Building a Framework for the Graph-theoretic Analysis of Dynamic Networks

Benjamin Schiller & Thorsten Strufe, P2P Networks, TU Darmstadt
SummerSim’13 / WIP / 08.07.2013
Motivation

compute benchmark
Outline

• Terminology

• Requirements

• Architecture

• Example

• Summary & outlook
Snapshot-based Graph Analysis

Snapshot Data 1
initialize
Graph G₁
Metric
calculate
Value v₁

t = 1

Snapshot Data 2
initialize
Graph G₂
Metric
calculate
Value v₂

t = 2

Snapshot Data 3
initialize
Graph G₃
Metric
calculate
Value v₃

t = 3
Stream-based Graph Analysis

Input Data

Graph $G_1$

Metric

Value $v_1$

$\text{initialize}$

$t = 1$

Updates $U_{1,2}$

Graph $G_2$

Metric

Value $v_2$

$\text{update}$

$t = 2$

Updates $U_{2,3}$

Graph $G_3$

Metric

Value $v_3$

$\text{update}$

$t = 3$
Graph Updates

\[
G_i = (V_i, E_i)
\]

\[
U_{i,j} := U_{i,j}^{E+} \cup U_{i,j}^{E-} \cup U_{i,j}^{V+} \cup U_{i,j}^{V-}
\]

\[
V_j := (V_i \cup U_{i,j}^{V+}) \setminus U_{i,j}^{V-}
\]

\[
E_j := (E_i \cup U_{i,j}^{E+}) \setminus U_{i,j}^{E-} \setminus (U_{i,j}^{V-} \times V_i) \cup (V_i \times U_{i,j}^{V-})
\]
Graph Updates - Example

\[ U_{i,j} = \{ \text{add}(a, e), \text{rm}(c, d), \text{rm}(d) \} \]

\[ G_i = G_{i,j}^0 \xrightarrow{u_1 = \text{add}(a, e)} G_{i,j}^1 \xrightarrow{u_2 = \text{rm}(c, d)} G_{i,j}^2 \xrightarrow{u_3 = \text{rm}(d)} G_{i,j}^3 = G_j \]
### Updating Metrics

1. **Before Single (BS)** \( G_{i,j}^{k-1}, u_k \in U_{i,j} \)

\[
U_{i,j} = \{ \text{add}(a, e), \text{rm}(c, d), \text{rm}(d) \}
\]

\[
G_i = G_{i,j}^0 \quad G_{i,j}^1 \quad G_{i,j}^2 \quad G_{i,j}^3 = G_j
\]

\[
u_1 = \text{add}(a, e) \quad u_2 = \text{rm}(c, d) \quad u_3 = \text{rm}(d)
\]
Updading Metrics

1. Before Single (BS) \[ G_{i,j}^{k-1}, u_k \in U_{i,j} \]

2. After Single (AS) \[ G_{i,j}^k, u_k \in U_{i,j} \]

\[ U_{i,j} = \{\text{add}(a, e), \text{rm}(c, d), \text{rm}(d)\} \]
Updating Metrics

1. Before Single (BS) \( G_{i,j}^{k-1}, u_k \in U_{i,j} \)

2. After Single (AS) \( G_{i,j}^k, u_k \in U_{i,j} \)

3. Before Batch (BB) \( G_{i,j}, U_{i,j} \)

\[
U_{i,j} = \{\text{add}(a, e), \text{rm}(c, d), \text{rm}(d)\}
\]

\[
\begin{align*}
G_i &= G_{i,j}^0 \\
G_i^1 &= G_{i,j}^1 \\
G_i^2 &= G_{i,j}^2 \\
G_i^3 &= G_{i,j}^3 = G_j
\end{align*}
\]

\[
\begin{align*}
u_1 &= \text{add}(a, e) \\
u_2 &= \text{rm}(c, d) \\
u_3 &= \text{rm}(d)
\end{align*}
\]
1. Before Single (BS) \( G_{i,j}^{k-1}, u_k \in U_{i,j} \)

2. After Single (AS) \( G_{i,j}^k, u_k \in U_{i,j} \)

3. Before Batch (BB) \( G_i, U_{i,j} \)

\( U_{i,j} = \{ \text{add}(a, e), \text{rm}(c, d), \text{rm}(d) \} \)

4. After Batch (AB) \( G_j, U_{i,j} \)

\[
G_i = G_{i,j}^0 \\
u_1 = \text{add}(a, e) \\
G_i = G_{i,j}^1 \\
u_2 = \text{rm}(c, d) \\
G_i = G_{i,j}^2 \\
u_3 = \text{rm}(d) \\
G_i = G_{i,j}^3 \\
G_i = G_j
\]
Updating Metrics

1. Before Single (BS) \( G_{i,j}^{k-1}, u_k \in U_{i,j} \)

2. After Single (AS) \( G_{i,j}^k, u_k \in U_{i,j} \)

3. Before Batch (BB) \( G_i, U_{i,j} \)
   \[
   U_{i,j} = \{ \text{add}(a,e), \text{rm}(c,d), \text{rm}(d) \}
   \]

4. After Batch (AB) \( G_j, U_{i,j} \)

5. Re-Computation (RC) \( G_j \)
   \[
   G_i = G_{i,j}^0 \xrightarrow{u_1 = \text{add}(a,e)} G_{i,j}^1 \xrightarrow{u_2 = \text{rm}(c,d)} G_{i,j}^2 \xrightarrow{u_3 = \text{rm}(d)} G_{i,j}^3 \equiv G_j
   \]
Framework Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>SNAP</th>
<th>Stinger</th>
<th>Graphstream</th>
<th>DNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph types</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>• Directed, undirected, multi-graph</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graph data structures</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>• Interchangeable implementations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metrics</td>
<td>✗</td>
<td>✗</td>
<td>(✓)</td>
<td>✓</td>
</tr>
<tr>
<td>• Support all 5 modes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes of input data</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>(✓)</td>
</tr>
<tr>
<td>• Stream, offline, synthetic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visualization</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>(✓)</td>
</tr>
<tr>
<td>• Metric values, graph</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benchmarking of algorithms</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>• Repeat execution, aggregate results</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Architecture

