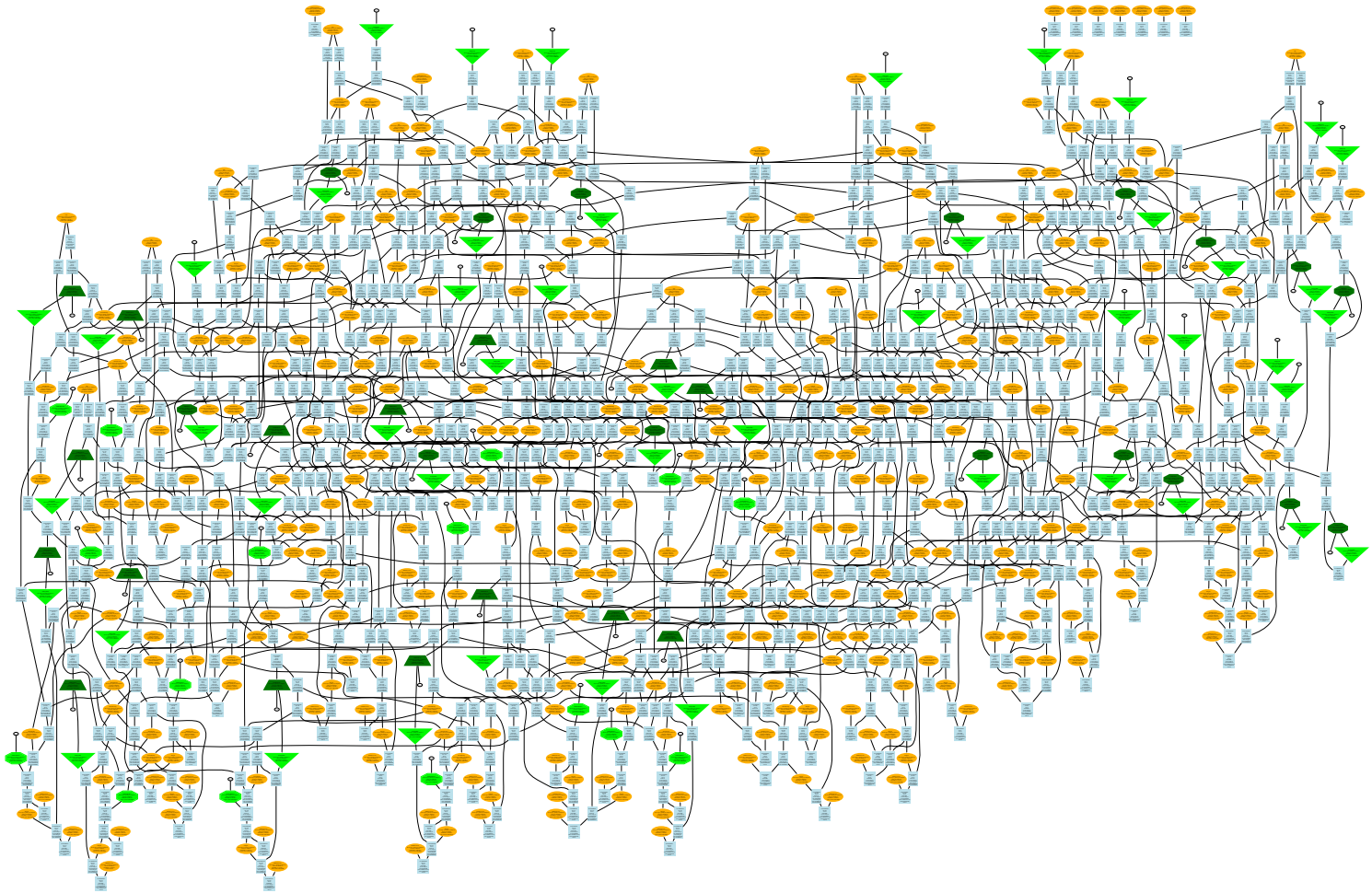
Random Graph Generation for Scheduling Simulations

Daniel Cordeiro, Grégory Mounié, **Swann Perarnau**, Denis Trystram, Jean-Marc Vincent, Frédéric Wagner

Moais and Mescal INRIA Teams, CNRS LIG Lab, Grenoble University
France

Simutools 2010

MOAIS/MESCAL Research Domain: HPC



Molecular Dynamics Simulation: Data/Communications Graph.

MOAIS/MESCAL Research Domain: HPC

Part of the Grid5000 Experimental TestBed.

Various Needs for Synthetic Workloads

Validation

- Unit Testing
- Random Testing

Performance Evaluation

- Structural Studies
- Quantitative Studies
- Expected Workloads

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

Even the count is unknown above 12 nodes.

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

Even the count is unknown above 12 nodes.

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

Even the count is unknown above 12 nodes.

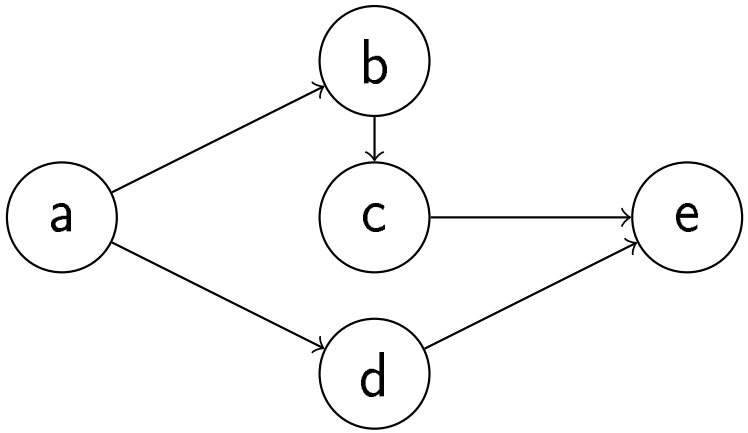
Specific Classes of Random Graphs

Our focus.

