valueOf(currentVertex)` then
  - `valueOf(currentVertex) ← minimumDistance;`
  - For each outgoing edge `e from currentVertex` do
    - `sendMessageTo(targetOf(e), (minimumDistance + valueOf(e)));`
  - End

- VoteToHalt();
For a Geek like me, code is easier to get

```java
@Override
public void compute(Iterator<DoubleWritable> msgIterator) {
    // 1. init value
    if (getSuperstep() == 0) {
        setVertexValue(new DoubleWritable(Double.MAX_VALUE));
    }
    // 2. set the minimum distance to MAX
    double minDist = isSource() ? 0d : Double.MAX_VALUE;
    // 3. Read the messages and update the min distance
    // Check whether one of the previous node is nearer than the others
    while (msgIterator.hasNext()) {
        minDist = Math.min(minDist, msgIterator.next().get());
    }
    // 4. Check whether the current value is > than the min distance sent by a previous vertex
    if (minDist < getVertexValue().get()) {
        setVertexValue(new DoubleWritable(minDist));
        // 5. Send to my neighbor my shortest distance + weight on edge
        for (LongWritable targetVertexId : this) {
            FloatWritable edgeValue = getEdgeValue(targetVertexId);
            sendMsg(targetVertexId, new DoubleWritable(minDist + edgeValue.get()));
        }
    }
    voteToHalt();
}
```

https://github.com/apache/giraph

*Moss, IT Crowd*
Launching the code in Giraph

Just for information & Fun

```java
@ Override
public int run(String[] argArray) throws Exception {
    if (argArray.length != 4) {
        throw new IllegalArgumentException(
            "run: Must have 4 arguments <input path> <output path> " +
            "<source vertex id> <# of workers>"
        );
    }
    GiraphJob job = new GiraphJob(getConf(), getClass().getName());
    job.setVertexClass(getClass());
    job.setVertexInputFormatClass(SimpleShortestPathsVertexInputFormat.class);
    job.setVertexOutputFormatClass(SimpleShortestPathsVertexOutputFormat.class);
    FileInputFormat.addInputPath(job, new Path(argArray[0]));
    FileOutputFormat.setOutputPath(job, new Path(argArray[1]));
    job.getConfiguration().setLong(SimpleShortestPathsVertex.SOURCE_ID, Long.parseLong(argArray[2]));
    job.setWorkerConfiguration(Integer.parseInt(argArray[3]), Integer.parseInt(argArray[3]), 100.0f);
    if (job.run(true) == true) {
        return 0;
    } else {
        return -1;
    }
}
```
Let’s play with Giraph II
Implementing Page Rank
Re-thinking PageRank for Giraph Processing

Thinking in term of supersteps & messages

Definition of the vertex value

Definition of the messages

Remember the PageRank equation

\[
PR(v) = \frac{(1 - p)}{|V|} + p \cdot \sum_{u \in d_{in}(v)} \frac{PR(u)}{d_{out}(u)}
\]

3 Mins to think!
Re-thinking PageRank for Giraph Processing

Thinking in term of supersteps & messages

**Definition of the vertex value**
The PageRank tentative

**Definition of the messages**
The PageRank tentative divided by \( \#_{\text{out edges}} \)

Remember the PageRank equation

\[
PR(v) = \frac{(1 - p)}{|V|} + p \sum_{u \in d_{in}(v)} \frac{PR(u)}{d_{out}(u)}
\]
PageRank in Giraph

Dive into the algorithm

---

**Definition of the vertex value**

The PageRank tentative

**Definition of the messages**

The PageRank tentative divided by #out edges

1. Init vertex value with 1/Size of the Graph
2. On each step
   1. The vertex read the message from its neighbor
   2. Each message contains PR tentative of ingoing vertex
   3. Compute the page rank for the current vertex with p=0.85
   4. Send the message to all outgoing edges
   5. After a fixed number of supersteps (iterations), Vertex vote to halt

\[
PR(v) = \frac{(1-p)}{|V|} + p \sum_{u \in d_{in}(v)} \frac{PR(u)}{d_{out}(u)}
\]

---

PageRank algorithm distilled
A deeper look at the algorithm

```java
Input : Messages: Set of received messages
if NumberOfSuperstep ≥ 1 then
    sum ← 0;
    foreach message m ∈ Messages do
        sum ← sum + valueOf(m);
    end
    valueOf(currentVertex)← 0.15 / SizeOfGraph + 0.85 × sum;
end
if NumberOfSuperstep < MaximumNumberOfIteration then
    N = SizeOf({outgoing edges from currentVertex})
    sendMessageToAllNeighbors(valueOf(currentVertex) / N);
else
    VoteTohalt()
end
```
PageRank for Giraph Processing

