AREA-Oriented DAG-Scheduling: A Preliminary Assessment

Arnold L. Rosenberg
Northeastern University
⟨ rsnbrg@ccs.neu.edu ⟩ or ⟨ rsnbrg@cs.umass.edu ⟩

Collaborators	:
Gennaro Cordasco	Univ. Naples 2
Rosario De Chiara	Poste Italiane Res. Ctr.
Trilce Estrada	Univ. New Mexico
Rajmohan Rajaraman	Northeastern Univ.
Scott T. Roche	Northeastern Univ.
Michela Taufer	Univ. Delaware

An Emerging Challenge for HPC: ***Dynamically Heterogeneous Computing Platforms***

The constituent *"workers"* in many modern computing platforms

— e.g., *clouds*, or *grids*, *desktop grids*, *volunteer computing projects*

provide computing power at rates that cannot be known reliably $a\ priori$.

An Emerging Challenge for HPC: ***Dynamically Heterogeneous Computing Platforms***

The constituent *"workers"* in many modern computing platforms provide computing power at rates that cannot be known reliably *a priori*.

In fact, the workers' computing power can change

— *at unexpected times*

An Emerging Challenge for HPC: *Dynamically Heterogeneous Computing Platforms*

The constituent *"workers"* in many modern computing platforms provide computing power at rates that cannot be known reliably *a priori*.

In fact, the workers' computing power can change

— *at unexpected times*

— *in unexpected ways*

An Emerging Challenge for HPC: *Dynamically Heterogeneous Computing Platforms*

The constituent *"workers"* in many modern computing platforms provide computing power at rates that cannot be known reliably *a priori*.

In fact, the workers' computing power can change

— *at unexpected times*

— *in unexpected ways*

The platforms exhibit *DYNAMIC HETEROGENEITY*

Is HPC achievable for Dynamically Heterogeneous platforms?

When jobs have *interchore dependencies* (modeled as DAGs)—

how can we cope with the *temporal unpredictability* of workers?

Is HPC achievable for Dynamically Heterogeneous platforms?

"When jobs have *interCHORE dependencies*"

We use the *granularity-neutral* term "chore" to capture "jobs," "tasks," etc., of arbitrary granularities.

Is HPC achievable for Dynamically Heterogeneous platforms?

When jobs have *interchore dependencies* (modeled as DAGs)—

how can we cope with the *temporal unpredictability* of workers?

Dynamic heterogeneity makes critical-path analysis impossible

Is HPC achievable for Dynamically Heterogeneous platforms?

When jobs have *interchore dependencies* (modeled as DAGs)—

how can we cope with the *temporal unpredictability* of workers?

Dynamic heterogeneity makes critical-path analysis impossible

One possible goal for coping is to

maximize the number of chores that are eligible for allocation at every step of the computation

Is HPC achievable for Dynamically Heterogeneous platforms?

When jobs have *interchore dependencies* (modeled as DAGs)—

how can we cope with the *temporal unpredictability* of workers?

Dynamic heterogeneity makes critical-path analysis impossible

One possible goal for coping is to

maximize the number of chores that are eligible for allocation at every step of the computation

THIS IS NOT ALWAYS ACHIEVABLE!

Is HPC achievable for Dynamically Heterogeneous platforms?

When jobs have *interchore dependencies* (modeled as DAGs)—

how can we cope with the *temporal unpredictability* of workers?

Dynamic heterogeneity makes critical-path analysis impossible

One possible goal for coping is to

maximize the number of chores that are eligible for allocation at every step of the computation

THIS IS NOT ALWAYS ACHIEVABLE!

Many DAGs do not admit schedules that *always* maximize the number of eligible chores.

Is HPC achievable for Dynamically Heterogeneous platforms?

When jobs have *interchore dependencies* (modeled as DAGs)—

how can we cope with the *temporal unpredictability* of workers?

Dynamic heterogeneity makes critical-path analysis impossible

One possible goal for coping is to

maximize the number of chores that are eligible for allocation at every step of the computation

This is not always achievable!

Many DAGs do not admit schedules that *always* maximize the number of eligible chores.

BUT WE CAN ALWAYS

MAXIMIZE THE AVERAGE NUMBER OF ELIGIBLE CHORES.

DAG-Schedules and Their AREAs

A (computation-) DAG $\mathcal{G}$ represents a computational job.

DAG-Schedules and Their AREAs

A (computation-) DAG $\mathcal{G}$ represents a computational job.

- Each *node* of $\mathcal{G}$ is a *chore*

DAG-Schedules and Their AREAs

A (computation-) DAG $\mathcal{G}$ represents a computational job.

- Each *node* of $\mathcal{G}$ is a *chore*
- Each *arc* of $\mathcal{G}$ is an *interchore dependency*

DAG-Schedules and Their AREAs

A (computation-) DAG $\mathcal{G}$ represents a computational job.

- Each $node$ of $\mathcal{G}$ is a $chore$
- Each arc of $\mathcal{G}$ is an $interchore\ dependency$

  Arc $(u \rightarrow v)$ means that chore v cannot be executed before chore u

  Chore u is a ***parent*** of chore v in $\mathcal{G}$

DAG-Schedules and Their AREAs

A (computation-) DAG $\mathcal{G}$ represents a computational job.

- Each $node$ of $\mathcal{G}$ is a $chore$
- Each arc of $\mathcal{G}$ is an $interchore\ dependency$

A *schedule* Σ for $\mathcal{G}$ is a rule for selecting the next *eligible* chore to execute

(We measure time in an event-driven manner)

DAG-Schedules and Their AREAs

A (computation-) DAG $\mathcal{G}$ represents a computational job.

- Each $node$ of $\mathcal{G}$ is a $chore$
- Each arc of $\mathcal{G}$ is an $interchore\ dependency$

A *schedule* Σ for $\mathcal{G}$ is a rule for selecting the next *eligible* chore to execute

Thus, Σ is a *topological sort* of $\mathcal{G}$.

DAG-Schedules and Their AREAs

Chore v of DAG $\mathcal{G}$ is *eligible (for execution) at step* t

if all of v's parents have been executed by step t

DAG-Schedules and Their AREAs

Chore v of DAG $\mathcal{G}$ is *eligible (for execution) at step* t

if all of v's parents have been executed by step t

$E_\Sigma(t) \stackrel{\text{def}}{=}$ the number of chores that are eligible at step t of Σ's execution of $\mathcal{G}$

DAG-Schedules and Their AREAs

Chore v of DAG $\mathcal{G}$ is *eligible (for execution) at step* t

if all of v's parents have been executed by step t

$E_\Sigma(t) \stackrel{\text{def}}{=}$ the number of chores that are eligible at step t of Σ's execution of $\mathcal{G}$

$$AREA(\Sigma) \stackrel{\text{def}}{=} E_\Sigma(0) + E_\Sigma(1) + \cdots + E_\Sigma(N_{\mathcal{G}})$$

—the unnormalized average number of eligible chores during Σ's execution of $\mathcal{G}$

DAG-Schedules and Their AREAs

Chore v of DAG $\mathcal{G}$ is *eligible (for execution) at step* t

if all of v's parents have been executed by step t

$E_\Sigma(t) \stackrel{\mathrm{def}}{=}$ the number of chores that are eligible at step t of Σ's execution of $\mathcal{G}$

$$AREA(\Sigma) \ \stackrel{\mathrm{def}}{=} \ E_\Sigma(0) + E_\Sigma(1) + \cdots + E_\Sigma(N_\mathcal{G})$$

—the unnormalized average number of eligible chores during Σ's execution of $\mathcal{G}$

(“AREA” invokes the analogy with Riemann sums as approximations of integrals)

DAG-Schedules and Their AREAs

Chore v of DAG $\mathcal{G}$ is *eligible (for execution) at step* t

if all of v's parents have been executed by step t

$E_\Sigma(t) \stackrel{\text{def}}{=}$ the number of chores that are eligible at step t of Σ's execution of $\mathcal{G}$

$$AREA(\Sigma) \ \stackrel{\text{def}}{=}\ E_\Sigma(0) + E_\Sigma(1) + \cdots + E_\Sigma(N_\mathcal{G})$$

$$\approx\approx\approx\approx\approx\approx\approx\approx\approx$$

The (general version of the) AREA-MAX problem

GIVEN DAG $\mathcal{G}$, FIND A SCHEDULE Σ WITH MAXIMAL $AREA(\Sigma)$

is $\mathbf{NP}$-complete

DAG-Schedules and Their AREAs

Chore v of DAG $\mathcal{G}$ is *eligible (for execution) at step* t

if all of v's parents have been executed by step t

$E_\Sigma(t) \stackrel{\text{def}}{=}$ the number of chores that are eligible at step t of Σ's execution of $\mathcal{G}$

$$AREA(\Sigma) \ \stackrel{\text{def}}{=} \ E_\Sigma(0) + E_\Sigma(1) + \cdots + E_\Sigma(N_\mathcal{G})$$