Traces / Collected Workloads

Hard to generalize results.

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.

Objectives

Goals

- Provide tools for the performance evaluation of schedulers.
- Ensure quality of those.

Challenges

- Analyze in details classical generation methods.
- Provide a reference implementation.

Main contribution

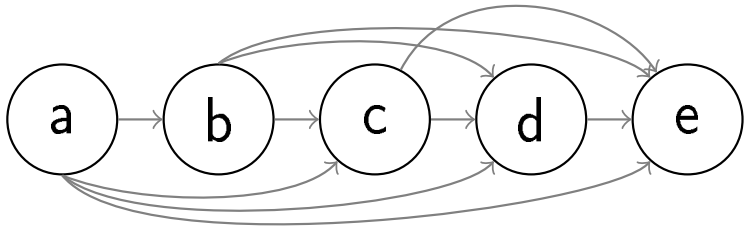
- GGen: a framework to build and analyze DAG generation methods.

Outline

Outline

Erdős-Rényi Methods: G(n, p) [Erdős, 1959]

	a	b	c	d	e
a	X	-	-	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



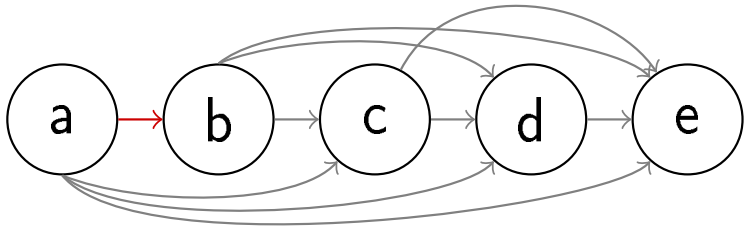
Parameters

n : Number of nodes.

p : Probability to choose any possible edge.

Erdős-Rényi Methods: G(n, p) [Erdős, 1959]

	a	b	c	d	e
a	X	-	-	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



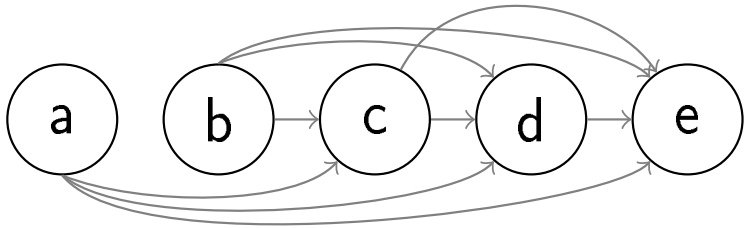
Parameters

n : Number of nodes.

p : Probability to choose any possible edge.

Erdős-Rényi Methods: G(n, p) [Erdős, 1959]

	a	b	c	d	e
a	X	0	-	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



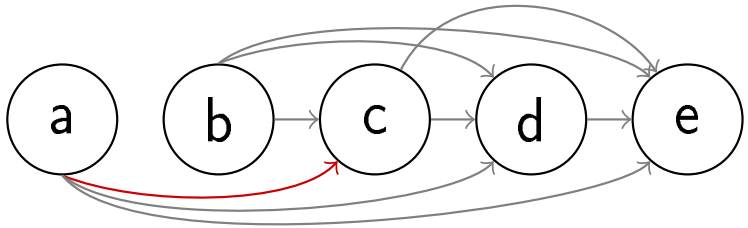
Parameters

n : Number of nodes.

p : Probability to choose any possible edge.

Erdős-Rényi Methods: G(n, p) [Erdős, 1959]

	a	b	c	d	e
a	X	0	-	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



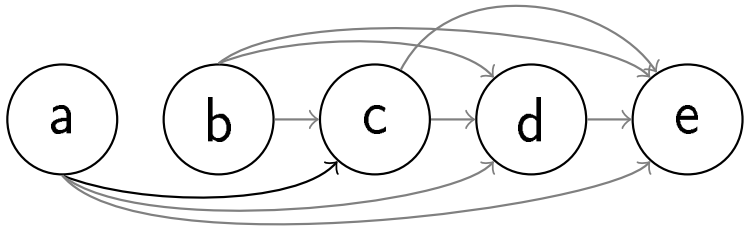
Parameters

n : Number of nodes.

p : Probability to choose any possible edge.

Erdős-Rényi Methods: G(n, p) [Erdős, 1959]

	a	b	c	d	e
a	X	0	1	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



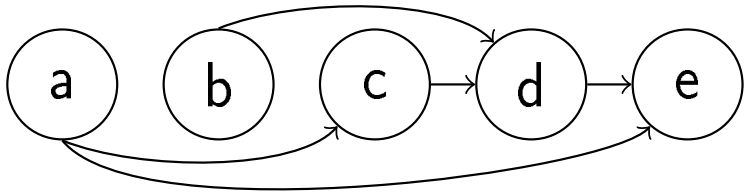
Parameters

n : Number of nodes.

p : Probability to choose any possible edge.

Erdős-Rényi Methods: G(n, p) [Erdős, 1959]

	a	b	c	d	e
a	X	0	1	0	1
b	X	X	0	1	0
c	X	X	X	1	0
d	X	X	X	X	1
e	X	X	X	X	X



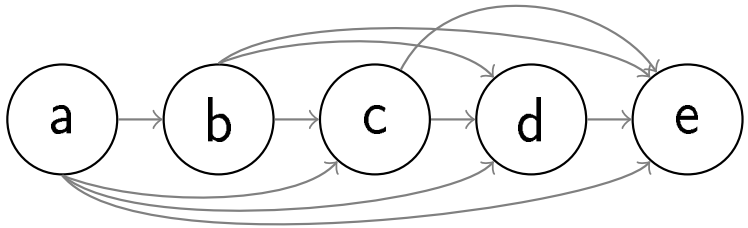
Parameters

n : Number of nodes.

p : Probability to choose any possible edge.