For a Geek like me, code is easier to get

```java
@Override
public void compute(Iterator<DoubleWritable> msgIterator) {
    if (getSuperstep() >= 1) {
        double sum = 0;
        //Read the message from last step and sum them (second term in the equation)
        while (msgIterator.hasNext()) {
            sum += msgIterator.next().get();
        }
        //Compute the equation
        DoubleWritable vertexValue = new DoubleWritable((0.15f / getNumVertices()) + 0.85f * sum);
        //Set the page rank value
        setValue(vertexValue);
    }
    //Check iteration
    if (getSuperstep() < MAX_SUPERSTEPS) {
        long edges = getNumOutEdges();
        sendMessage(new DoubleWritable(getVertexValue().get() / edges));
    } else {
        voteToHalt();
    }
}
```

https://github.com/apache/giraph

*Moss, IT Crowd*
For the Geeks - what’s the meaning of the `sendMesgToAllEdges`?

```java
@Override
public final void sendMsgToAllEdges(final DoubleWritable msg) {
    if (msg == null) {
        throw new IllegalArgumentException(
            "sendMsgToAllEdges: Cannot send null message to all edges");
    }

    final MutableVertex<LongWritable, DoubleWritable, FloatWritable, DoubleWritable> vertex = this;
    verticesWithEdgeValues.forEachKey(new LongProcedure() {
        @Override
        public boolean apply(long destVertexId) {
            vertex sendMessage(new LongWritable(destVertexId), msg);
            return true;
        }
    });
}
```

https://github.com/apache/giraph

*Moss, IT Crowd*
Test: Write a classification Example

Up to you guys – Classification of customer by product

15 mins

Definition of the vertex value
Definition of the messages

1. Starting from n root nodes, each having one color
2. Propagate the color to all neighbor nodes
3. The color is propagated if there is no nearest root colored node
4. Use the SSSP to define the distance

```java
public enum Color {
    GREEN, RED, ORANGE
}
```
Test: Write a classification Example

Up to you guys – Classification of customer by product

1. Starting from n root nodes, each having one color
2. Propagate the color to all neighbor nodes
3. The color is propagated if there is no nearest root colored node
4. Use the SSSP to define the distance

```java
public enum Color {
    GREEN, RED, ORANGE
}
```
Test: Write a classification Example

Up to you guys – Classification of customer by product

Definition of the vertex value
[Color Label, Distance to the root node of this color]

```
public class ColorDistance implements Writable {
    private long distance = 0;
    private Color color = Color.RED;

    /**
     * Default constructor
     **/
    public ColorDistance(Color color, long distance) {
        this.color = color;
        this.distance = distance;
    }

    /**
     * return the distance between source color node and the current node */
    public long getDistance() { return this.distance; }

    /**
     * return the color of the vertex */
    public Color getColor() { return this.color; }

    @Override
    public void readFields(DataInput in) throws IOException {
        // ...
    }

    @Override
    public void write(DataOutput out) throws IOException {
        // ...
    }
}
```

Definition of the messages
[Color, Distance to the root node of this color]

```
public abstract class LongLongNullModifiedBloomVertex extends MutableVertex<LongWritable, ColorMessage, NullWritable, ModifiedBloomVertex> {
    /** Int represented vertex id */
    private long id;
    /** ColorDistance represented vertex value */
    private ColorMessage value;
    /** Int array of neighbor vertex ids */
    private long[] neighbors;
    /** ColorDistance array of messages */
    private List<ColorDistance> messages;

    @Override
    public LongWritable getVertexId() {
        return new LongWritable(id);
    }

    @Override
    public ColorDistance getVertexValue() {
        return new this.value;
    }

    @Override
    public void setVertexValue(ColorDistance vertexValue) {
        this.value = vertexValue;
    }
}
```
1. Init vertex value to larger possible value for all vertices except the source colored vertices
2. On each step
   1. The vertex read the message from its neighbor
   2. Each message contains the distance between the source & current vertex through the last vertex and the propagated color
   3. If the value is less than the received value we update the value and set the color
   4. Send the message to all neighbor as min distance + weighted edge

Definition of the vertex value
[Color Label, Distance to the root node of this color]

Definition of the messages
[Color, Distance to the root node of this color]
Test: Write a classification Example

