IC-Optimal Schedules that Accommodate Heterogeneous Clients

Mark Sims
University of Massachusetts
Amherst, MA, USA

Gennaro Cordasco
Universita di Salerno
Fisciano, ITALY

Arnold L. Rosenberg
Univ. Massachusetts and Colorado State Univ.
Amherst, MA, USA Ft. Collins, CO, USA

A New Modality of *Collaborative Computing:* Internet-Based Computing (IC)

- The *owner* of a massive job enlists the aid of remote *clients* to compute the job's (compute-intensive) tasks.
- The owner (server) allocates tasks to clients, one at a time.
- A client receives its $(k+1)$th task after returning the results from its kth task.

Challenges in Internet-Based Computing

When jobs have *intertask dependencies* (modeled as *dags*)—

temporal unpredictability complicates scheduling of tasks.

Challenges in Internet-Based Computing

When jobs have *intertask dependencies* (modeled as *dags*)—

temporal unpredictability complicates scheduling of tasks:

- *Clients become available at unpredictable times.*

Challenges in Internet-Based Computing

When jobs have *intertask dependencies* (modeled as *dags*)—

temporal unpredictability complicates scheduling of tasks:

- *Clients become available at unpredictable times.*
- *Clients can be unexpectedly slow:*
  —*They are not dedicated.*

Challenges in Internet-Based Computing

When jobs have *intertask dependencies* (modeled as *dags*)—

temporal unpredictability complicates scheduling of tasks:

- *Clients become available at unpredictable times.*
- *Clients can be unexpectedly slow:*
  —*They are not dedicated.*
  —*They communicate over the Internet.*

Our Overall Goal

Determine how to schedule a *dag of tasks* in a way that—

Informally:

- *lessens the danger of a computation's stalling*

- *enhances the utilization of client resources*

Our Overall Goal

Determine how to schedule a *dag of tasks* in a way that—

Informally:

- *lessens the danger of a computation's stalling*

- *enhances the utilization of client resources*

Formally:

- *maximizes the number of tasks that are eligible for allocation at every step of the computation*

Formalizing the Theory's Framework/Goal

The Internet-Computing (IC) Scenario

- The *job* is represented by a (finite or infinite) dag $\mathcal{G}$:

The Internet-Computing (IC) Scenario

- The *job* is represented by a (finite or infinite) dag $\mathcal{G}$:
  - Each node of $\mathcal{G}$ represents a *task*.

The Internet-Computing (IC) Scenario

- The ***job*** is represented by a (finite or infinite) dag $\mathcal{G}$:
  - Each node of $\mathcal{G}$ represents a ***task***.
  - Arc $(u \rightarrow v)$ of $\mathcal{G}$ represents an ***intertask dependency***:
    $\rightarrow$task v cannot be *executed* until its ***parent*** task u is.

The Internet-Computing (IC) Scenario

- The *job* is represented by a (finite or infinite) dag $\mathcal{G}$:
  - Each node of $\mathcal{G}$ represents a *task*.
  - Arc $(u \rightarrow v)$ of $\mathcal{G}$ represents an *intertask dependency*:
    $\rightarrow$task v cannot be *executed* until its *parent* task u is.
  - Task v is ELIGIBLE (to be executed) when all of its parents *have been executed*.
    $\rightarrow$*source* (= parentless) tasks are ELIGIBLE immediately.

IC Quality/Optimality of a Schedule

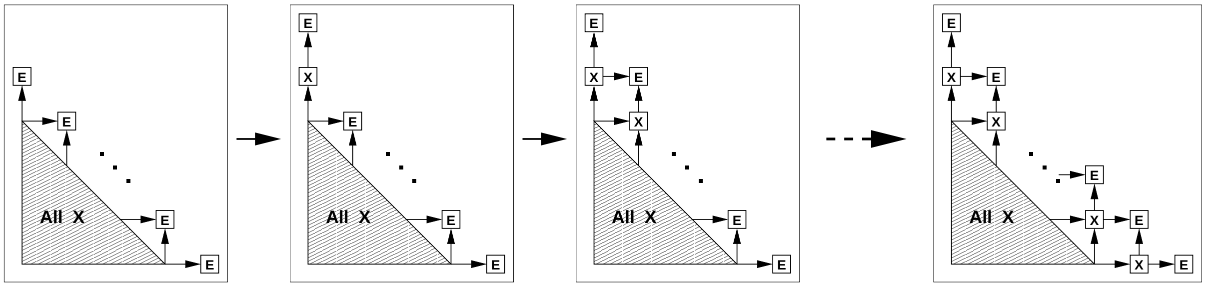
The **IC quality** of a schedule for a dag:

—the rate of producing ELIGIBLE nodes — *the larger, the better.*

Schedule Σ is **IC optimal**:

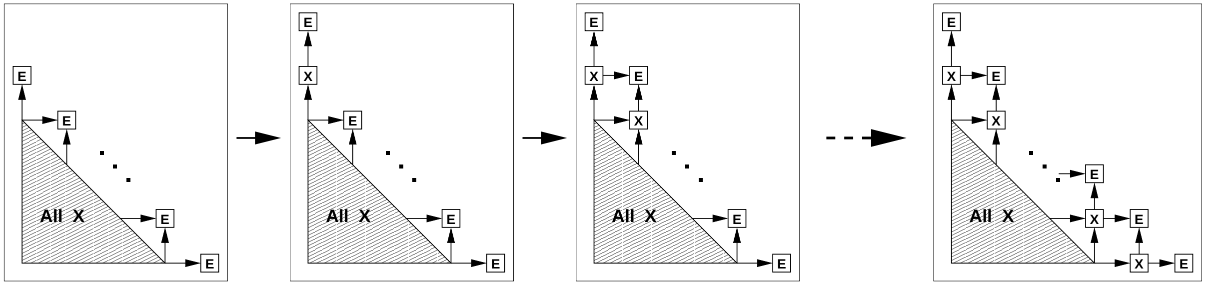
—It *maximizes* the number of ELIGIBLE nodes *for all steps* t.