≈≈≈≈≈≈≈≈≈

The (general version of the) AREA-MAX problem

GIVEN DAG $\mathcal{G}$, FIND A SCHEDULE Σ WITH MAXIMAL $AREA(\Sigma)$

is $\mathbf{NP}$*-complete*

—via reduction from Minimum Weighted Completion Time

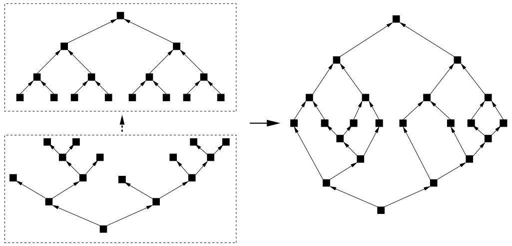
$\boxed{\textbf{Responding to the NP-completeness of AREA-MAX}}$

1. $\boxed{\textit{Efficient AREA-maximizing schedules for many DAG-families}}$

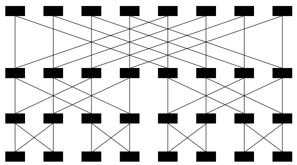
2. Efficient heuristics that “seem” to work well

Efficient AREA-maximizing schedules for many DAG-families

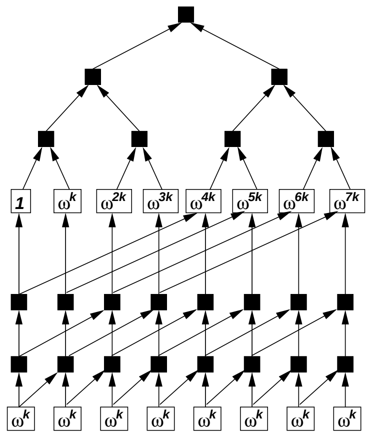
Expansive-Reductive (ER) DAGs



Convolutional DAGs



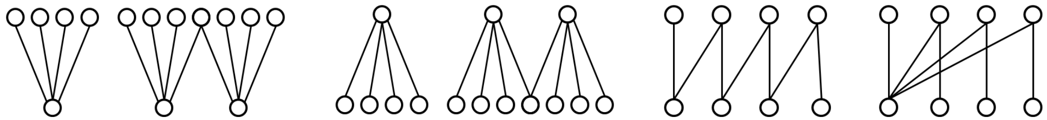
Compositions of ER, Convolutional DAGs



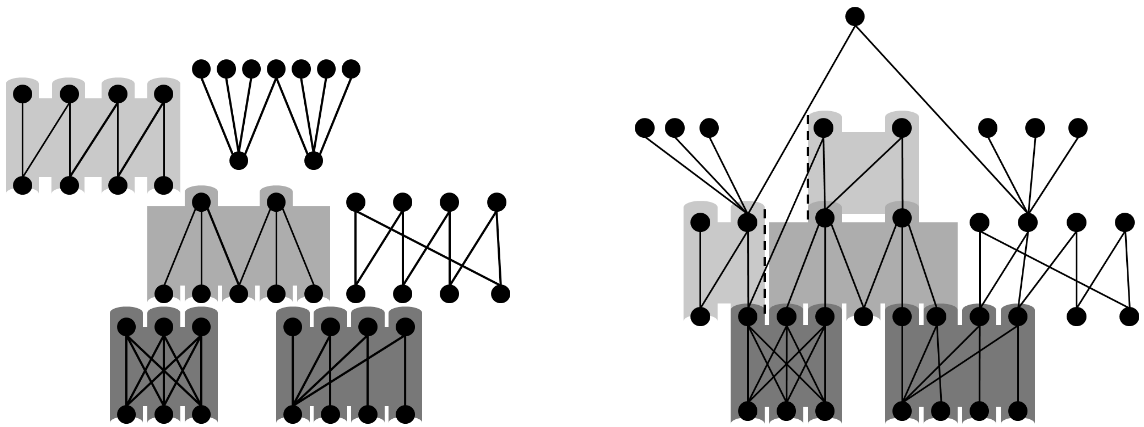
Efficient AREA-maximizing schedules for many DAG-families

LEGO-DAGs — a family of families of significant DAGs

Some bipartite building-block DAGs. (All arcs point upward.)



Composing building blocks (left) into a *LEGO*-DAG (right)

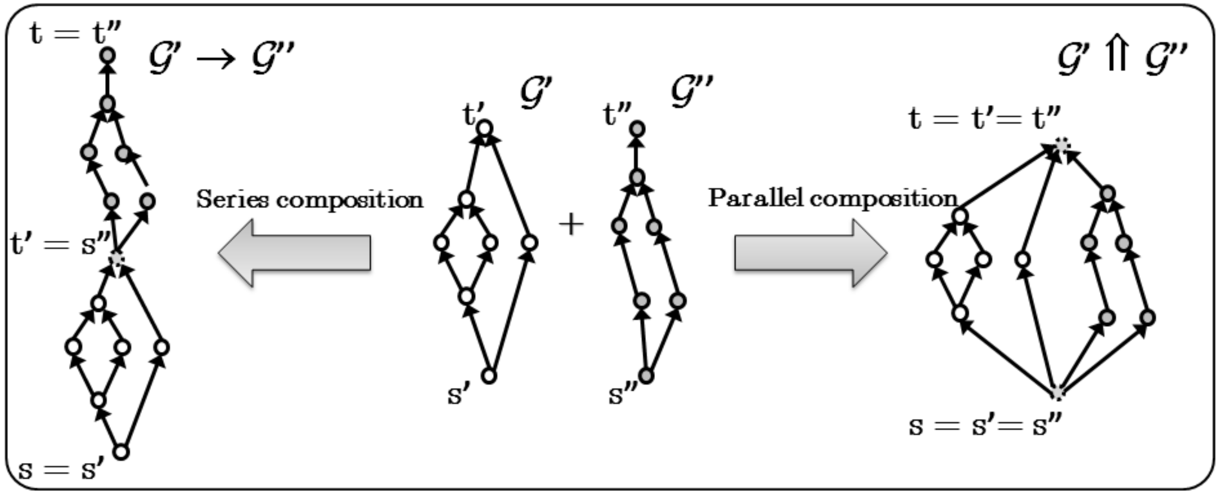


Efficient AREA-maximizing schedules for many DAG-families

A *REALLY* important special family: *Series-Parallel DAGs* (*SP-DAGs*)

- model multi-threaded computations
- lead to a good DAG-scheduling heuristic

The defining compositions that produce *SP-DAGs*

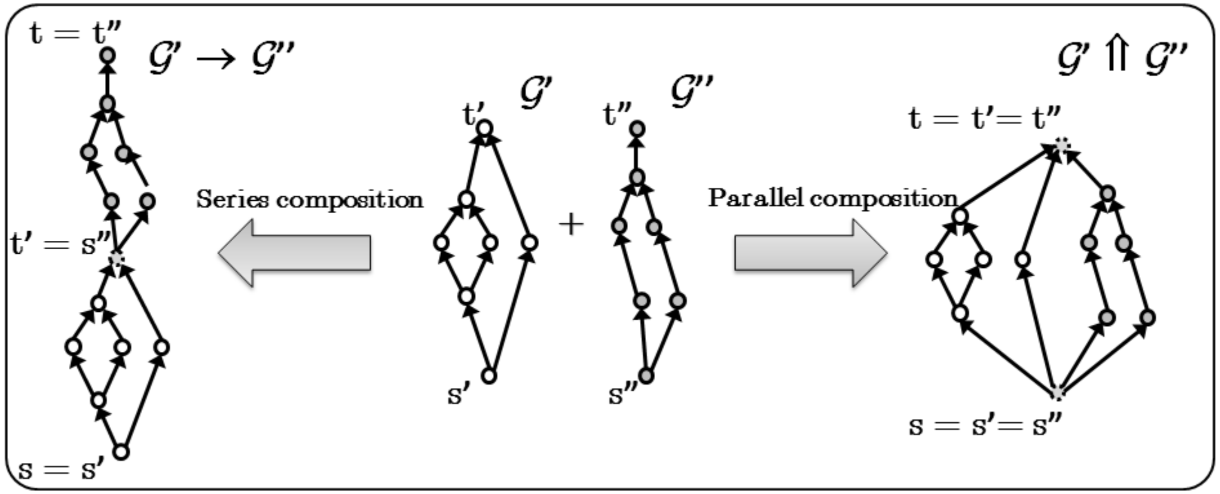


Efficient AREA-maximizing schedules for many DAG-families

A really important special family: *Series-Parallel DAGs* (*SP-DAGs*)