Erdős-Rényi Methods: G(n, M) [Erdős, 1959]

	a	b	c	d	e
a	X	-	-	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



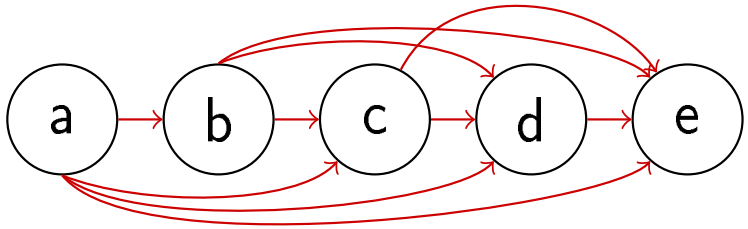
Parameters

n : Number of nodes.

M : Number of edges to choose uniformly from all the possible ones.

Erdős-Rényi Methods: G(n, M) [Erdős, 1959]

	a	b	c	d	e
a	X	-	-	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



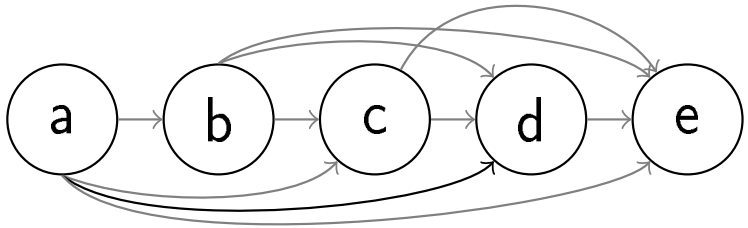
Parameters

n : Number of nodes.

M : Number of edges to choose uniformly from all the possible ones.

Erdős-Rényi Methods: G(n, M) [Erdős, 1959]

	a	b	c	d	e
a	X	-	-	1	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



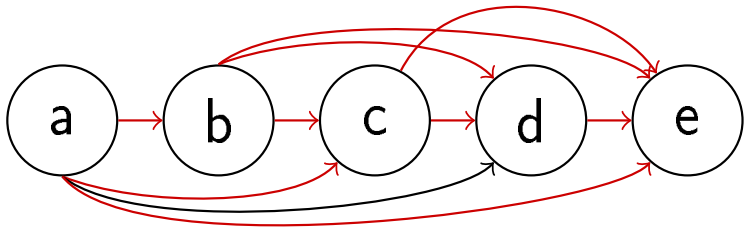
Parameters

n : Number of nodes.

M : Number of edges to choose uniformly from all the possible ones.

Erdős-Rényi Methods: G(n, M) [Erdős, 1959]

	a	b	c	d	e
a	X	-	-	1	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



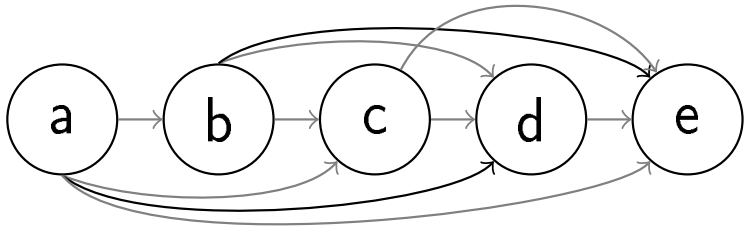
Parameters

n : Number of nodes.

M : Number of edges to choose uniformly from all the possible ones.

Erdős-Rényi Methods: G(n, M) [Erdős, 1959]

	a	b	c	d	e
a	X	-	-	1	-
b	X	X	-	-	1
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X



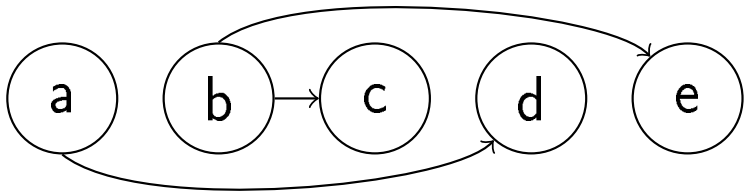
Parameters

n : Number of nodes.

M : Number of edges to choose uniformly from all the possible ones.

Erdős-Rényi Methods: G(n, M) [Erdős, 1959]

	a	b	c	d	e
a	X	0	0	1	0
b	X	X	1	0	1
c	X	X	X	0	0
d	X	X	X	X	0
e	X	X	X	X	X



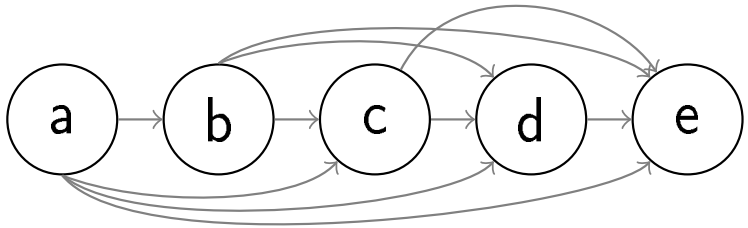
Parameters

n : Number of nodes.

M : Number of edges to choose uniformly from all the possible ones.

Layer-by-Layer [Kasahara *et al.*, 2002]

	a	b	c	d	e
a	X	-	-	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X

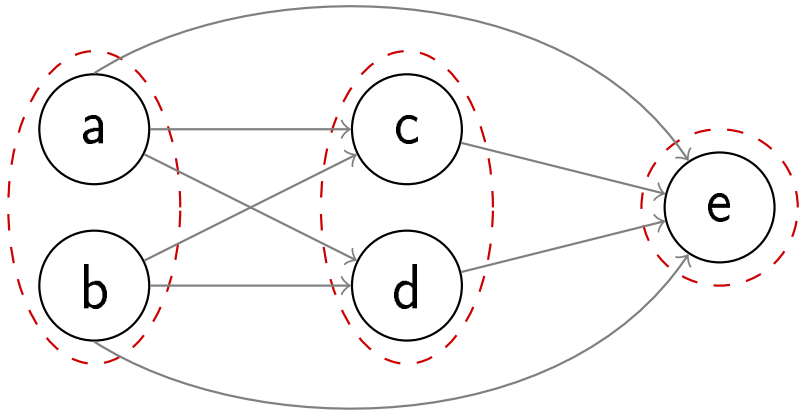


Parameters

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

Layer-by-Layer [Kasahara *et al.*, 2002]