How Important is IC Quality/Optimality?



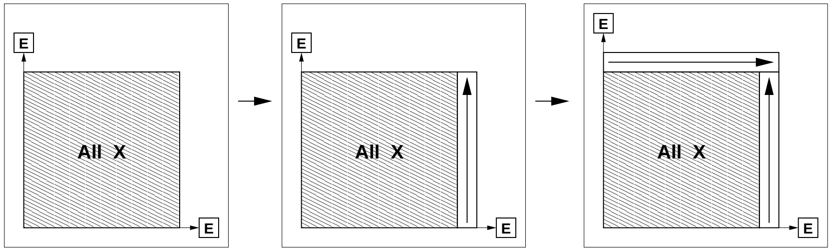
⇑ Roughly $\sqrt{T}$ ELIGIBLE nodes at step T ⇑

How Important is IC Quality/Optimality?



$\Uparrow$ Roughly $\sqrt{T}$ ELIGIBLE nodes at step T $\Uparrow$

$\Downarrow$ Never more than 3 ELIGIBLE nodes $\Downarrow$



Progress Thus Far

1. A formal framework for studying scheduling for IC

Progress Thus Far

1. A formal framework for studying scheduling for IC
2. Under idealized assumptions:
   (a) optimal scheduling strategies for familiar classes of dags:
   - 2-D *evolving meshes*
   - (2-D) *reduction-meshes*
   - (binary) *reduction-trees*
   - *butterfly dags*

Progress Thus Far

1. A formal framework for studying scheduling for IC

2. Under idealized assumptions:

   (a) optimal scheduling strategies for familiar classes of

   dags:
   - 2-D *evolving meshes*
   - (2-D) *reduction-meshes*
   - (binary) *reduction-trees*
   - *butterfly dags*

   computations:
   - *convolutions (FFT)*
   - *Discrete Laplace Transform*
   - *matrix multiplication*
   - *numerical integration*

Progress Thus Far

1. A formal framework for studying scheduling for IC

2. Under idealized assumptions:

   (a) optimal scheduling strategies for familiar classes of dags and computations

   (b) a foundation for an algorithmic scheduling theory (schedules "well-structured" dags optimally)

Progress Thus Far

1. A formal framework for studying scheduling for IC

2. Under idealized assumptions:

   (a) optimal scheduling strategies for familiar classes of dags and computations

   (b) a foundation for an algorithmic scheduling theory (schedules "well-structured" dags optimally)

3. Initial—*positive*—simulation-based assessment of computational impact

An Initial Assessment of the Theory's Impact

A Makespan-Based Experiment

- Generate random dags that admit IC-optimal schedules.

A Makespan-Based Experiment

- Generate random dags that admit IC-optimal schedules.
- For each dag, generate 50 random arrival patterns of Clients.

A Makespan-Based Experiment

- Generate random dags that admit IC-optimal schedules.
- For each dag, generate 50 random arrival patterns of Clients.
- Compare Makespan of IC-optimal schedule against:
  - the FIFO scheduler, which inserts new ELIGIBLE tasks on a FIFO queue, ordered by out-degree

A Makespan-Based Experiment

- Generate random dags that admit IC-optimal schedules.
- For each dag, generate 50 random arrival patterns of Clients.
- Compare Makespan of IC-optimal schedule against:
  - the FIFO scheduler, which inserts new ELIGIBLE tasks on a FIFO queue, ordered by out-degree
  - the LIFO scheduler, which inserts new ELIGIBLE tasks on a stack, ordered by out-degree

A Makespan-Based Experiment

- Generate random dags that admit IC-optimal schedules.
- For each dag, generate 50 random arrival patterns of Clients.
- Compare Makespan of IC-optimal schedule against:
  - the FIFO scheduler, which inserts new ELIGIBLE tasks on a FIFO queue, ordered by out-degree
  - the LIFO scheduler, which inserts new ELIGIBLE tasks on a stack, ordered by out-degree
  - the GREEDY scheduler, which inserts new ELIGIBLE tasks on a MAX-priority queue, ordered by out-degree.

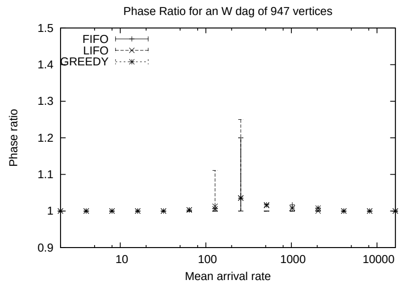
A Makespan-Based Experiment

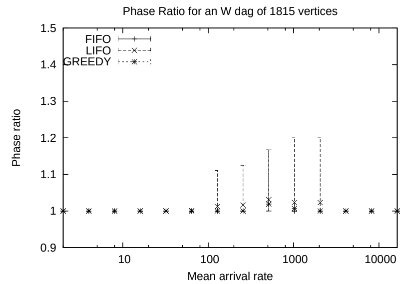
- Generate random dags that admit IC-optimal schedules.
- For each dag, generate 50 random arrival patterns of Clients.
- Compare Makespan of IC-optimal schedule against:
  - the FIFO scheduler, which inserts new ELIGIBLE tasks on a FIFO queue, ordered by out-degree
  - the LIFO scheduler, which inserts new ELIGIBLE tasks on a stack, ordered by out-degree
  - the GREEDY scheduler, which inserts new ELIGIBLE tasks on a MAX-priority queue, ordered by out-degree.