- model multi-threaded computations
- lead to a good DAG-scheduling heuristic

The defining compositions that produce *SP-DAGs*



One can find an AREA-maximizing schedule for any SP-DAG in quadratic time

Responding to the NP-completeness of AREA-MAX

1. Efficient AREA-maximizing schedules for many DAG-families

2. *Efficient heuristics that "seem" to work well*

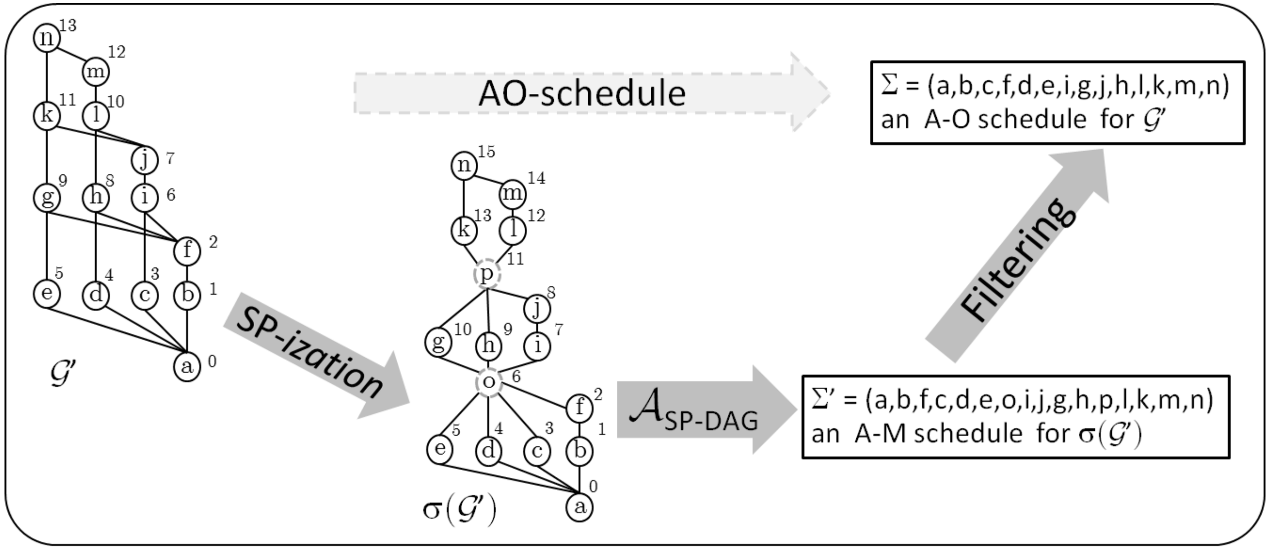
Converting an arbitrary DAG to a SP-DAG

In quadratic time, one can convert any DAG $\mathcal{G}$

to an SP-DAG $\sigma(\mathcal{G})$ (via an "SP-ization")

that has "roughly as much" parallelism as $\mathcal{G}$

A sample SP-ization. (Note "additional" [synchronizing] chores)



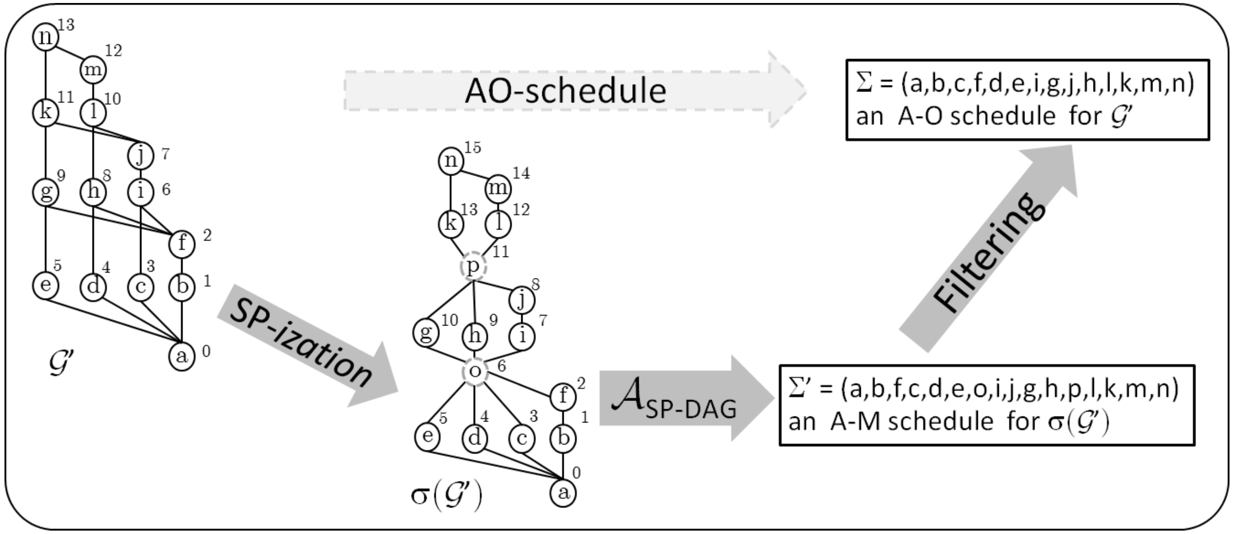
The efficient AOSPD heuristic —that "seems" to work well

The **AOSPD** DAG-scheduling heuristic

$\mathcal{G}$ is the DAG that you want to schedule with large AREA

If $\mathcal{G}$ is an SP-DAG
then use our AREA-maximizing SP-DAG scheduler
else

1. Transform $\mathcal{G}$ to the SP-DAG $\sigma(\mathcal{G})$ using a prescribed SP-izer
2. Use our AREA-maximizing SP-DAG scheduler to schedule $\sigma(\mathcal{G})$
3. "Filter" the resulting schedule to eliminate the "additional" chores



$\boxed{\textbf{EXPERIMENTAL SECTION}}$

Assessing the A-O Paradigm's Impact — via the AOSPD heuristic

Our assessment performs the following tests:

- AOSPD's schedules' MAKESPANs vs. other "oblivous" schedulers'
- AOSPD's schedules' AREAs vs. other "oblivous" schedulers'
- AOSPD's schedules' AREAs vs. true AREA-Maximizing schedules
  —for DAGs whose A-M schedules we know how to generate efficiently
- Test the hypothesis that larger AREA means smaller MAKESPAN
  —Is there a positive correlation?

The Competing “Oblivious” Schedulers

- The **FIFO** scheduler:
  Stores newly eligible chores in a fifo queue
  —very lightweight, not very effective

- The **Static-Greedy** scheduler:
  Stores newly eligible chores in a MAX-priority queue
  ordered by *outdegree*
  —much less lightweight, more effective

- The **Dynamic-Greedy** scheduler:
  Stores newly eligible chores in a MAX-priority queue
  ordered by *yield* (number of chores they would render eligible)
  —rather heavyweight, quite effective (*one-step optimal*)

The Experimental Protocol

- **The MAKESPAN experiment**

  Compare AOSPD's **performance to** heuristic H's **via the ratio** $T(H) \div T(AO)$

  – Chore execution-times are distributed normally (positive half):
  mean $= 1$;   std-dev $\in \{0.1, 0.5\}$

  – The number of available workers at step (c_t) is distributed exponentially with rate parameter λ:
  $P[c_t = x] = \lambda e^{-\lambda x}$; $\lambda \in \{2^{-k} | k \in [0..7]\}$

- **The AREA experiment**

  Compare AOSPD's **performance to** heuristic H's **via the ratio** $AREA(AO) \div AREA(H)$

The Experimental Protocol

- **The MAKESPAN experiment**

  Compare AOSPD's **performance to** heuristic H's **via the ratio** $T(H) \div T(AO)$

  - Chore execution-times are distributed normally (positive half):
    mean $= 1$; std-dev $\in \{0.1, 0.5\}$
  - The number of available workers at step (c_t) is distributed exponentially with rate parameter λ:
    $P[c_t = x] = \lambda e^{-\lambda x}$; $\lambda \in \{2^{-k} | k \in [0..7]\}$

