GGen: Random Graph Generation for Scheduling Simulations

Swann Perarnau, Denis Trystram, Jean-Marc Vincent

MOAIS/MESCAL Inria Teams
IUF, Grenoble University

ROADEF 2011
MOAIS/MESCAL Research Domain: HPC

Molecular Dynamics Simulation: Data/Communications Graph.
Part of the Grid5000 Experimental TestBed.
Motivation: Simulation of Scheduling Algorithms

Input Characteristics: Directed Acyclic Graph

- Vertices are tasks to execute.
- Edges are precedence constraints or communications.
- Additional annotations for costs.
Workload Characterization

Uniform Generation of Random Graphs
Combinatorial Approach.

Specific Classes of Random Graphs
Graphs respecting a set of well known properties.

Traces / Collected Workloads
Identified instances from real/academic environments.
Workload Characterization

Uniform Generation of Random Graphs

Impractical.

Specific Classes of Random Graphs

Graphs respecting a set of well known properties.

Traces / Collected Workloads

Identified instances from real/academic environments.
The Workload Generation Problem

Workload Characterization

Uniform Generation of Random Graphs

Impractical.

Specific Classes of Random Graphs

Graphs respecting a set of well known properties.

Traces / Collected Workloads

Hard to generalize results.
Workload Characterization

Uniform Generation of Random Graphs
Impractical.

Specific Classes of Random Graphs
Our focus.

Traces / Collected Workloads
Hard to generalize results.
The Workload Generation Problem

GGen Objectives

Challenges

- Implementations are rarely provided.
- Analysis of each classical method.

State of the project

- A framework to generate and analyze DAGs.
- In-depth analysis of generation methods and their influence on schedulers.
Outline

1. Generation Methods: an Overview
2. Schedulers Sensibility: A Case Study
3. GGen: A Graph Generation and Analysis Framework
4. Future Works
Outline

1. Generation Methods: an Overview
2. Schedulers Sensibility: A Case Study
3. GGen: A Graph Generation and Analysis Framework
4. Future Works
Layer-by-Layer [Kasahara et al., 2002]

Parameters

\( n \): Number of nodes.
\( l \): Number of layers.
\( p \): Probability to choose any possible edge.
Layer-by-Layer [Kasahara et al., 2002]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>e</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Parameters

\( n \) : Number of nodes.

\( l \) : Number of layers.

\( p \) : Probability to choose any possible edge.
Layer-by-Layer [Kasahara et al., 2002]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>b</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>d</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>e</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Parameters

- \( n \): Number of nodes.
- \( l \): Number of layers.
- \( p \): Probability to choose any possible edge.
Layer-by-Layer [Kasahara et al., 2002]

Parameters

\( n \) : Number of nodes.

\( l \) : Number of layers.

\( p \) : Probability to choose any possible edge.
Layer-by-Layer [Kasahara et al., 2002]

Parameters

\( n \) : Number of nodes.

\( l \) : Number of layers.

\( p \) : Probability to choose any possible edge.
Layer-by-Layer [Kasahara et al., 2002]

**Parameters**

- $n$: Number of nodes.
- $l$: Number of layers.
- $p$: Probability to choose any possible edge.

```
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
```
Random Orders [Winkler, 1985]

Parameters

\( n \): Number of nodes.

\( k \): Number of total orders to intersect.
Random Orders [Winkler, 1985]

Parameters

\( n \): Number of nodes.
\( k \): Number of total orders to intersect.
Random Orders [Winkler, 1985]

Parameters

$n$: Number of nodes.

$k$: Number of total orders to intersect.
Random Orders [Winkler, 1985]

Parameters

\( n \) : Number of nodes.

\( k \) : Number of total orders to intersect.
Random Orders [Winkler, 1985]

Parameters

\( n \) : Number of nodes.
\( k \) : Number of total orders to intersect.
Summary

Needs

- Many generation methods and variants.
- Analyze each method characteristics.

GGen first steps

- Provide a reference implementation
- Use Grid’5000 for big analysis campaigns
**Critical Path Analysis**

- **Erdos Renyi \( G(n,p) \) method**
  - Mean Longest Path vs. Probability to Create Edge
  - Number of Chosen Edges vs. Number of Chosen Edges

- **Erdos Renyi \( G(n,M) \) method**
  - Mean Longest Path vs. Number of Chosen Edges

- **Layer-By-Layer \( p=0.2 \)**
  - Layer-By-Layer \( p=0.8 \)
  - Mean Longest Path vs. Number of Layers

- **Random Orders method**
  - Number of Intersected Total Orders

**Experimental Design**

- **Sample Size:** 1,000
- **Number of nodes:** 100
- **Confidence Intervals:** 95%
Outline

1. Generation Methods: an Overview
2. Schedulers Sensibility: A Case Study
3. GGen: A Graph Generation and Analysis Framework
4. Future Works
Why generation parameters matter

Scheduler Sensibility
Small variations in the input lead to big differences in performance.

Experimental Design

- Select various schedulers,
- Measure their performance on a reference data set,
- Change the data set in subtle ways,
- Measure the variation in performance.
### Input Characteristics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>mean</td>
<td>sd</td>
<td>mean</td>
</tr>
<tr>
<td>$T_{small}$</td>
<td>500</td>
<td>746</td>
<td>27</td>
<td>9.98</td>
<td>5.7</td>
</tr>
<tr>
<td>$T_{moy}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{big}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Simulation Parameters

- Varying number of processors.
- Network is a complete graph.
- 1000 simulations per data point.
Base performance of schedulers

Figure: speed-up on a varying number of processors (on $T_{\text{small}}$).
Performance variation

Figure: Sensibility to computational costs, on $T_{small}$ modified with an exponential distribution.
Outline

1. Generation Methods: an Overview
2. Schedulers Sensibility: A Case Study
3. GGen: A Graph Generation and Analysis Framework
4. Future Works
Choose a scheduling algorithm.

2. Characterize the input data.

3. Choose a generation method.

4. Generate workload.

5. Check the quality of the input.

6. Simulate the scheduling algorithm.

7. Analyse results.
Classical Performance Evaluation Process

1. Choose a scheduling algorithm.
2. Characterize the input data.
3. Choose a generation method.
4. Generate workload.
5. Check the quality of the input.
6. Simulate the scheduling algorithm.
7. Analyse results.
GGen: Software Architecture

Transform | Analysis | Annotations | Generate
-----------|----------|-------------|---------
Graph      | Properties | Random     |
Igraph Library | GNU Scientific Library |
**Random Graph Generator**
Contains most classical methods.
Easily extensible code.
Standard output format: Graphviz DOT.

**Technical Info**
C Code under GPL compatible license.
Both a library and binaries utilities.
Publicly available at http://ggen.ligforge.imag.fr/

**Demo**
Available on demand during the conference.
Outline

1. Generation Methods: an Overview
2. Schedulers Sensibility: A Case Study
3. GGen: A Graph Generation and Analysis Framework
4. Future Works
Ongoing and Future Works

On Graph Generation

- More statistical studies.
- More graph classes.
- Integration of new methods as they appear.

On Simulations

- More distributions tested.
- Influence of generation parameters.
- Automated analysis.
Thanks

Thank you for your attention.

Demo available on demand.

GGen is publicly available at http://ggen.ligforge.imag.fr/