	a	b	c	d	e
a	X	-	-	-	-
b	X	X	-	-	-
c	X	X	X	-	-
d	X	X	X	X	-
e	X	X	X	X	X

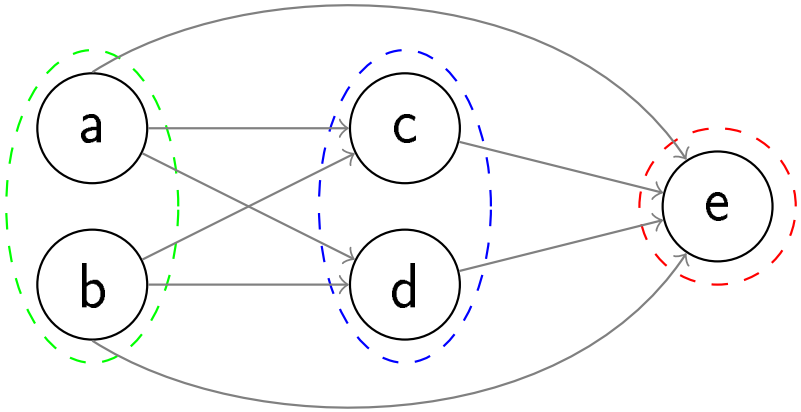


Parameters

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

Layer-by-Layer [Kasahara *et al.*, 2002]

	a	b	c	d	e
a	X	X	-	-	-
b	X	X	-	-	-
c	X	X	X	X	-
d	X	X	X	X	-
e	X	X	X	X	X



Parameters

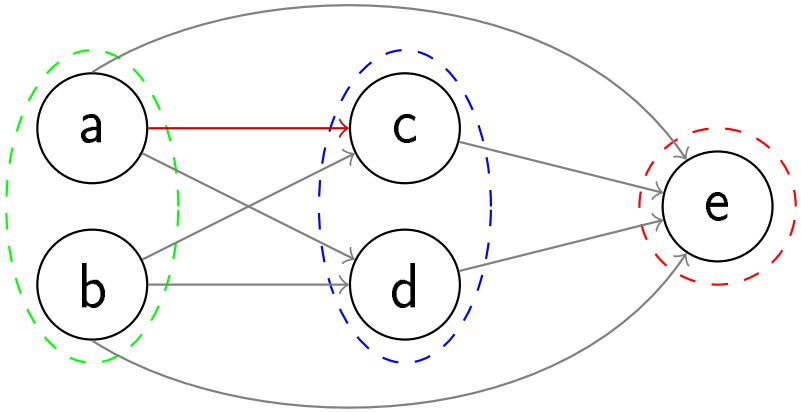
n : Number of nodes.

l : Number of layers.

p : Probability to choose any possible edge.

Layer-by-Layer [Kasahara *et al.*, 2002]

	a	b	c	d	e
a	X	X	-	-	-
b	X	X	-	-	-
c	X	X	X	X	-
d	X	X	X	X	-
e	X	X	X	X	X



Parameters

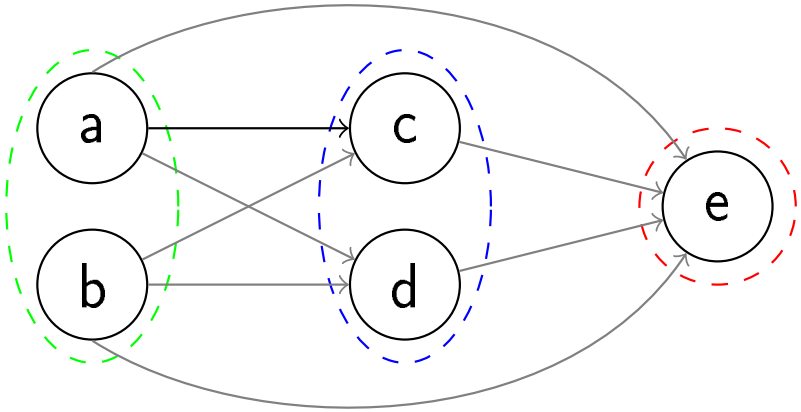
n : Number of nodes.

l : Number of layers.

p : Probability to choose any possible edge.

Layer-by-Layer [Kasahara *et al.*, 2002]

	a	b	c	d	e
a	X	X	1	-	-
b	X	X	-	-	-
c	X	X	X	X	-
d	X	X	X	X	-
e	X	X	X	X	X



Parameters

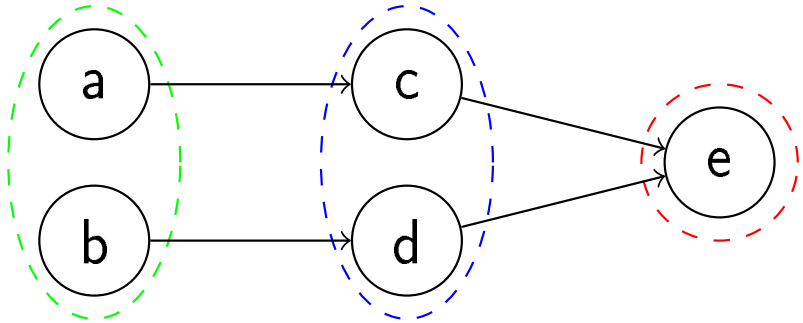
n : Number of nodes.

l : Number of layers.

p : Probability to choose any possible edge.