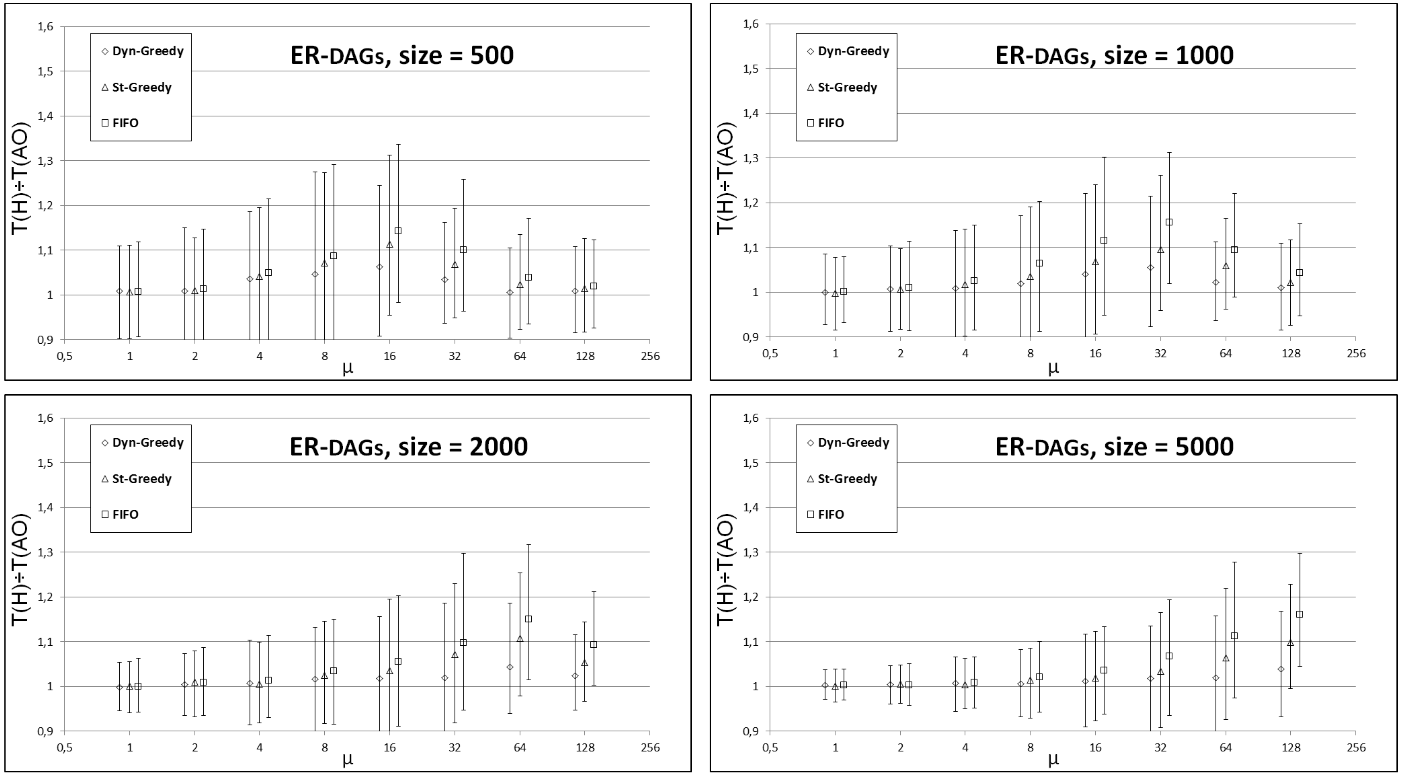
- **The AREA experiment**

  Compare AOSPD's **performance to** heuristic H's **via the ratio** $AREA(AO) \div AREA(H)$

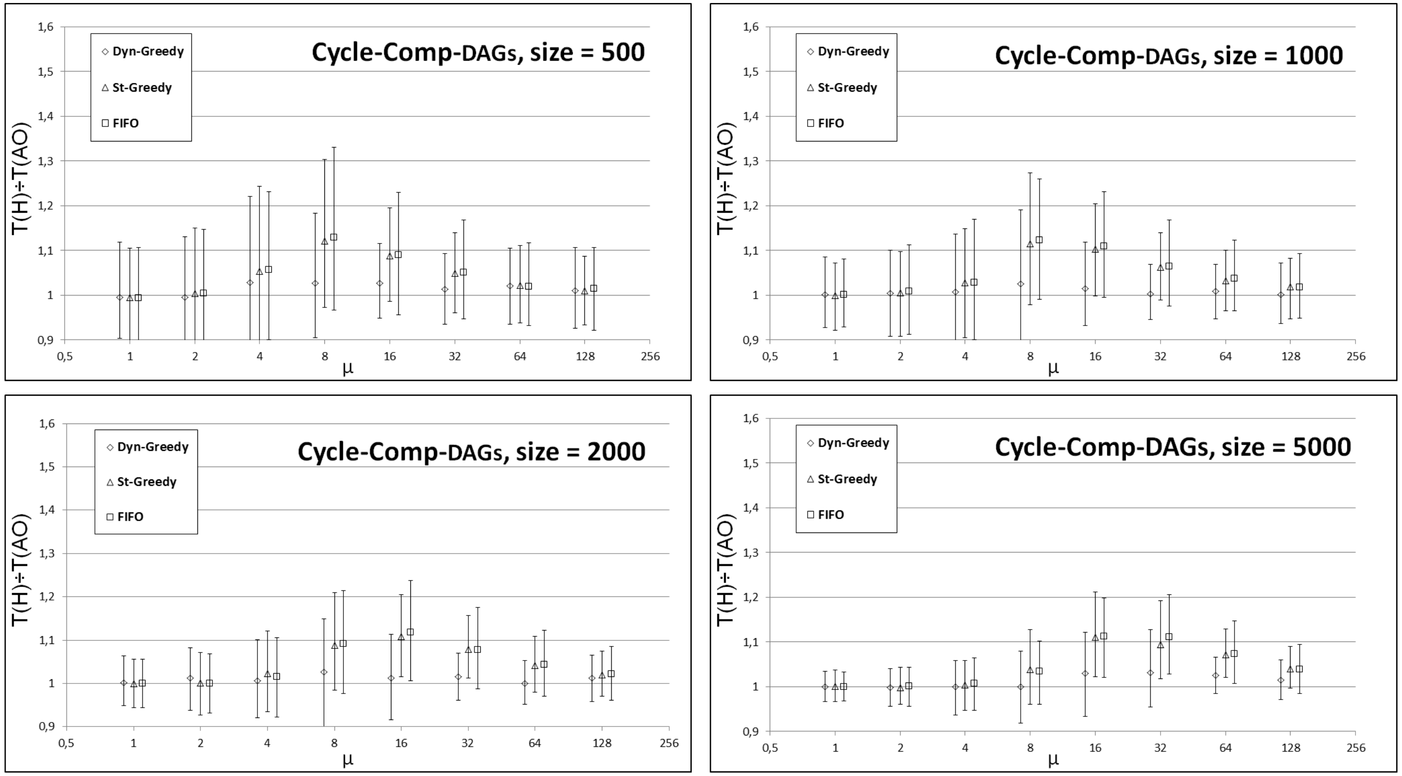
RECALL THAT $\begin{cases} \text{SMALLER MAKESPAN IS BETTER} \\ \text{BIGGER AREA IS BETTER} \end{cases}$