Stream

**Graph generator**

initialize

Graph

**Update generator**

compile

Batch

**Input Data**

Graph $G_1$

Metric Value $v_1$

**Updates**

$t = 1$

Graph $G_2$

Value $v_2$

$t = 2$

Graph $G_3$

Value $v_3$

$t = 3$

**Series run**

**Values**

**Statistics**

**Plotter**

**Graph generator**

**Update generator**

**Input Data**

Graph $G_1$

Metric Value $v_1$

**Updates**

$t = 1$

Graph $G_2$

Value $v_2$

$t = 2$

Graph $G_3$

Value $v_3$

$t = 3$

**Series run**

**Values**

**Statistics**

**Plotter**
Example

```java
DirectedGraphDatastructures ds = new DirectedGraphDatastructures(
    DirectedGraphAlAl.class, DirectedNodeAl.class,
    DirectedEdge.class);
Name = “ArrayList”;

Metric m1 = new ClosedTriangleClusteringCoefficientRecomp();
Metric m2 = new ClosedTriangleClusteringCoefficientUpdate();
Metric[] metrics = new Metric[] { m1, m2 };

int nodes = 200;
int edges = 200;
DirectedGraphGenerator gg = new RandomDirectedGraph(nodes, edges, ds);

int addEdges = 300;
DirectedBatchGenerator bg = new RandomDirectedEdgeAdditions(addEdges, ds);

int runs = 100;
int batches = 100;
Series s1 = new Series(gg, bg, metrics, "data/" + name + "/", name);
SeriesData sd = s1.generate(runs, batches);

SeriesData[] s = new SeriesData[] { sd };
Plotting.plot(s, "data/" + name + "-plots-avg/", PlotType.average);
Plotting.plot(s, "data/" + name + "-plots-conf/", PlotType.confidence2);
```
Example - Metrics
Example - Statistics
Example - Changing the Datastructures

```java
DirectedGraphDatastructures ds = new DirectedGraphDatastructures(
    DirectedGraphAlHs.class, DirectedNodeHs.class,
    DirectedEdge.class);
String name = "HashSet";
```
Summary & Outlook

• DNA - Dynamic Network Analyzer
  – Interchangeable graph data structures
  – Well-defined interfaces for metrics / generators
  – Result visualization
  – Benchmarking of algorithms
  – \url{http://www.p2p.tu-darmstadt.de/research/dna/}

• Work in Progress
  – Imprecisions on two levels (data structures vs. heuristics)
  – Integration with other systems
DNA - Building a Framework for the Graph-theoretic Analysis of Dynamic Systems
Implemented components

- Graphs and data structures
  - Directed, undirected
  - ArrayList / HashSet / HashMap / Array (nodes, edges, adjacencies)
- Graph generator
  - Random, ring, clique
- Update generator
  - RandomX, combinator, timeline
- Metrics
  - Degree, clustering coefficients
  - Rich-club, connected components
  - Shortest paths, betweenness centrality
- Series
  - Aggregation and run repition
- Plotter
  - Gnuplot interface for metrics & statistics
Series Generation Workflow

**Algorithm 1: Series generation**

**Input:** GraphGenerator gg, UpdateGenerator ug, Metric m, int runs, int batches

**Output:** Series data

```plaintext
foreach r ∈ [1..runs]do
    Graph g = gg.generate(); ▷ graph initialization
    m.init(g); ▷ initial computation
    m.write(run.$r/batch.0/metric.0/)
    Runtimes.write(run.$r/batch.0/runtimes)
    Statistics.write(run.$r/batch.0/statistics)
```

```plaintext
foreach b ∈ [1..batches]do
    Batch batch = ug.generate(); ▷ generate batch
    m.updateBeforeBatch(g, batch); ▷ BB mode
    foreach Update u : batch do
        m.updateBeforeUpdate(g, b, u); ▷ BS mode
        g.apply(u); ▷ update application
        m.updateAfterUpdate(g, b, u); ▷ AS mode
    end
    m.updateAfterBatch(g, batch); ▷ AB mode
    m.reComputation(g); ▷ RC mode
    m.write(run.$r/batch.$b/metric.0/)
    Runtimes.write(run.r/batch.b/runtimes)
    Statistics.write(run.r/batch.b/statistics)
end
end

Aggregation.write(aggr/);
```

---

Benjamin Schiller | P2P Networks | Technische Universität Darmstadt
Weights

\[ w^V_i : V_i \to \mathbb{R} \quad w^E_i : E_i \to \mathbb{R} \]

\[ U_{i,j} := U^E_{i,j}^+ \cup U^E_{i,j}^- \cup U^V_{i,j}^+ \cup U^V_{i,j}^- \cup U^V_{i,j}^w \cup U^E_{i,j}^w \]

\[ w_j(v/e) := \begin{cases} 
  w^V/E_i (v/e) & v/e \in V_i/E_i \\
  w^* & (v/e, w^*) \in U^E_w/V_w \\
  w^* & \text{otherwise}
\end{cases} \]
Filesystenm - Example

DNA - Building a Framework for the Graph-theoretic Analysis of Dynamic Systems