Layer-by-Layer [Kasahara *et al.*, 2002]

	a	b	c	d	e
a	X	X	1	0	0
b	X	X	0	1	0
c	X	X	X	X	1
d	X	X	X	X	1
e	X	X	X	X	X



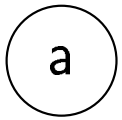
Parameters

n : Number of nodes.

l : Number of layers.

p : Probability to choose any possible edge.

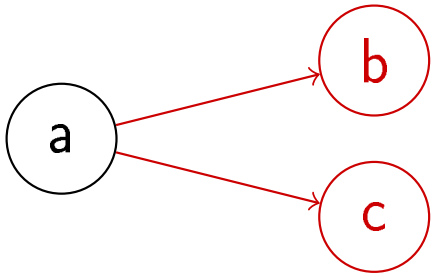
Fan-in/Fan-out [Dick *et al.*, 1998]



Parameters

n : Lower limit of the number of nodes.
od : Maximal out-degree allowed on each node.
id : Maximal in-degree allowed on each node.

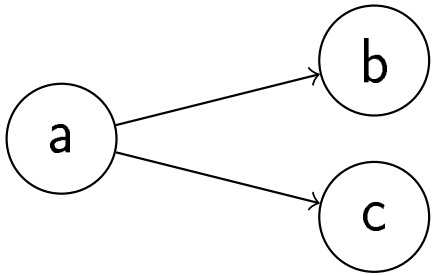
Fan-in/Fan-out [Dick *et al.*, 1998]



Parameters

n : Lower limit of the number of nodes.
od : Maximal out-degree allowed on each node.
id : Maximal in-degree allowed on each node.

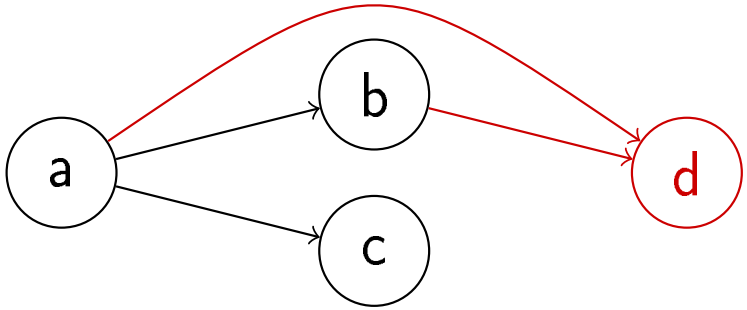
Fan-in/Fan-out [Dick *et al.*, 1998]



Parameters

n : Lower limit of the number of nodes.
od : Maximal out-degree allowed on each node.
id : Maximal in-degree allowed on each node.

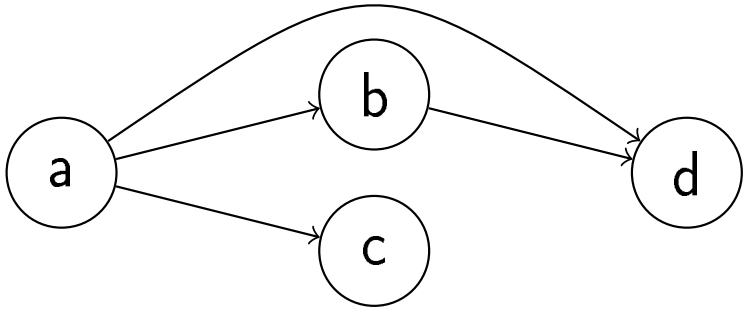
Fan-in/Fan-out [Dick *et al.*, 1998]



Parameters

n : Lower limit of the number of nodes.
od : Maximal out-degree allowed on each node.
id : Maximal in-degree allowed on each node.

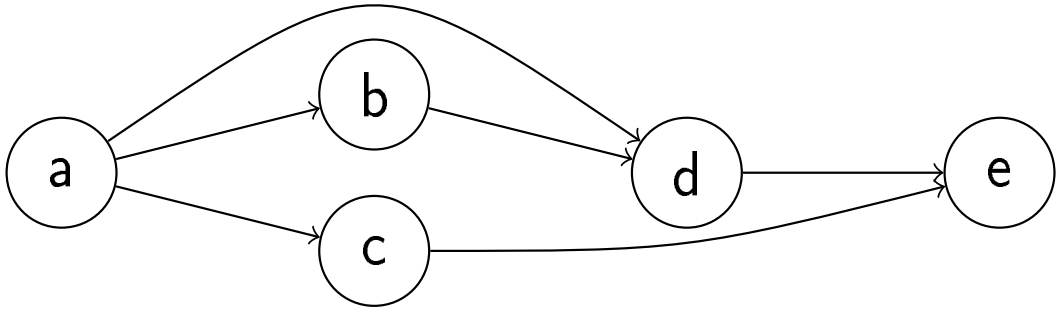
Fan-in/Fan-out [Dick *et al.*, 1998]



Parameters

n : Lower limit of the number of nodes.
od : Maximal out-degree allowed on each node.
id : Maximal in-degree allowed on each node.

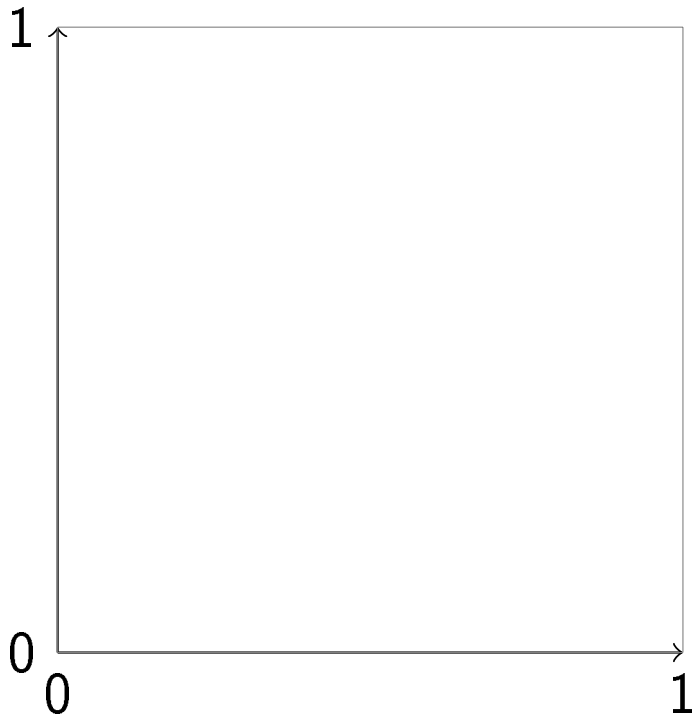
Fan-in/Fan-out [Dick *et al.*, 1998]



Parameters

n : Lower limit of the number of nodes.
od : Maximal out-degree allowed on each node.
id : Maximal in-degree allowed on each node.