\# EXPERIMENTAL RESULTS

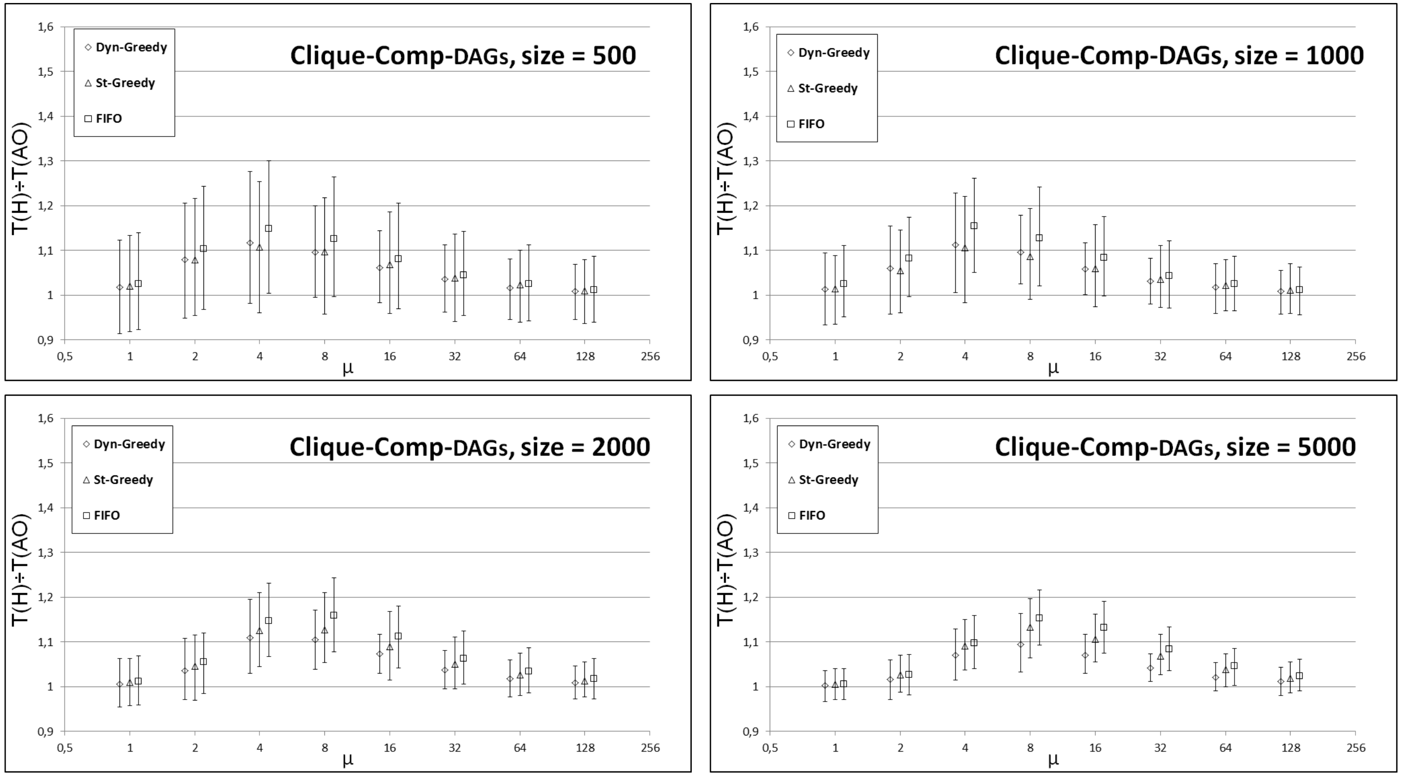
AO's schedules' MAKESPANs vs. Competitors': Expansive-Reductive (ER) DAGs



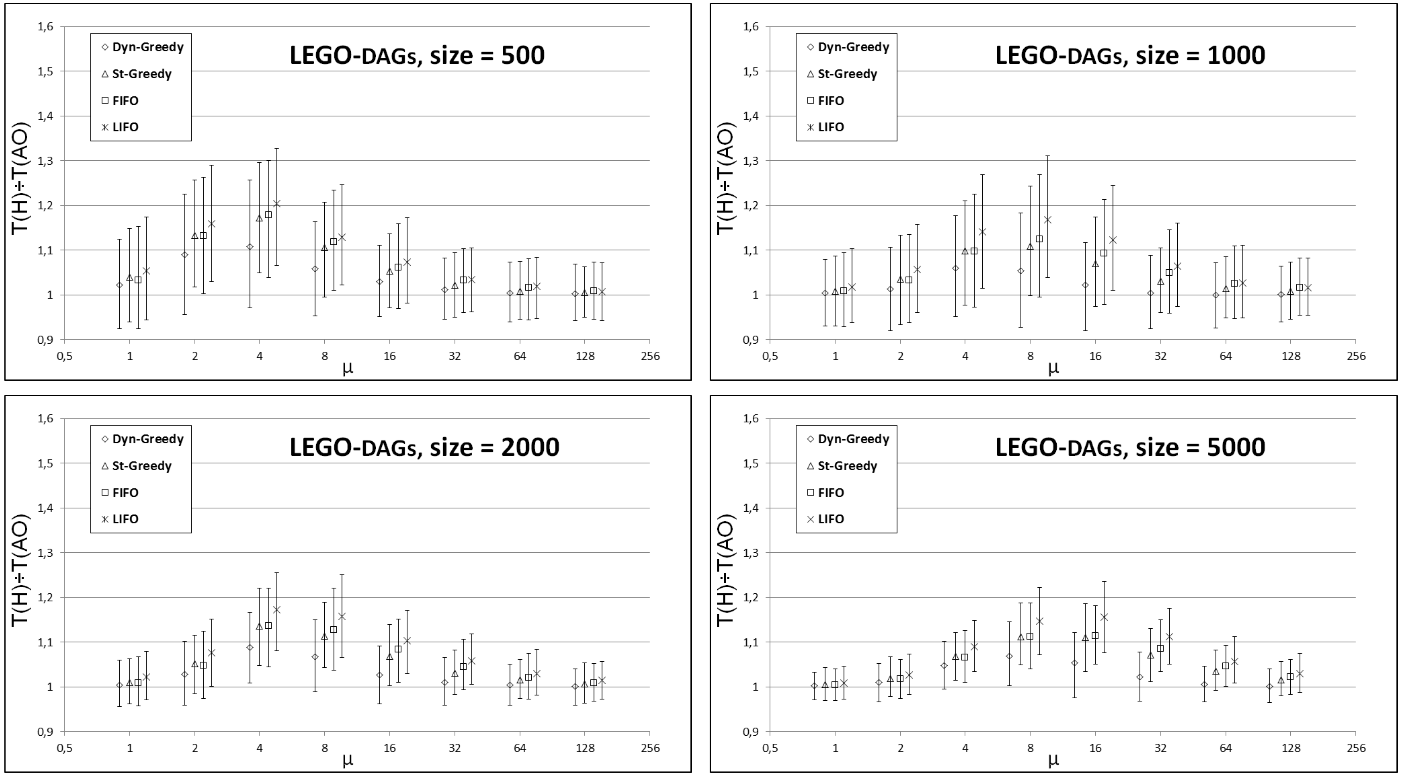
AO's schedules' MAKESPANs vs. Competitors': Cycle-composition DAGs



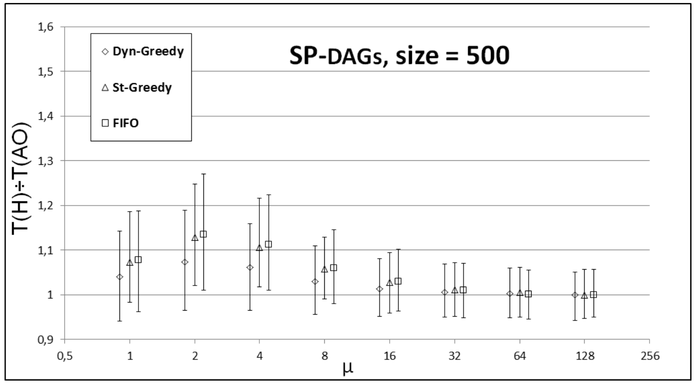
AO's schedules' MAKESPANs vs. Competitors': Clique-composition DAGs

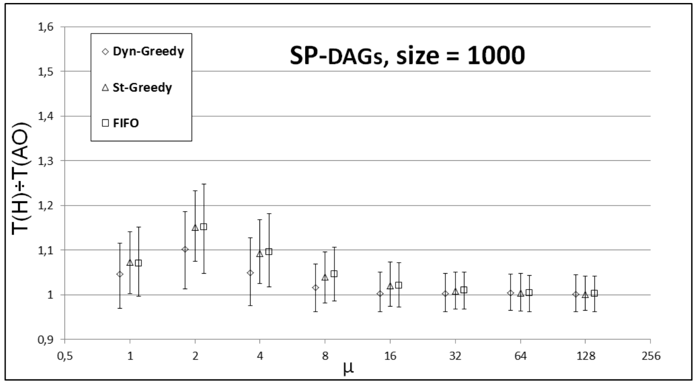


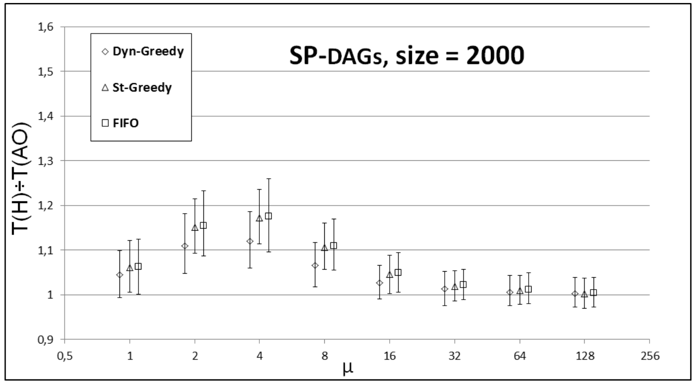
AO's schedules' MAKESPANs vs. Competitors': <u>LEGO-DAGs</u>