Task execution times distributed normally: mean$= 1$*; std_dev*$= 0.1$

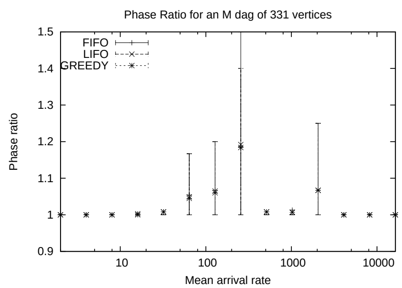
Mkspn-Based *Ratios:* Mks(heuristic) ÷ Mks(ICO)

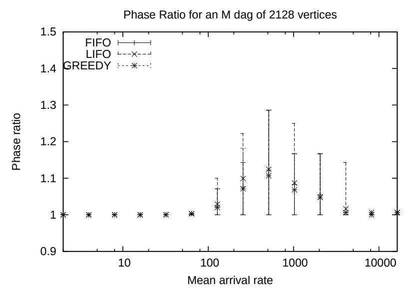
Two different expansive dags:





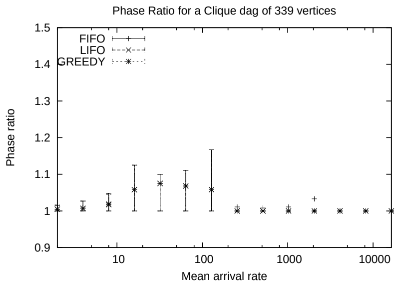
Two different reductive dags:

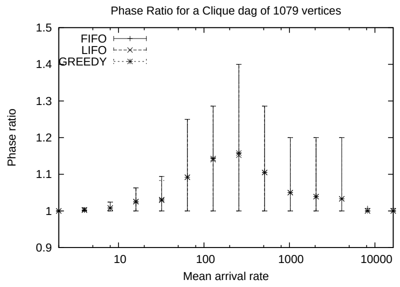




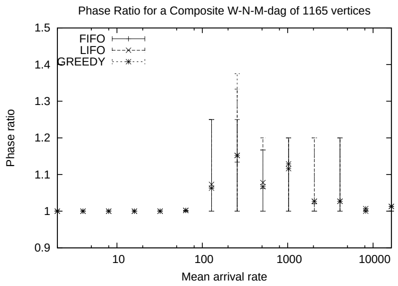
Mkspn-Based *Ratios:* Mks(heuristic) ÷ Mks(ICO)

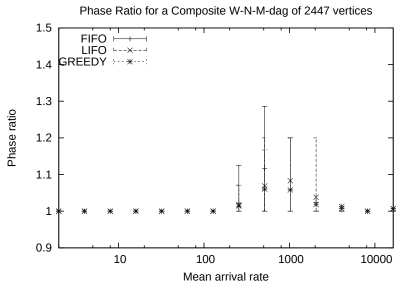
Two different clique-based dags (cycle-based are similar):





Two different expansive-reductive dags:





Progress Thus Far

1. A formal framework for studying scheduling for IC

2. Under idealized assumptions:

   (a) optimal scheduling strategies for familiar classes of dags and computations

   (b) a foundation for an algorithmic scheduling theory (schedules "well-structured" dags optimally)

3. Initial—*positive*—simulation-based assessment of computational impact

BUT—

THE THEORY TREATS ALL DAG NODES AS EQUIVALENT!

Progress Thus Far

1. A formal framework for studying scheduling for IC

2. Under idealized assumptions:

   (a) optimal scheduling strategies for familiar classes of dags and computations

   (b) a foundation for an algorithmic scheduling theory (schedules "well-structured" dags optimally)

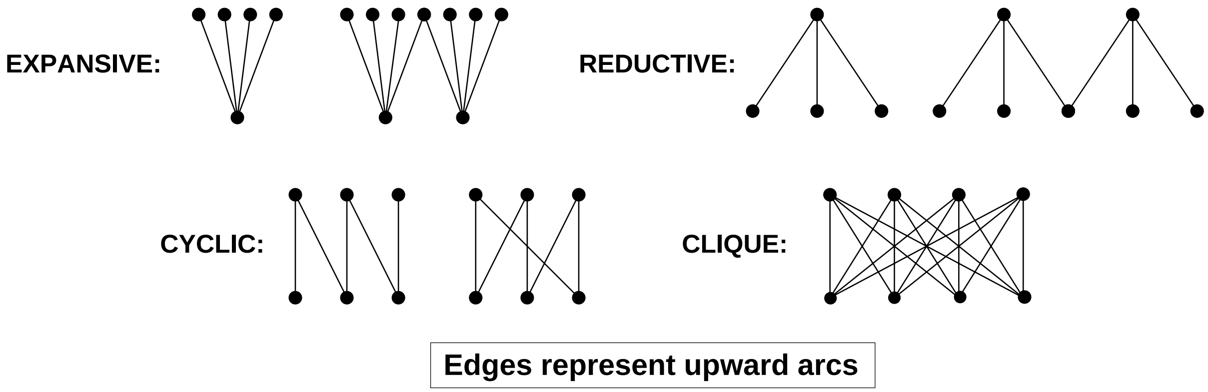
3. Initial—*positive*—simulation-based assessment of computational impact

HOW CAN WE DEAL WITH THE *HETEROGENEITY* OF REMOTE CLIENTS?

Toward a Decomposition-Based Scheduling Theory:

1. Select a Set of "Building Block" Dags

Start with *bipartite "building block" dags* that we know how to schedule optimally. A small sampler:



2. Establish "Priorities" among the Building Blocks

Say that $\begin{cases} \mathcal{G}_1 \text{ admits an IC-optimal schedule } \Sigma_1 \\ \mathcal{G}_2 \text{ admits an IC-optimal schedule } \Sigma_2 \end{cases}$

$\underline{\underline{\mathcal{G}_1 \rhd \mathcal{G}_2}}$ means:

To execute both $\mathcal{G}_1$ and $\mathcal{G}_2$, the following schedule is IC optimal:

1. Follow Σ_1 on $\mathcal{G}_1$
2. Follow Σ_2 on $\mathcal{G}_2$

2. Establish "Priorities" among the Building Blocks

Say that $\begin{cases} \mathcal{G}_1 \text{ admits an IC-optimal schedule } \Sigma_1 \\ \mathcal{G}_2 \text{ admits an IC-optimal schedule } \Sigma_2 \end{cases}$

$\underline{\underline{\mathcal{G}_1 \rhd \mathcal{G}_2}}$ means:

To execute both $\mathcal{G}_1$ and $\mathcal{G}_2$, the following schedule is IC optimal:

1. Follow Σ_1 on $\mathcal{G}_1$ 2. Follow Σ_2 on $\mathcal{G}_2$

≈≈≈≈≈≈≈≈≈

The relation $\rhd$ is: • *transitive* • *easily tested*.