Random Orders [Winkler, 1985]

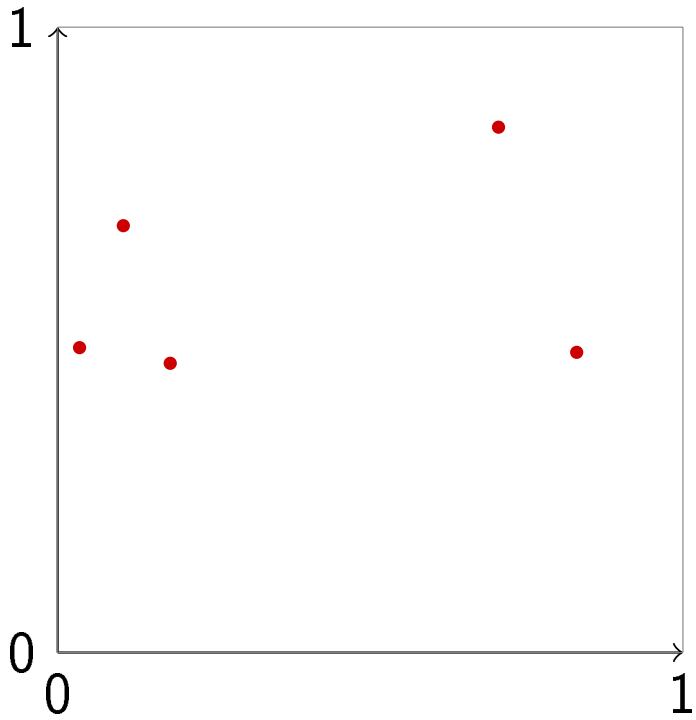


Parameters

n : Number of nodes.

k : Number of total orders to intersect.

Random Orders [Winkler, 1985]

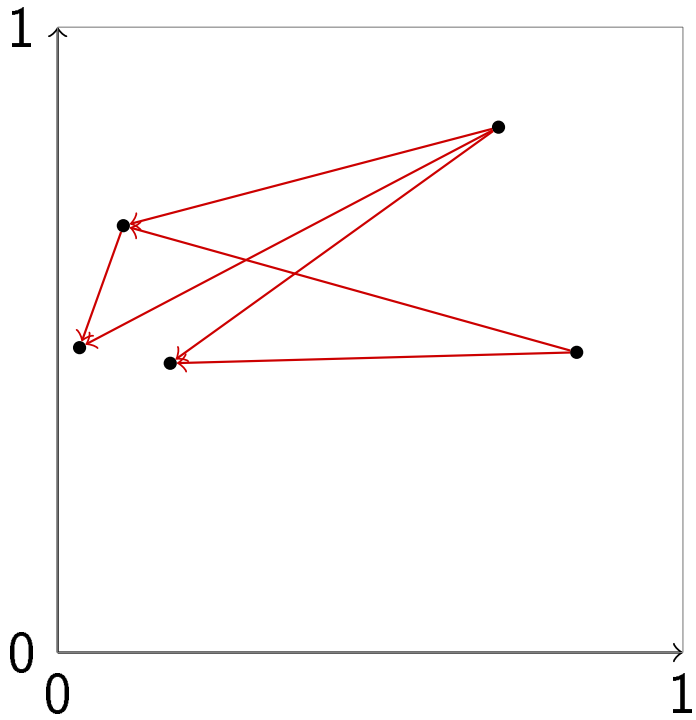


Parameters

n : Number of nodes.

k : Number of total orders to intersect.

Random Orders [Winkler, 1985]

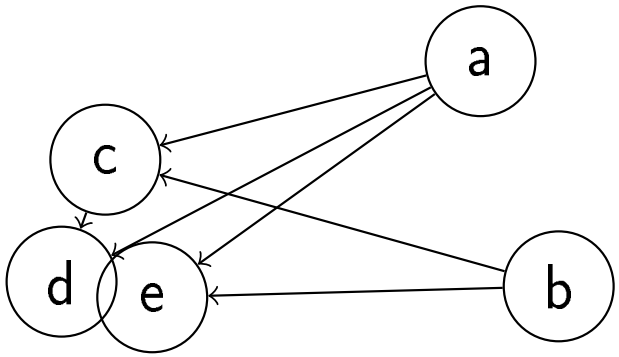


Parameters

n : Number of nodes.

k : Number of total orders to intersect.

Random Orders [Winkler, 1985]

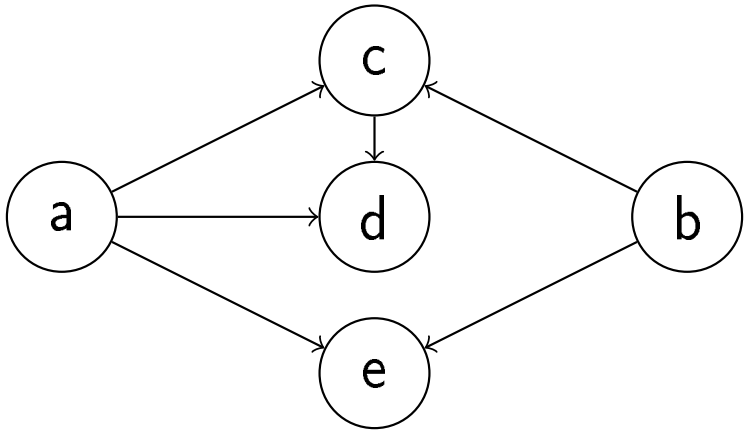


Parameters

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

Many generation methods and variants.