Up to you guys – Classification of customer by product

```java
@Override
public void compute(Iterator<ColorDistance> msgIterator) {
    // Initialisation phase
    if (getSuperstep() == 0) {
        if (isSource()) {
            setVertexValue(new ColorDistance(Color.RED, Double.MAX_VALUE));
        }
    }
    double minDist = isSource() ? 0d : Double.MAX_VALUE;
    Color propagatedColor = Color.RED;

    // 2. Define the nearest Color recived
    while (msgIterator.hasNext()) {
        ColorDistance msg_i = msgIterator.next().get();
        if (msg_i.getDistance() < minDist) {
            propagatedColor = msg_i.getColor();
            minDist = msg_i.getDistance();
        }
    }

    // 3. Check the Vertex value
    if (minDist < getVertexValue().get()) {
        setVertexValue(new ColorDistance(msg_i.getColor(), minDist));
        // 4. Send to all neighbor the new distance and the color to propagate
        sendMsgToAllEdges(getVertexValue(propagatedColor, minDist + 1));
    }
    voteToHalt();
}
```
Intermediate Conclusion
Can I use graph mining algorithm on huge graphs using distributed framework coming from the web?
Intermediate Conclusion

Can we do graph mining on large graphs using the distributed approach?

Yes you can, but …

1. Need to choose a implicit distributed framework
2. This will constraint the programming model & the storage
3. Need to re-design the algorithm to fully exploit the framework

If I can mine the graph - does it mean that I have a data warehouse?

What do we miss to have a full graph data warehouse?
AGENDA

1 / Introduction

2 / Focus on two graph mining algorithms

3 / Introduction of Distributed Processing Framework

4 / Graph Data warehouse – an emerging challenge

5 / Conclusion
Links between Data Warehouse & Data Mining
Is it the same?
Data warehouse & mining

Definition of interactions

Data Mining algorithms are involved in many steps of the DW

1. Identifying key attributes
2. Finding related measures
3. Limiting the scope of queries

Mining space
Multi-dimensional cube space for mining

Generating features & target
By using OLAP queries

Multi-step OLAP process
Using data mining as building blocks

Speeding up model construction
Using data cube computation

OLAP framework are often integrated with mining frameworks

-> OLAM (On-Line Analytic Mining) & exploratory multi-dimensional mining [1]

Graph is fine but stop to play, be an adult

Come back in a professional & Business environment, come back to relational DB
The graph is a constraint
It is not because it is fun, it is because the relationship model brings a value

Let’s take the Social Network example

1. We can model a friend relationship in a m-n
2. In Average ~ 100 Friends
3. Friends of Friends request – 100^2 join requests

Storing a SN in a Relational DB is not a problem
Unless you need traversal queries for mining
A Graph in a relational DB

Two main important issues

1. Cost of Joins when traversing
2. Almost transferring the totality of the graph between the client and the DB

We have seen that Distributed Graph Processing frameworks use the data locality to minimize the cost

Data Application Server
I got a distributed processing framework & mining algorithms

Now do I have a Graph Data warehouse?

…BTW what is exactly a Data warehouse?

Aim at providing software, modeling approaches & tools to analyze a set of data in a collection of DB.
Conceptual modeling
An important topic of research

Aim at providing software, modeling approaches & tools to analyze a set of data in a collection of DB

Research topic focus
1. Improvement of the Snowflake & Star model
2. Models enabling the to define levels of hierarchies
3. Role played by a measure in different dimension
4. Properties such as additive, derive

Conceptual modeling
The multiDim model – a conceptual model for Data Warehouse & OLAP Applications

Hierarchy of dimensions

Cardinality child parent

Measures

Conceptual modeling reached a certain level of maturity

Extracting information by Queries

1. Rollup (increasing the level of aggregation)
2. Drill-down (decreasing the level of aggregation or increasing detail) along one or more dimension hierarchies
3. Slice and dice (selection and projection)
4. Pivot (re-orienting the multidimensional view of data).

Summary

Functional layers for OLAP

- **QUERY LAYER**
  - OLAP Cube
    - Snowflake Models & SQL request
- **TRANSLATION LAYER**
  - OLAP
    - OLAP Processing framework
- **PROCESSING FRAMEWORK**
- **STORAGE**
  - Relational
    - Traditional storage
I got a distributed processing framework & mining algorithms
Now do I have a Graph Data warehouse?!
Let’s take the Data warehouse process

Define what is missing if we have a graph model instead of a relational model

Global process overview

Sources

Data Warehouse

Dashboard

Aggregated Graph

ETL

Modeling

Queries

Need to be able to model intermediate structure keeping the relationship as a central place while Defining navigation path, roles in navigation, summarization pros, etc.
Why navigation path matters?

Central element in the traversal and then in graph mining

Define the way one could traverse the graph

- Item
- Person
- Group

Roles in paths

Hierarchies in paths

Additiveness in paths

Used in:
1. Classification
2. Ranking
3. Collaborative filtering

Members

Bought by
Bought

Friends of
Belongs to
Processing layers

Dealing with distributed frameworks while keeping an high level query layer

- QUERY LAYER
- TRANSLATION LAYER
- DISTRIBUTED PROCESSING FRAMEWORK
- GRAPH STORAGE
Challenges @Processing layers

Dealing with distributed frameworks while keeping an high level query layer

**QUERY LAYER**

What kind of query language to expose?
SQL - PigLatin – SPARQL?