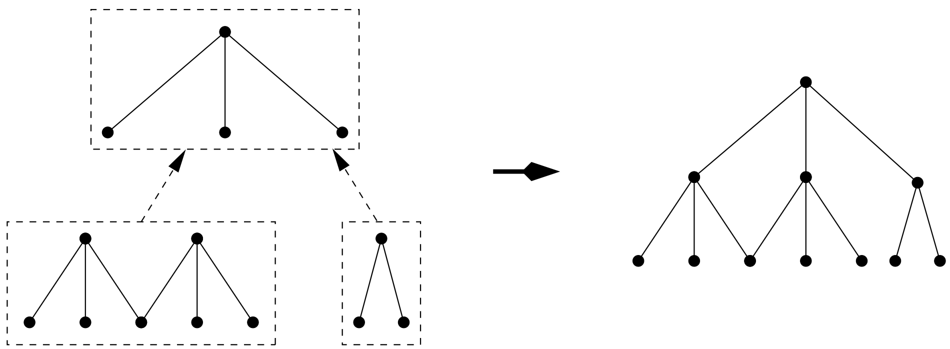
Complex Dags via "Composition"

Compose $\mathcal{G}_1$ with $\mathcal{G}_2$:

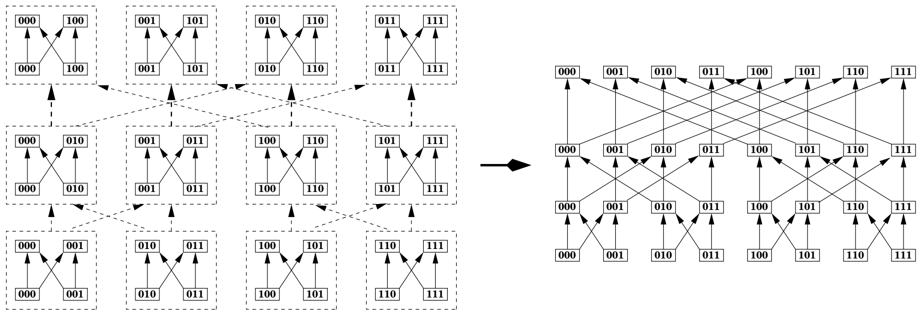
Merge/Identify some k sources of $\mathcal{G}_2$ with some k sinks of $\mathcal{G}_1$.

The dag obtained is *composite of type* $\mathcal{G}_1 \Uparrow \mathcal{G}_2$.

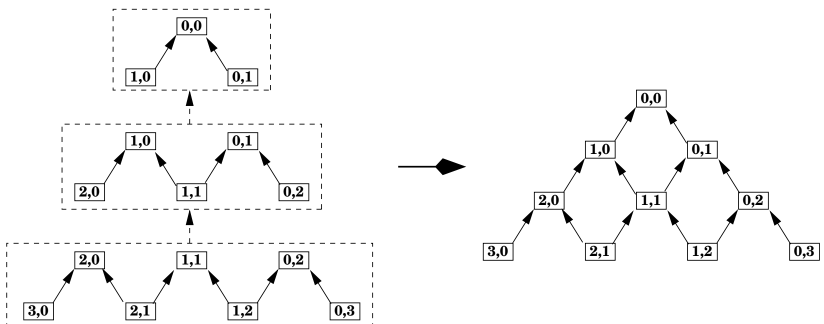
Example: $\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \mathcal{G}_3$ (*Composition is associative.*)



Familiar Dags as Compositions of Building Blocks



≈≈≈≈≈≈≈



Why "Composition" and "Priority" Are Important

Theorem.

IF:

- *the dag $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \cdots \Uparrow \mathcal{G}_n$*
- *each $\mathcal{G}_i$ admits the IC-optimal schedule Σ_i*
- $\mathcal{G}_1 \triangleright \mathcal{G}_2 \triangleright \cdots \triangleright \mathcal{G}_n$

THEN: the following schedule for $\mathcal{G}$ is IC optimal:

Execute $\mathcal{G}$ by executing each $\mathcal{G}_i$ (using Σ_i) in $\triangleright$-order.

Why “Composition” and “Priority” Are Important

Theorem.

IF:

- *the dag $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \cdots \Uparrow \mathcal{G}_n$*
- *each $\mathcal{G}_i$ admits the IC-optimal schedule Σ_i*
- $\mathcal{G}_1 \triangleright \mathcal{G}_2 \triangleright \cdots \triangleright \mathcal{G}_n$

THEN: the following schedule for $\mathcal{G}$ is IC optimal:

Execute $\mathcal{G}$ by executing each $\mathcal{G}_i$ (using Σ_i) in $\triangleright$-order.

≈≈≈≈≈≈≈≈≈

- Parsing $\mathcal{G}$ into $\mathcal{G}_1, \ldots, \mathcal{G}_n$
- Testing $\triangleright$-priorities

} are *computationally efficient*.

Why "Composition" and "Priority" Are Important

Theorem.