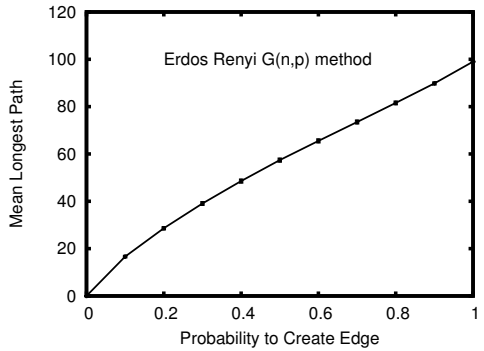
Difficulties for Performance Evaluation

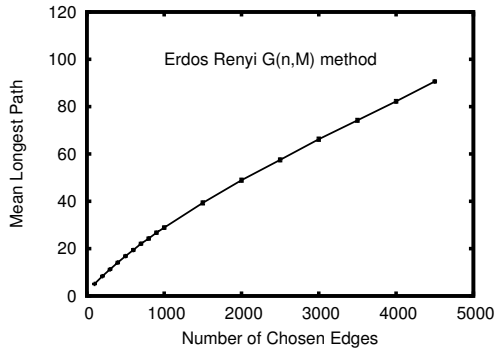
Input characteristics are a must.

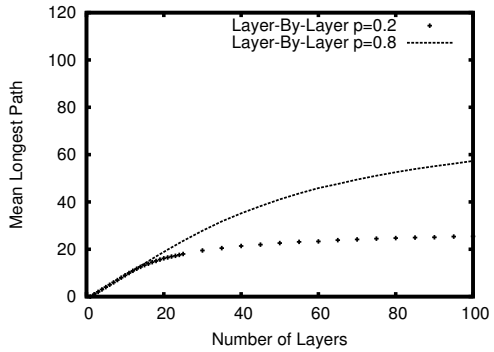
Problems

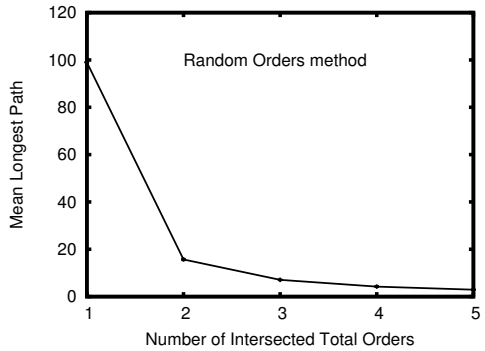
- Analytical results are very hard to obtain.
- Implementation details impact quality of the generators.

Critical Path Analysis









Experimental Design	
Sample Size:	1,000
Number of nodes:	100
Confidence Intervals:	95%

Outline

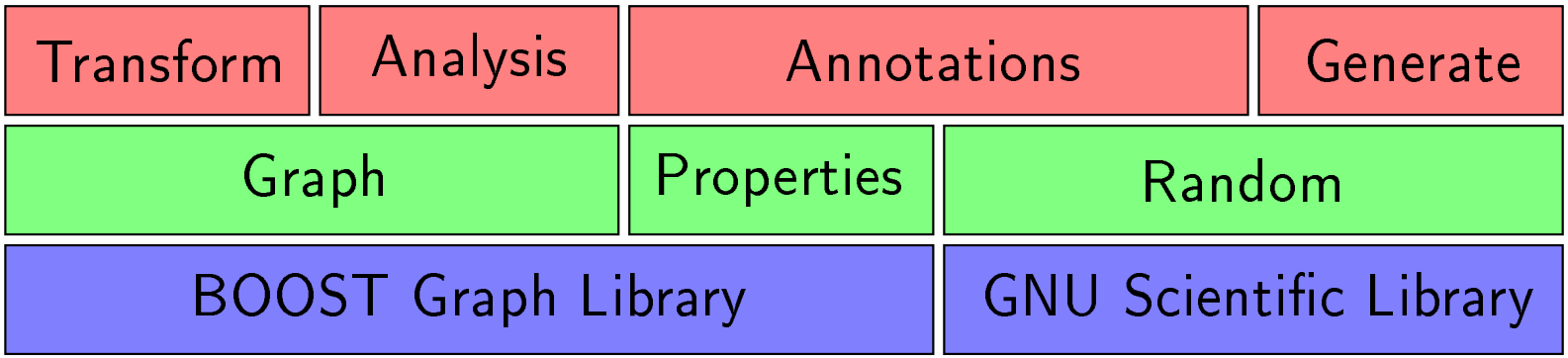
Classical Performance Evaluation Process

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.

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



GGen: Technical Info

Random Graph Generator

Contains all previously shown methods.
Easily extensible code.
Standard output format: Graphviz DOT.

Technical Info

C/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.

GGen: Example Code

```
#include <boost/config.hpp>
#include <boost/graphviz.hpp>
#include <ggen/random.hpp>
#include <ggen/graph-generation.hpp>

int main(void) {
        Graph *g = NULL;
        generation_context *cntxt= NULL;
        ggen::ggen_rng *rng = NULL;

        ggen::vertices_size n = 100;
        double p = 0.3;

        rng = new ggen::ggen_rng();
        rng->allocate(GGEN_RNG_DEFAULT);
        rng->seed(time(NULL));
        cntxt->set_rng(rng);

        g = ggen::generation::erdos_gnp(cntxt,n,p,true);

        write_graphviz(std::cout,*g);

        delete g;
        delete rng;
        delete cntxt;
}
```

Outline

Comparison of List Schedulers

1. Build a priority list of all tasks.
2. At each step of the scheduling:
   1. Find an available resource.
   2. Assign the highest priority task to it.