**TRANSLATION LAYER**

How to infer a physical execution plan?
Data materialization issue is completely different from OLAP

**DISTRIBUTED PROCESSING FRAMEWORK**

How to deal with the distributed aspects?
Integration of the processing FWK?

**GRAPH STORAGE**

How to deal with the graph nature?
If I have a graph DB how do I use Giraph?
New Warehousing & OLAP multi-dimensional network model

A graph on which vertex = tuple in a table
Attributes of this table = multi-dimensional spaces

Zhao and al., Graph cube: on warehousing and OLAP multidimensional networks, in Proceedings of the 2011 international conference on Management of data
New Warehousing & OLAP multi-dimensional network model
A graph on which vertex = tuple in a table
Attributes of this table = multi-dimensional spaces

1. Shown we can execute **standard OLAP operations** while leveraging the graph aspects
2. Defined the algorithm to obtain the **aggregated networks** from queries
3. Present a **materialization approach**

Zhao and al., Graph cube: on warehousing and OLAP multidimensional networks, in Proceedings of the 2011 international conference on Management of data
Showing structural behaviors

Examples for operation on multi-dimensional networks

Summarizing on the multi-dimensional network on the dimension “Gender”

Drill-down operation
What is the network structure as grouped by both gender & location?

Summarizing on the multi-dimensional network on the dimensions “Gender” & “Location”

2 females in CA take 55.6% of the total Male-Female connections

Zhao and al., Graph cube: on warehousing and OLAP multidimensional networks, in Proceedings of the 2011 international conference on Management of data
Queries on GraphCube

1. The cuboid queries

Has as output the aggregate network corresponding to a specific aggregation of the multi-dimensional network

What is the network structure between various location & profession combinations?

The answer = the aggregated network in the desired cuboid in the graph cube

Figure 6: Traditional Cuboid Queries

Zhao and al., Graph cube: on warehousing and OLAP multidimensional networks, in Proceedings of the 2011 international conference on Management of data
Queries on GraphCube

2. Crossboid query

Queries which crosses multiple multi-dimensional spaces of the networks (Cuboids)

What is the network structure between the user “3” and various locations?

"What is the network structure between users grouped by gender and users grouped by location?"

"What is the network structure between users and the locations?"

Figure 7: Crossboid Queries Straddling Multiple Cuboids

Zhao and al., Graph cube: on warehousing and OLAP multidimensional networks, in Proceedings of the 2011 international conference on Management of data
New Warehousing & OLAP multi-dimensional network model

A graph on which vertex = tuple in a table
Attributes of this table = multi-dimensional spaces

1. Shown we can execute standard OLAP operation while leveraging the graph aspects
2. Defined the algorithm to obtain the aggregated networks from queries
3. Present a materialization approach

- Only consider vertex of the same type
- Only centralized processing
- Then materialization policy is inspired by legacy central DW

Zhao and al., Graph cube: on warehousing and OLAP multidimensional networks, in Proceedings of the 2011 international conference on Management of data
AGENDA

1 / Introduction

2 / Focus on two graph mining algorithms

3 / Introduction of Distributed Processing Framework

4 / Graph Data warehouse – an emerging challenge

5 / Conclusion
Conclusion

Today building **blocs exist** to mine large graphs
Up to you to assemble them for a dedicated purpose
Conclusion

Structuring linked data as graph is an **emerging & important** requirement

**Important challenges for Mining algorithms**
Adapting the logic to include the global relationship

**Important challenges for the processing layer**
Re-design algorithms – integrating the storage layer - using emerging Big data frameworks

However implicit distributed graph processing frameworks are emerging

**Still far from the concept of Graph Data Warehouse**
Lack of modeling – uniform stack – Query language – Re-design the materialization
THANK YOU

SABRI SKHIRI / RESEARCH DIRECTOR EURA NOVA

sabri.skhiri@euranova.eu / twitter@sskhiri /http://blog.euranova.eu
REFERENCES

Apache Mahout, Scalable machine learning
http://mahout.apache.org/

Apache Hadoop, Distributed computing framework
http://hadoop.apache.org/

Apache Giraph, open source implementation of Pregel implementation
http://incubator.apache.org/giraph/

NAIAD, open source implementation of Scala
http://naiad-processing.org

Cassandra, NoSQL column oriented storage
http://cassandra.apache.org/

HBase, NoSQL column oriented storage
http://hbase.apache.org/

PigLatin, high level query framework
http://pig.apache.org/

Scribe, log aggregator framework
https://github.com/facebook/scribe