IF:

- *the dag $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \cdots \Uparrow \mathcal{G}_n$*
- *each $\mathcal{G}_i$ admits the IC-optimal schedule Σ_i*
- $\mathcal{G}_1 \rhd \mathcal{G}_2 \rhd \cdots \rhd \mathcal{G}_n$

THEN: *the following schedule for $\mathcal{G}$ is IC optimal:*

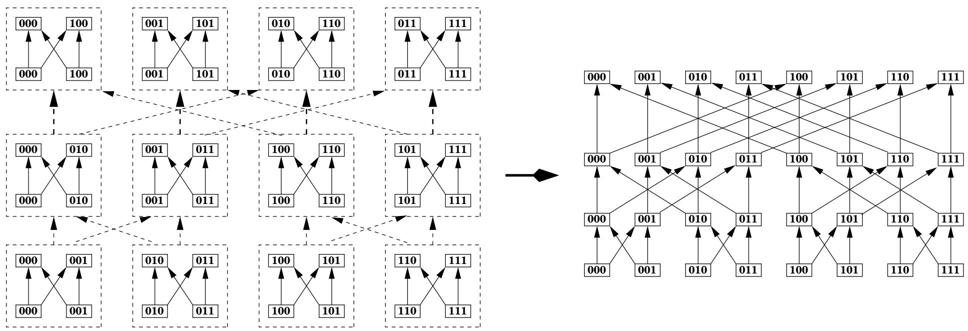
Execute $\mathcal{G}$ by executing each $\mathcal{G}_i$ (using Σ_i) in $\rhd$-order.

≈≈≈≈≈≈≈≈≈

EFFICIENT ALGORITHMS IMPLEMENT THIS THEOREM
ON A LARGE CLASS OF "WELL-STRUCTURED" DAGS

Clarification 1

Even if $\mathcal{G}_1 \triangleright \mathcal{G}_2 \triangleright \cdots \triangleright \mathcal{G}_n$, the composition $\mathcal{G}$ can be *very* nonlinear:

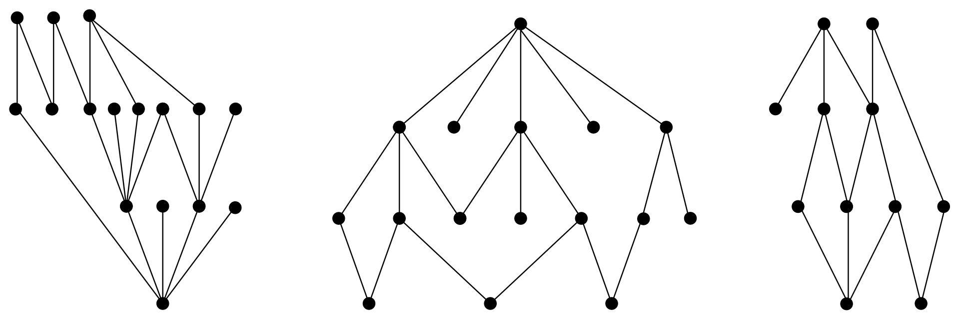


The building-block butterfly $\mathcal{B}$ has "self $\triangleright$-priority," so that—

$$\mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B} \triangleright \mathcal{B}$$

Clarification 2

Composite dags that admit IC-optimal schedules can be *very* nonuniform in structure:



Clarification 3

We have other systematic ways of crafting IC-optimal schedules

Clarification 3

We have other systematic ways of crafting IC-optimal schedules,

—but the "$\triangleright$-priority chain" method has many benefits

Clarification 3

We have other systematic ways of crafting IC-optimal schedules,

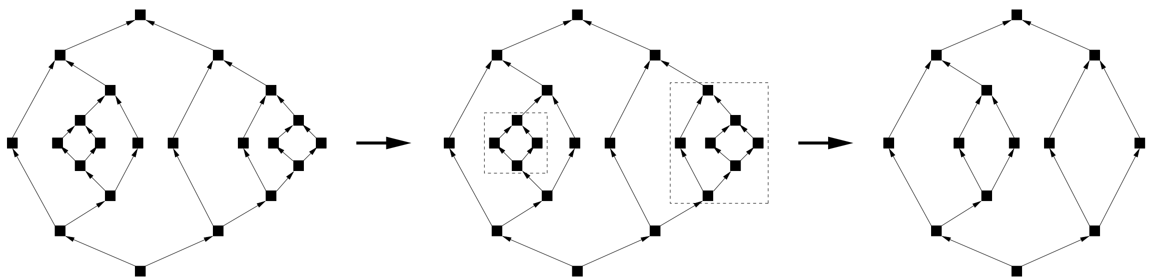
—but the "$\triangleright$-priority chain" method has many benefits

—including "perturbability."

Task Clustering that Preserves IC Optimality

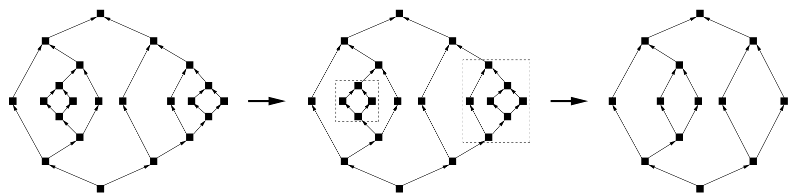
Two Ad Hoc Task-Clusterings (for intuition)

A Divide-and-Conquer Computation:

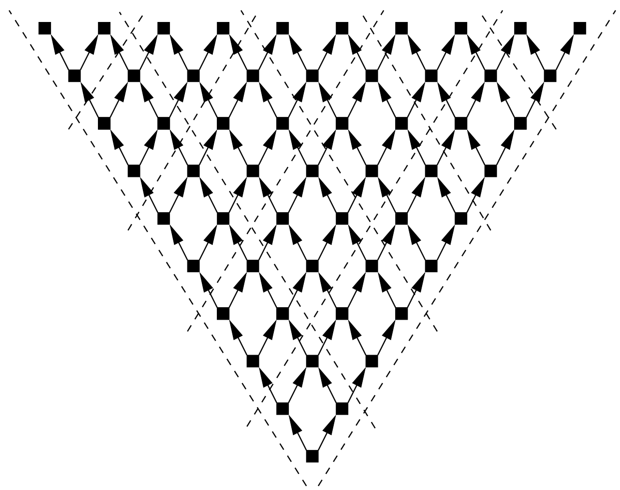


Two Ad Hoc Task-Clusterings (for intuition)

A Divide-and-Conquer Computation:



A Wavefront Computation:



Toward Formal Task-Clusterings

A fattened task F in dag $\mathcal{G}$.