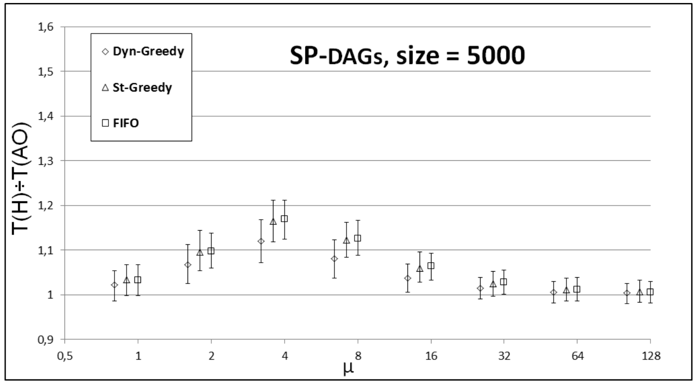


AO's schedules' MAKESPANs vs. Competitors': <u>Series-Parallel DAGs</u>



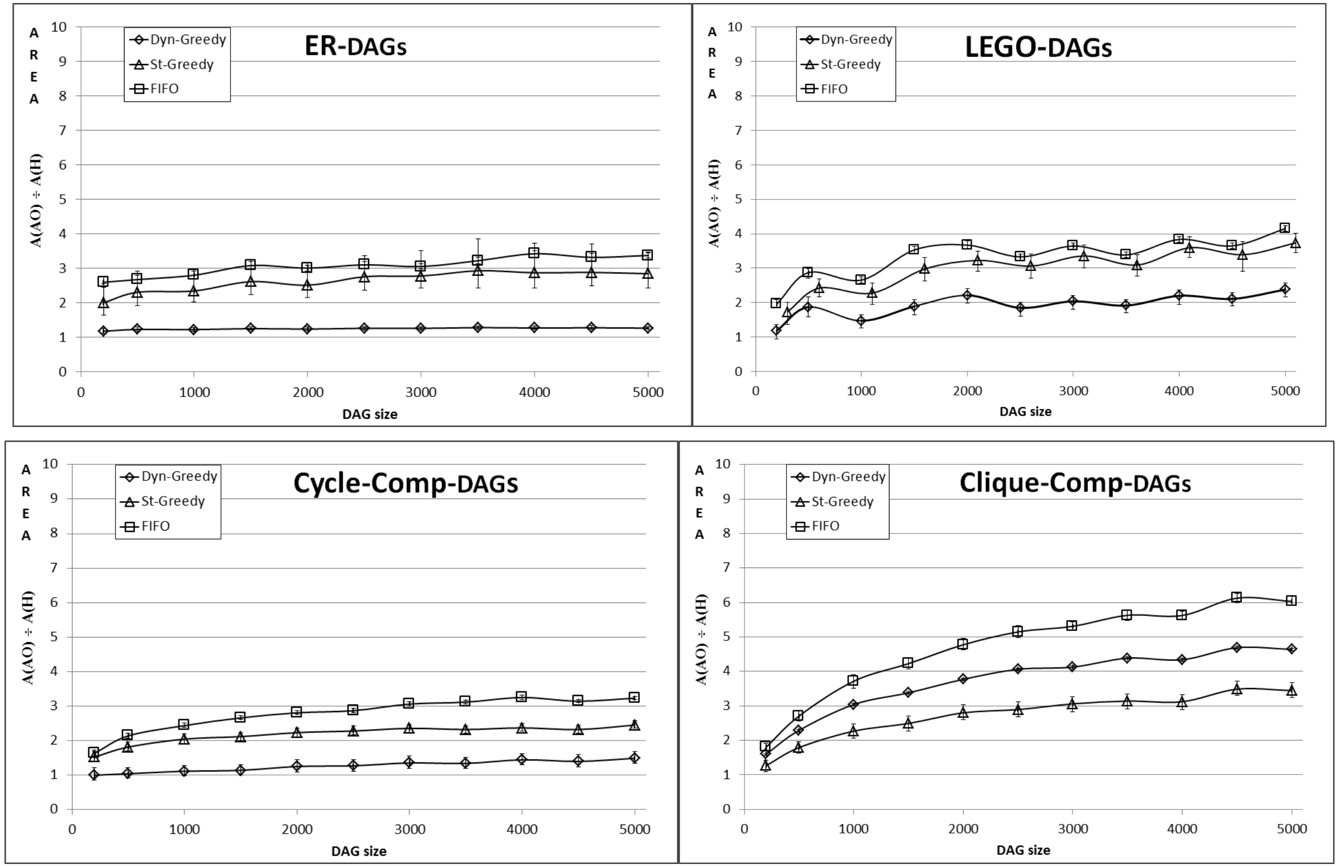






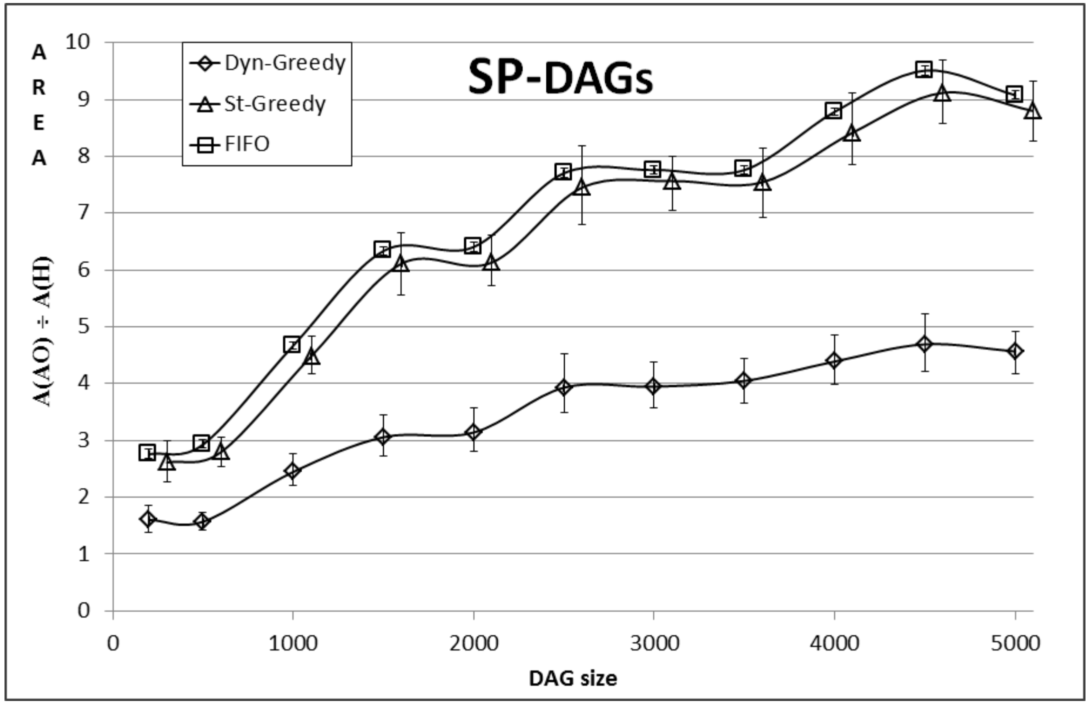
AO's schedules' AREAs vs. Competitors': ER-DAGs, Cycle-composition DAGs, Clique-composition DAGs, LEGO-DAGs

Mean ratios and ranges: $AREA(AO) \div AREA(H)$



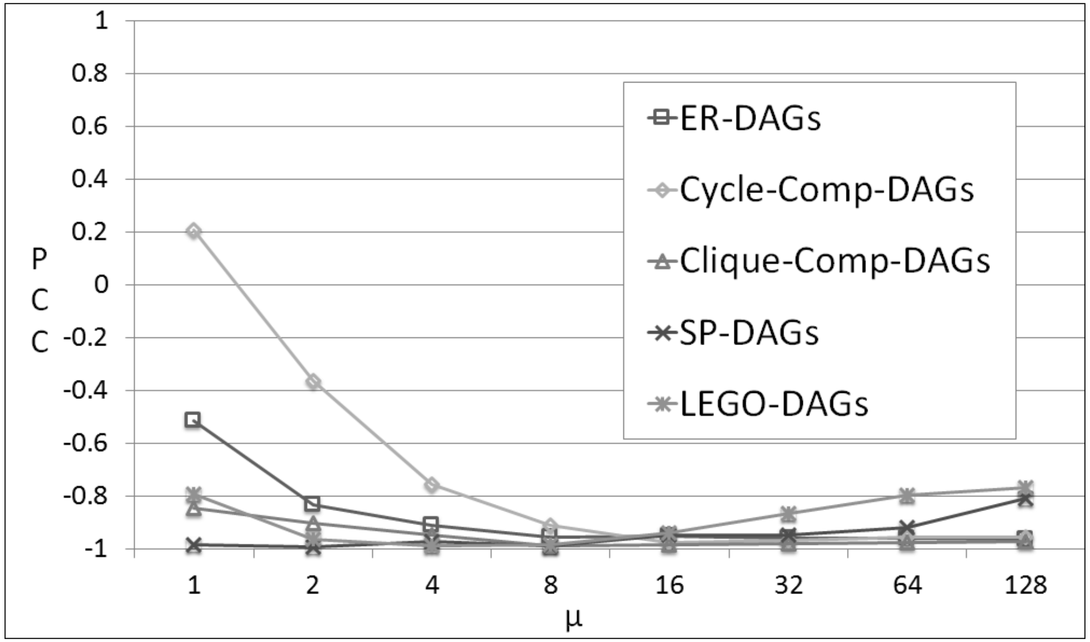
AO's schedules' AREAs vs. Competitors': Series-Parallel DAGs

Mean ratios and ranges: $AREA(AO) \div AREA(H)$



Does larger AREA mean smaller MAKESPAN?

