Towards a property graph generator for benchmarking

Arnau Prat-Pérez
Joan Guisado-Gámez
Xavier Fernández-Salas

Davide Basilio Bartolini
Siegfried Depner
Petr Koupy
Why a property graph generator?

- Graph-based analysis is becoming more and more popular
Why a property graph generator?

- For the field to advance, **many benchmarking initiatives** have appeared

- LDBC Social Network Benchmark
- Graphalytics
- gMark
- LUBM
- LinkBench
- GAP
- 500

**LDBC**
**Graphalytics**
**gMark**
**LUBM**
**LinkBench**
Why a property graph generator?

- Benchmarks need datasets, preferably **real ones**
Why a property graph generator?

- But ...
Why a property graph generator?

- But ...
Why a property graph generator?

- Synthetic graph generators
- However, each benchmark has specific data needs
  - each benchmark designer implements its own
  - *time consuming task* sometimes *reinventing the wheel*
Why a property graph generator?

- Tool that, given some “graph specification”, produces a synthetic graph with the specified characteristics

- DataSynth
  - https://github.com/DAMA-UPC/DataSynth
  - Written in Scala
  - Uses Apache Spark
Architecture Overview

Frontend
- **DSL Parser**
  - Scala based DSL with extensive use of code generation

Optimizer
- Execution Plan
- Optimizations possible for certain types of graphs

Backend
- **Apache Spark Runtime**
  - State of the art BigData framework
What features should DataSynth have?

- But what characteristics should a property graph generator be able to reproduce?
What features should DataSynth have?

- But what characteristics should a property graph generator be able to reproduce?

Properties and correlations/dependencies between them
- e.g. name is correlated with country
What features should DataSynth have?

- But what characteristics should a property graph generator be able to reproduce?

Properties and correlations/dependencies between them
- e.g. name is correlated with country

Variate Structure
- degree distributions
- community structure
- low diameter
- large connected component
- etc.
What features should DataSynth have?

- But what characteristics should a property graph generator be able to reproduce?

Variate Structure
- degree distributions
- community structure
- low diameter
- large connected component
- etc.

Properties and correlations/dependencies between them
- e.g. name is correlated with country

Property-Structure correlations/dependencies
- e.g. Chinese people tend to connect to Chinese people
  - represented as a $P(X,Y)$ of observing $X$ and $Y$ on a randomly picked edge.
What features should DataSynth have?

- But what characteristics should a property graph generator be able to reproduce?

Properties and correlations/dependencies between them
  - e.g. name is correlated with country

Property-Structure correlations/dependencies
  - e.g. Chinese people tend to connect to Chinese people
    - represented as a $P(X,Y)$ of observing $X$ and $Y$ on a randomly picked edge.

Variate Structure
  - degree distributions
  - community structure
  - low diameter
  - large connected component
  - etc.
But...

- Having a single algorithm for generating so many things seems too complex
  - Properties and property correlations
  - Realistic graph structure
  - Property-structure correlations

- There are tens of metrics to measure the structure of a graph, which ones to take (which possibly depend on the algorithms used)?
DataSynth's approach
DataSynth's approach

node property generation

<table>
<thead>
<tr>
<th>Id</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>China</td>
</tr>
<tr>
<td>2</td>
<td>Japan</td>
</tr>
<tr>
<td>3</td>
<td>China</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Germany</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lee</td>
</tr>
<tr>
<td>2</td>
<td>Hiroshi</td>
</tr>
<tr>
<td>3</td>
<td>Yang</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Wolfgang</td>
</tr>
</tbody>
</table>

structure generation

TIME
DataSynth's approach

<table>
<thead>
<tr>
<th>Id</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>China</td>
</tr>
<tr>
<td>2</td>
<td>Japan</td>
</tr>
<tr>
<td>3</td>
<td>China</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Germany</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lee</td>
</tr>
<tr>
<td>2</td>
<td>Hiroshi</td>
</tr>
<tr>
<td>3</td>
<td>Yang</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Wolfgang</td>
</tr>
</tbody>
</table>

node property generation

Matching preserving given joint probability distributions

\[ P(\text{China,China}) \approx 0.2 \]

structure generation

TIME

Person
<table>
<thead>
<tr>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>knows</td>
</tr>
<tr>
<td>date</td>
</tr>
</tbody>
</table>
DataSynth's approach

node property generation

<table>
<thead>
<tr>
<th>Id</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>China</td>
</tr>
<tr>
<td>2</td>
<td>Japan</td>
</tr>
<tr>
<td>3</td>
<td>China</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Germany</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lee</td>
</tr>
<tr>
<td>2</td>
<td>Hiroshi</td>
</tr>
<tr>
<td>3</td>
<td>Yang</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td>Wolfgang</td>
</tr>
</tbody>
</table>

Matching preserving given joint probability distributions

\[ P(\text{China},\text{China}) \approx 0.2 \]

edge property generation

<table>
<thead>
<tr>
<th>Id</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30/01/2015</td>
</tr>
<tr>
<td>2</td>
<td>4/06/2016</td>
</tr>
<tr>
<td>3</td>
<td>12/11/2016</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>30</td>
<td>03/03/2017</td>
</tr>
</tbody>
</table>
DataSynt's Approach

• Pros:
  − Accurate distributions of property values and correlations between properties
  − Does not limit us to a single way of generating the structure of a graph
    • We can use existing techniques and let the door open to new contributions
  − Pay for what we get

• Cons:
  − Heavy relies on a sophisticated matching approach to achieve accurate property-structure correlation
Property Generation

- We have a “Property Table” for each <type,property> pair
- We use a similar technique to that proposed by Myriad [1]
  - Highly parallel
  - Allows in-place data generation
    - Given and Id of an entity, I can generate its properties

Structure Generation

- We can use existing scalable graph generation techniques: BTER [1], Darwini [2], etc.
- Hadoop implementation of BTER implemented:
  - https://github.com/DAMA-UPC/BTERonH

Property-to-Structure Matching

Input

P(X,Y)

\[
\begin{array}{ccc}
0.3 & 0.067 & 0.067 \\
0.067 & 0.33 & 0.067 \\
0.067 & 0.067 & 0.17 \\
\end{array}
\]
Property-to-Structure Matching

Input

P(X,Y)

<table>
<thead>
<tr>
<th>0.3</th>
<th>0.067</th>
<th>0.067</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.067</td>
<td>0.33</td>
<td>0.067</td>
</tr>
<tr>
<td>0.067</td>
<td>0.067</td>
<td>0.17</td>
</tr>
</tbody>
</table>

Block Model

6,9

<table>
<thead>
<tr>
<th>9</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

4,5

2

6,9

2

7,10
Property-to-Structure Matching

Input

Block Model

Graph Partitioning
Next Steps

- Investigate further on the performance/quality of our Matching approach
  - Multithreaded/Distributed
  - Efficient for high-cardinality values
  - Understand when and when not works well
- Push for the DSL
- Integrate more existing structure generators
  - bi-partite graphs
- Long term: work towards “DGaaS” (Data Generation as a Service)