A *self-contained* set of nodes of $\mathcal{G}$:

- Every node $v \in F$ is ELIGIBLE — OR
- All of v's parents are also in F.

Toward Formal Task-Clusterings

A fattened task F in dag $\mathcal{G}$.

A *self-contained* set of nodes of $\mathcal{G}$:

- Every node $v \in F$ is ELIGIBLE — OR
- All of v's parents are also in F.

WE WANT FATTENED TASKS OF MANY SIZES

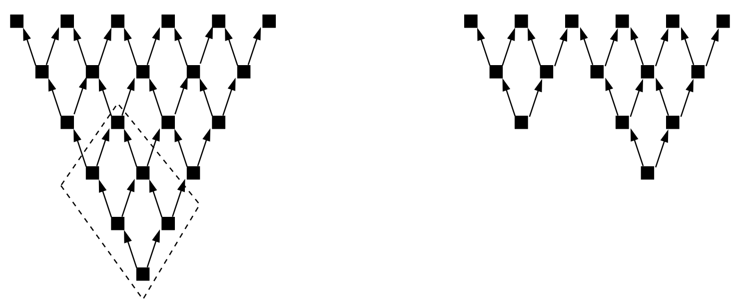
Toward Formal Task-Clusterings

A fattened task F in dag $\mathcal{G}$.

A *self-contained* set of nodes of $\mathcal{G}$:

- Every node $v \in F$ is ELIGIBLE — OR
- All of v's parents are also in F.

The residual dag $\mathcal{G}^{(F)}$ when F is removed from $\mathcal{G}$



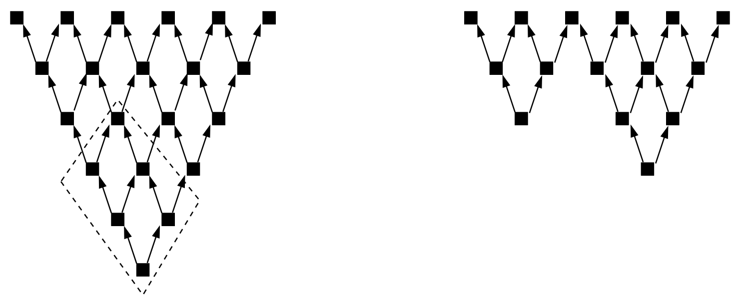
Toward Formal Task-Clusterings

A fattened task F in dag $\mathcal{G}$.

A *self-contained* set of nodes of $\mathcal{G}$:

- Every node $v \in F$ is ELIGIBLE — OR
- All of v's parents are also in F.

The residual dag $\mathcal{G}^{(F)}$ when F is removed from $\mathcal{G}$



WHEN $\mathcal{G}$ ADMITS AN IC-OPTIMAL SCHEDULE
WE WANT TO ENSURE THAT $\mathcal{G}^{(F)}$ DOES, TOO

The *Direct* Task-Clustering Strategy

One can view a schedule Σ for dag $\mathcal{G}$ as an *injection*

$$\Sigma : \mathcal{N}(\mathcal{G}) \longrightarrow \{1, 2, \ldots, |\mathcal{N}(\mathcal{G})|\}$$

The *Direct* Task-Clustering Strategy

One can view a schedule Σ for dag $\mathcal{G}$ as an *injection*

$$\Sigma : \mathcal{N}(\mathcal{G}) \longrightarrow \{1, 2, \ldots, |\mathcal{N}(\mathcal{G})|\}$$

For a k-node fattened task F, choose

$$\{\Sigma^{-1}(1),\ \Sigma^{-1}(2), \ldots,\ \Sigma^{-1}(k)\}$$

IF $\mathcal{G}$ ADMITS AN IC-OPTIMAL SCHEDULE THEN $\mathcal{G}^{(F)}$ DOES ALSO

The *Direct* Task-Clustering Strategy

One can view a schedule Σ for dag $\mathcal{G}$ as an *injection*

$$\Sigma : \mathcal{N}(\mathcal{G}) \longrightarrow \{1, 2, \ldots, |\mathcal{N}(\mathcal{G})|\}$$

For a k-node fattened task F, choose

$$\{\Sigma^{-1}(1),\ \Sigma^{-1}(2), \ldots,\ \Sigma^{-1}(k)\}$$

IF $\mathcal{G}$ ADMITS AN IC-OPTIMAL SCHEDULE
THEN $\mathcal{G}^{(F)}$ DOES ALSO