Correlating AREA and MAKESPAN for various DAG-familiess

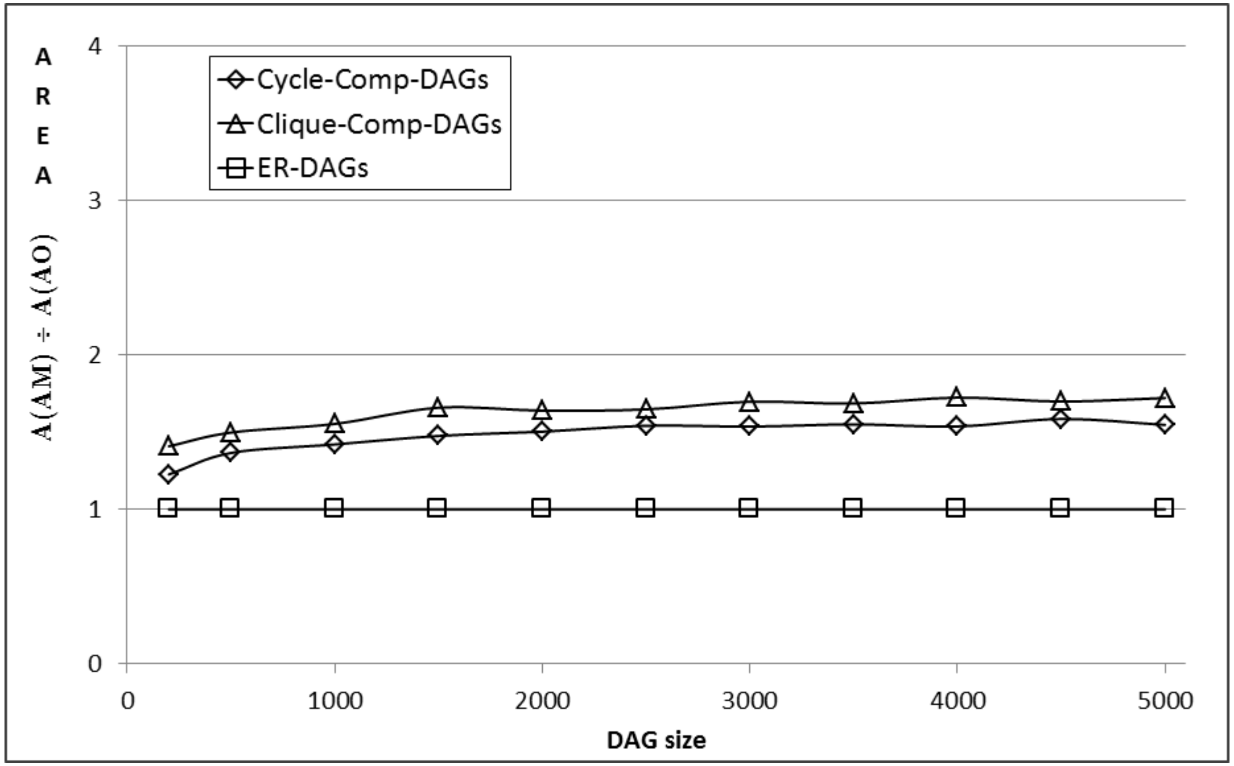


- μ is the "arrival rate" of available processors
- PCC is the *Pearson Product-Moment Correlation Coefficient* — measures correlation via both <u>strength</u> and <u>direction</u>

Comparing AREA-Maximization vs. AREA-Orientation via schedule AREA

Comparing the <u>AREAs</u> of A-M and A-O schedules

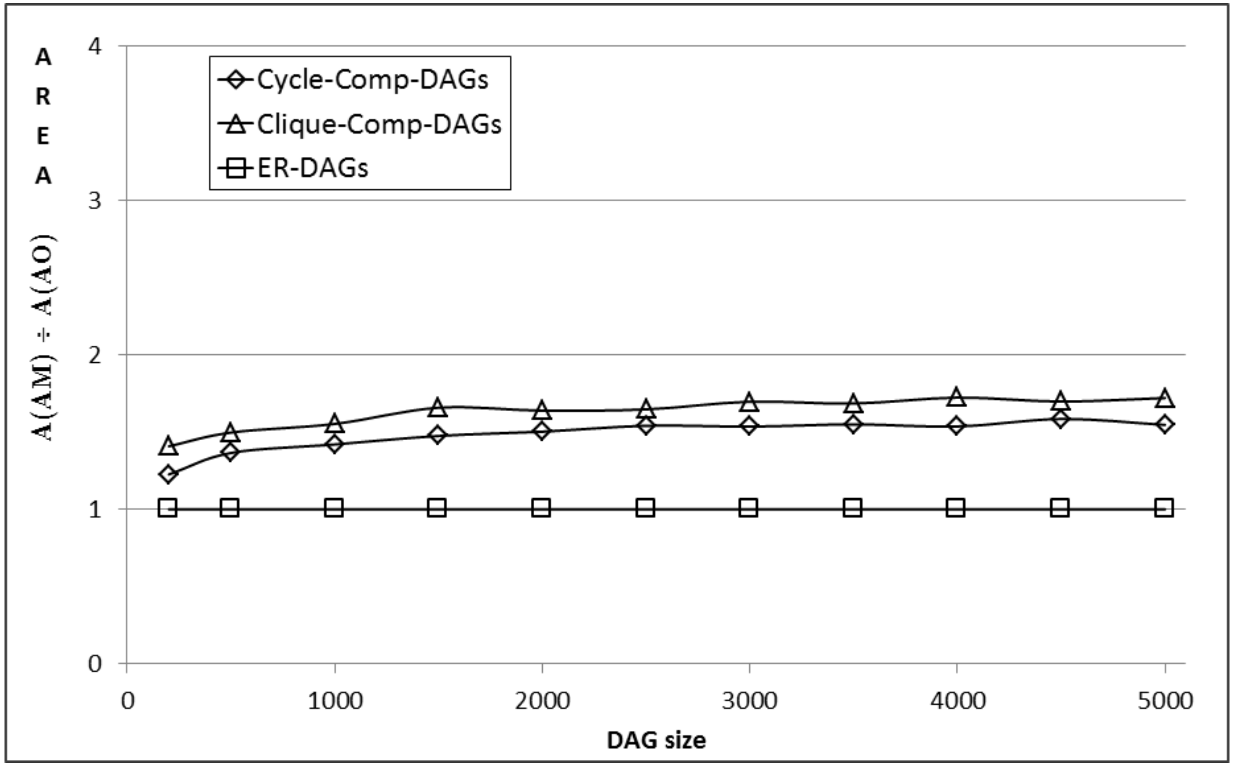
—via the ratio $AREA(AM) \div AREA(AO)$



Comparing AREA-Maximization vs. AREA-Orientation via schedule AREA

Comparing the AREAs of A-M and A-O schedules

—via the ratio $AREA(AM) \div AREA(AO)$



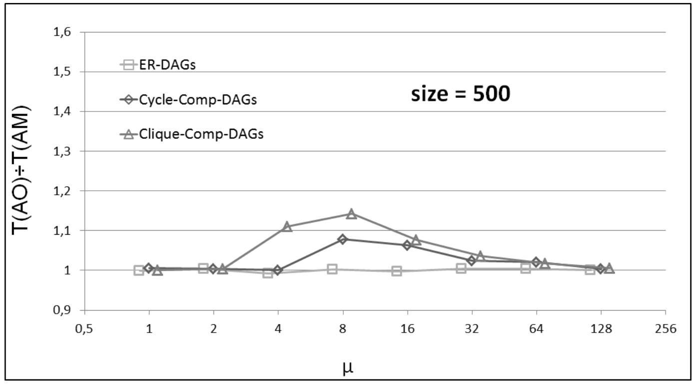
Observe that $AREA(AM) \ \leq \ 2 \times AREA(AO)$.