List Scheduling algorithms differ from each other on the strategy used to build their priority list.

Strategies simulated

BottomLevel	longest path to sink
Sons	out-degree
Reverse	– out-degree
Random	uniform random choice

Experimental Design	
Sample Size:	1,000
Number of nodes:	100
Task Exec. Time:	1
Number of Proc.:	Varying

More than 1,500,000 simulations for this experiment.

Makespan Analysis

Average Completion Time on 4 Processors

	Sons		BottomLevel		Reverse		Random	
	avg.	sd.	avg.	sd.	avg.	sd.	avg.	sd.
GNP(100,0.25)	36.1	2.9	35.2	3.1	36.9	2.8	36.5	2.9
GNM(100,300)	25.1	0.4	25.0	0.0	27.2	0.9	26.3	0.8
FiFo(100,10,10)	27.9	1.4	27.9	1.4	29.4	1.6	28.6	1.5
Layer(100,10,0.5)	25.9	0.5	25.8	0.4	27.3	0.7	26.3	0.6
RandomOrders(100,2)	25.4	0.5	25.4	0.5	29.1	1.2	27.1	0.9
GNP(100,0.25)	35.0	3.2	35.0	3.2	35.0	3.2	35.0	3.2
GNM(100,300)	12.4	1.7	12.3	1.7	12.5	1.7	12.4	1.7
FiFo(100,10,10)	11.8	2.3	11.8	2.3	13.3	2.1	12.6	2.3
Layer(100,10,0.5)	10.2	0.4	10.2	0.4	10.2	0.4	10.2	0.4
RandomOrders(100,2)	16.7	1.7	16.7	1.7	16.7	1.7	16.7	1.7

Average Completion Time on 16 Processors

Makespan Analysis

Average Completion Time on 4 Processors

	Sons		BottomLevel		Reverse		Random	
	avg.	sd.	avg.	sd.	avg.	sd.	avg.	sd.
GNP(100,0.25)	36.1	2.9	35.2	3.1	36.9	2.8	36.5	2.9
GNM(100,300)	25.1	0.4	25.0	0.0	27.2	0.9	26.3	0.8
FiFo(100,10,10)	27.9	1.4	27.9	1.4	29.4	1.6	28.6	1.5
Layer(100,10,0.5)	25.9	0.5	25.8	0.4	27.3	0.7	26.3	0.6
RandomOrders(100,2)	25.4	0.5	25.4	0.5	29.1	1.2	27.1	0.9
GNP(100,0.25)	35.0	3.2	35.0	3.2	35.0	3.2	35.0	3.2
GNM(100,300)	12.4	1.7	12.3	1.7	12.5	1.7	12.4	1.7
FiFo(100,10,10)	11.8	2.3	11.8	2.3	13.3	2.1	12.6	2.3
Layer(100,10,0.5)	10.2	0.4	10.2	0.4	10.2	0.4	10.2	0.4
RandomOrders(100,2)	16.7	1.7	16.7	1.7	16.7	1.7	16.7	1.7

Average Completion Time on 16 Processors

Makespan Analysis

Average Completion Time on 4 Processors

	Sons		BottomLevel		Reverse		Random	
	avg.	sd.	avg.	sd.	avg.	sd.	avg.	sd.
GNP(100,0.25)	36.1	2.9	35.2	3.1	36.9	2.8	36.5	2.9
GNM(100,300)	25.1	0.4	25.0	0.0	27.2	0.9	26.3	0.8
FiFo(100,10,10)	27.9	1.4	27.9	1.4	29.4	1.6	28.6	1.5
Layer(100,10,0.5)	25.9	0.5	25.8	0.4	27.3	0.7	26.3	0.6
RandomOrders(100,2)	25.4	0.5	25.4	0.5	29.1	1.2	27.1	0.9
GNP(100,0.25)	35.0	3.2	35.0	3.2	35.0	3.2	35.0	3.2
GNM(100,300)	12.4	1.7	12.3	1.7	12.5	1.7	12.4	1.7
FiFo(100,10,10)	11.8	2.3	11.8	2.3	13.3	2.1	12.6	2.3
Layer(100,10,0.5)	10.2	0.4	10.2	0.4	10.2	0.4	10.2	0.4
RandomOrders(100,2)	16.7	1.7	16.7	1.7	16.7	1.7	16.7	1.7

Average Completion Time on 16 Processors

Outline

GGen: An Open Framework for DAG Generation

DAG Generator

Easily extensible framework.
Inclusion of most classical methods.

Toolbox for DAG Analysis

Set of classical dag algorithms.
Easy to interface : DOT

Statistical Studies

Massive simulations and analysis campaigns.

Ongoing and Future Works

On Graph Generation

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

On Simulations

- *Harder* scheduling problems.
- Different domains (graph drawing, compiling,...)

Open Problems

- Find $m1$ and $m2$ such that $s1(m1) < s2(m1)$ and $s1(m2) > s2(m2)$.
- Mapping workload traces to generation methods.

Thanks

Thank you for your attention.

Demo available on demand.

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

Bibliography


R. P. Dick, D. L. Rhodes, and W. Wolf.
TGFF: Task Graphs For Free.
In *Proceedings of the 6th International Workshop on Hardware/Software Codesign*, pages 97–101, Washington, DC, USA, Mar. 1998. IEEE Computer Society.


P. Erdős and A. Rényi.
On random graphs I.
Publicationes Mathematicae Debrecen, 6:290–297, 1959.


T. Tobita and H. Kasahara.
A standard task graph set for fair evaluation of multiprocessor scheduling algorithms.
Journal of Scheduling, 5(5):379–394, 2002.


P. Winkler.
Random orders.
Order, 1(4):317–331, Dec. 1985.