THIS WORKS FOR ANY k

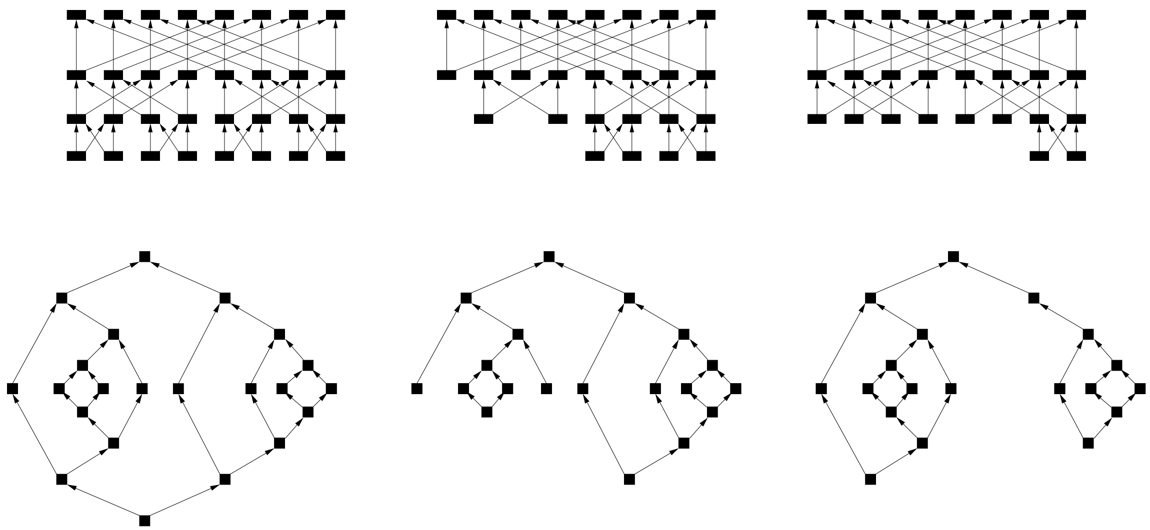
The *Direct* Task-Clustering Strategy—*and Competitors*

WAIT!! THE STORY IS NOT OVER!

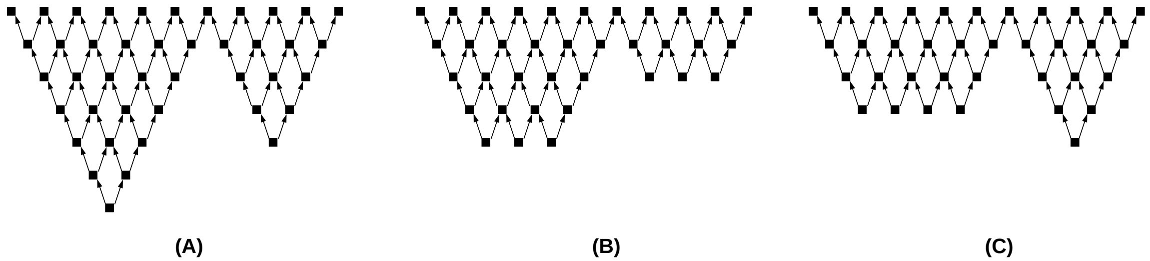
The *Direct* Task-Clustering Strategy—*and Competitors*

THE STORY IS NOT OVER!

Different IC-optimal schedules lead to very different residual dags



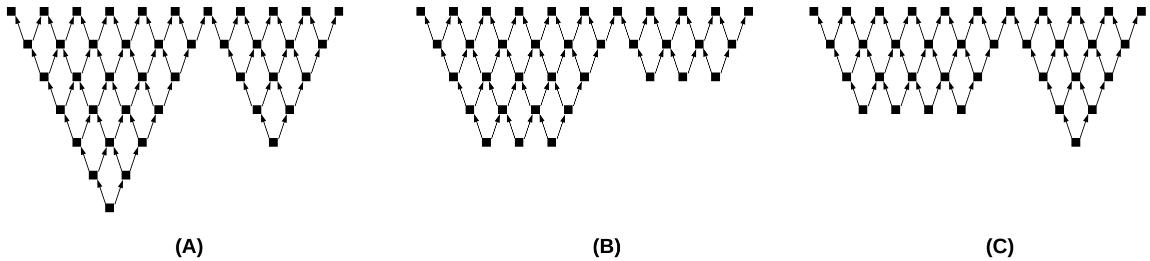
THE STORY IS ***REALLY*** NOT OVER!



(A) original dag $\mathcal{G}$

The *Direct* Task-Clustering Strategy—*and Competitors*

THE STORY IS *REALLY* NOT OVER!

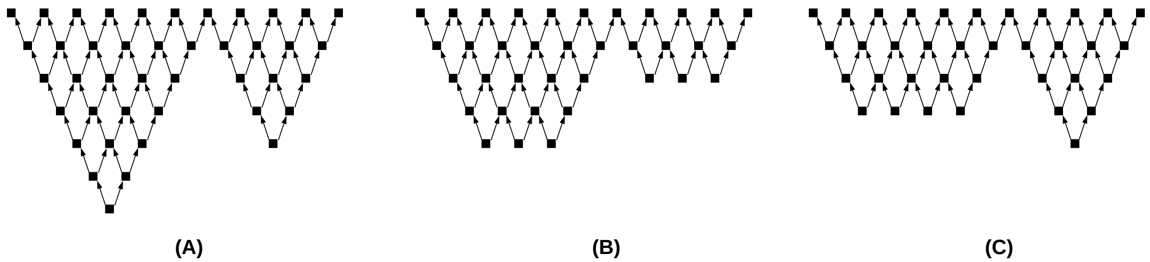


(A) original dag $\mathcal{G}$

(B)
- F_1 is a 6-node fattened task via IC-optimal schedule
- residual dag $\mathcal{G}^{(F_1)}$ admits IC-optimal schedule
- 8 arcs "cut" when removing F_1 from $\mathcal{G}$

The *Direct* Task-Clustering Strategy—*and Competitors*

THE STORY IS *REALLY* NOT OVER!