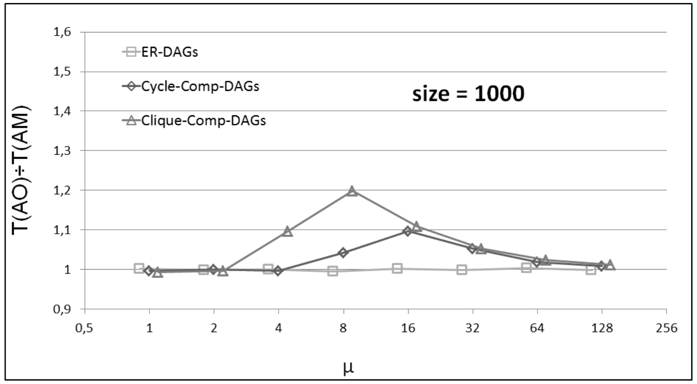
IS THIS ALWAYS TRUE?

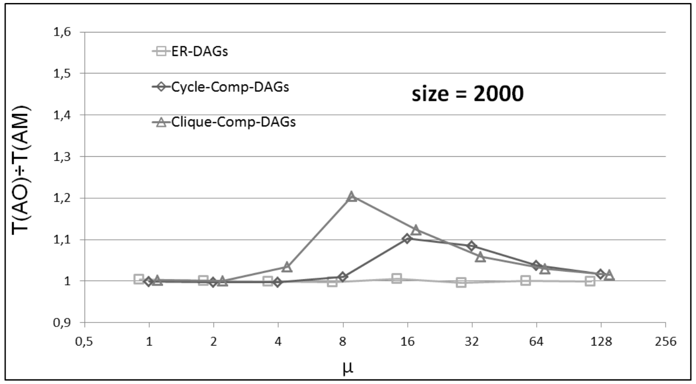
Comparing AREA-Maximization vs. AREA-Orientation via schedule MAKESPAN

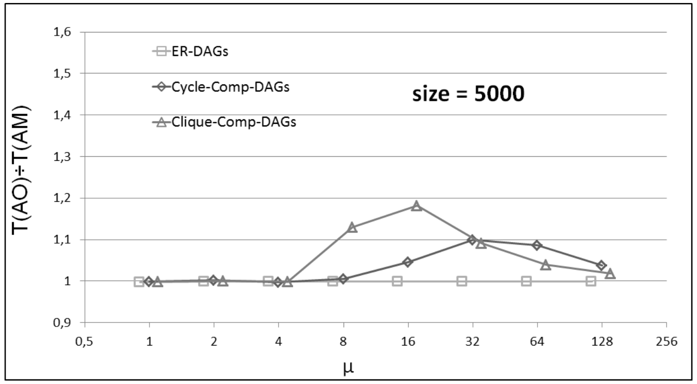
Comparing the MAKESPANs of A-M and A-O schedules

—via the ratio $T(AO) \div T(AM)$





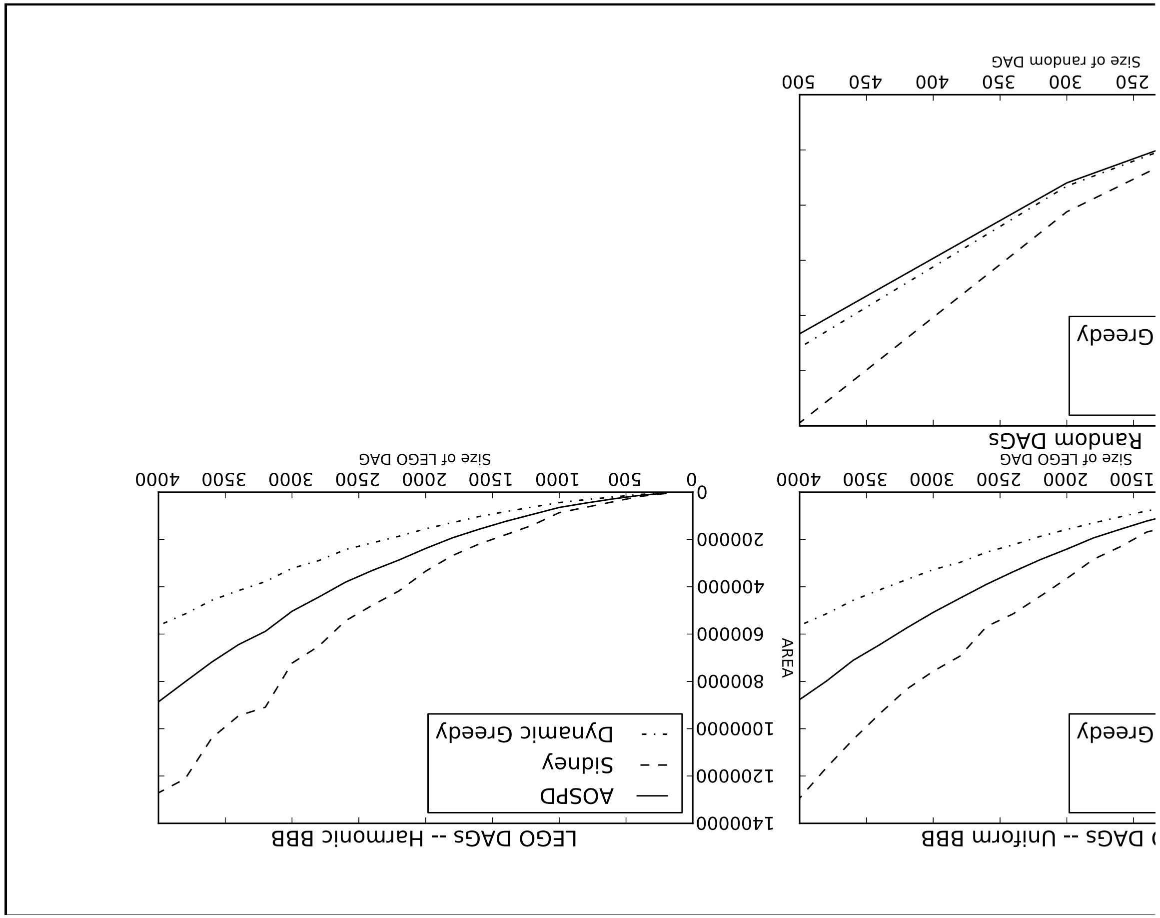


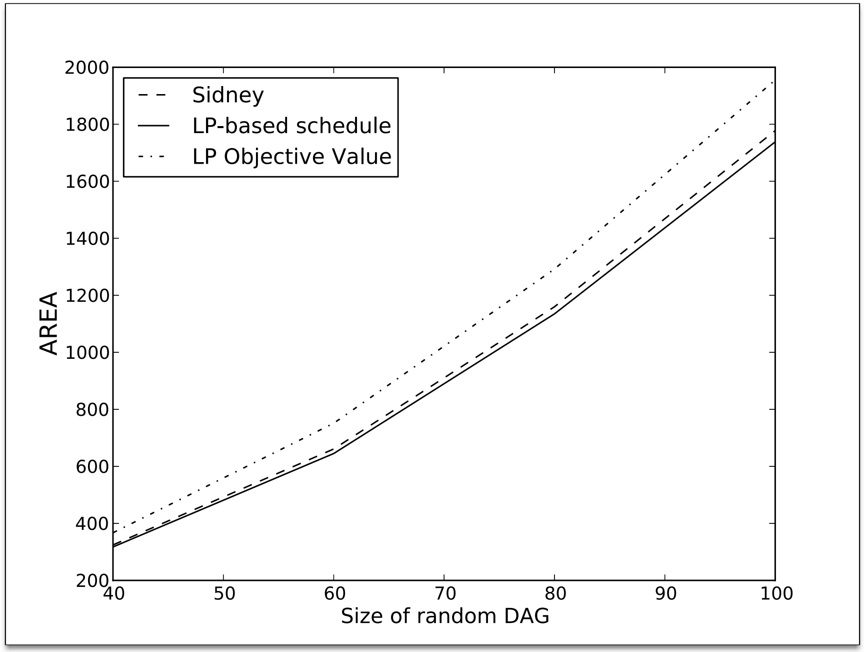


The SIDNEY Scheduling Heuristic

- Based on the Sidney decomposition of DAGs
- Experiments show that SIDNEY's schedules
  - have AREAs significantly larger than other heuristics'
  - have AREAs within 85% of optimal on randomly generated DAGs

The AREA-qualities of SIDNEY's schedules.




Sidney
LP-based schedule
LP Objective Value
AREA
Size of random DAG
2000
1800
1600
1400
1200
1000
800
600
400
200
40
50
60
70
80
90
100