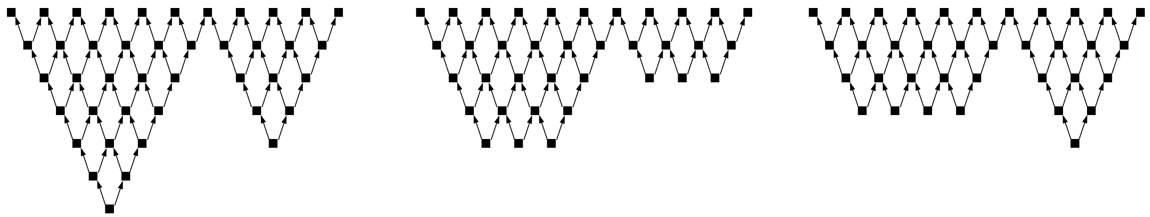
(A) original dag $\mathcal{G}$

(B)
- F_1 is a 6-node fattened task via IC-optimal schedule
- residual dag $\mathcal{G}^{(F_1)}$ admits IC-optimal schedule
- 8 arcs "cut" when removing F_1 from $\mathcal{G}$

(C)
- F_2 is a 6-node fattened task — *not* via IC-optimal schedule
- residual dag $\mathcal{G}^{(F_2)}$ admits IC-optimal schedule
- 6 arcs "cut" when removing F_2 from $\mathcal{G}$

The *Direct* Task-Clustering Strategy—*and Competitors*

THE STORY IS <u>*REALLY*</u> NOT OVER!



(A) (B) (C)

- (A) original dag $\mathcal{G}$
- (B)
  - F_1 is a 6-node fattened task via IC-optimal schedule
  - residual dag $\mathcal{G}^{(F_1)}$ admits IC-optimal schedule
  - 8 arcs "cut" when removing F_1 from $\mathcal{G}$
- (C)
  - F_2 is a 6-node fattened task — *not* via IC-optimal schedule
  - residual dag $\mathcal{G}^{(F_2)}$ admits IC-optimal schedule
  - 6 arcs "cut" when removing F_2 from $\mathcal{G}$

"CUT ARCS" ARE RESULTS FROM CLIENT TO SERVER

—So direct task-clusterings need not minimize communication cost!

Where Did the Competitors Come From?

Say that

- $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \cdots \Uparrow \mathcal{G}_n$
  each $\mathcal{G}_i$ admits an IC-optimal schedule
- $\mathcal{G}_1 \rhd \mathcal{G}_2 \rhd \cdots \rhd \mathcal{G}_n$

—so $\mathcal{G}$ admits an IC-optimal schedule.

Construct a fattened task by selecting any sequence of dags $\mathcal{G}_i$:

$$\mathcal{G}_{i_1} \rhd \mathcal{G}_{i_2} \rhd \cdots \rhd \mathcal{G}_{i_k} \quad \text{where} \quad i_1 < i_2 < \cdots < i_k$$

such that

the set F of all sources of the selected $\{\mathcal{G}_{i_j}\}_{j=1}^{k}$ is *self-contained*.

Where Did the Competitors Come From?

Say that

- $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \cdots \Uparrow \mathcal{G}_n$
  each $\mathcal{G}_i$ admits an IC-optimal schedule
- $\mathcal{G}_1 \rhd \mathcal{G}_2 \rhd \cdots \rhd \mathcal{G}_n$

—so $\mathcal{G}$ admits an IC-optimal schedule.

Construct a fattened task by selecting any sequence of dags $\mathcal{G}_i$:

$$\mathcal{G}_{i_1} \rhd \mathcal{G}_{i_2} \rhd \cdots \rhd \mathcal{G}_{i_k} \quad \text{where} \quad i_1 \ < \ i_2 \ < \cdots < \ i_k$$

such that

the set F of all sources of the selected $\{\mathcal{G}_{i_j}\}_{j=1}^{k}$ is *self-contained*.

THEN $\mathcal{G}^{(F)}$ ADMITS AN IC-OPTIMAL SCHEDULE.

Where Did the Competitors Come From?

Say that

- $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \cdots \Uparrow \mathcal{G}_n$
  each $\mathcal{G}_i$ admits an IC-optimal schedule
- $\mathcal{G}_1 \rhd \mathcal{G}_2 \rhd \cdots \rhd \mathcal{G}_n$

—so $\mathcal{G}$ admits an IC-optimal schedule.

Construct a fattened task by selecting any sequence of dags $\mathcal{G}_i$:

$$\mathcal{G}_{i_1} \rhd \mathcal{G}_{i_2} \rhd \cdots \rhd \mathcal{G}_{i_k} \quad \text{where} \quad i_1 < i_2 < \cdots < i_k$$

such that

the set F of all sources of the selected $\{\mathcal{G}_{i_j}\}_{j=1}^{k}$ is *self-contained*.

THEN $\mathcal{G}^{(F)}$ ADMITS AN IC-OPTIMAL SCHEDULE.

THIS FOLLOWS FROM THE TRANSITIVITY OF $\rhd$.

Where Did the Competitors Come From?

Say that

- $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \cdots \Uparrow \mathcal{G}_n$
  each $\mathcal{G}_i$ admits an IC-optimal schedule
- $\mathcal{G}_1 \triangleright \mathcal{G}_2 \triangleright \cdots \triangleright \mathcal{G}_n$

—so $\mathcal{G}$ admits an IC-optimal schedule.

Construct a fattened task by selecting any sequence of dags $\mathcal{G}_i$:

$$\mathcal{G}_{i_1} \triangleright \mathcal{G}_{i_2} \triangleright \cdots \triangleright \mathcal{G}_{i_k} \quad \text{where} \quad i_1 \ < \ i_2 \ < \cdots < \ i_k$$

such that

the set F of all sources of the selected $\{\mathcal{G}_{i_j}\}_{j=1}^{k}$ is *self-contained*.

THEN $\mathcal{G}^{(F)}$ ADMITS AN IC-OPTIMAL SCHEDULE.

THIS ALLOWS US TO OPTIMIZE OTHER CRITERIA ALSO,
E.G., COMMUNICATION

Stronger, but More Limited Clustering

We have identified several large families of dags that are *universal donors*

For every fattened task F, $\mathcal{G}^{(F)}$ admits an IC-optimal schedule.

Stronger, but More Limited Clustering

We have identified several large families of dags that are *universal donors*

For <u>every</u> fattened task F, $\mathcal{G}^{(F)}$ admits an IC-optimal schedule.

<u>SOME EXAMPLES</u>: