IC-Scheduling Theory: A New Scheduling Paradigm for Internet-Based Computing

Gennaro Cordasco
University of Salerno, Italy

Collaborators:
Arnold L. Rosenberg — Univ. of Massachusetts Amherst
Greg Malewicz — Google Inc.

Outline

Motivation

The Internet-Based Computing Pebble Game

Decomposition-Based Scheduling Theory

- Priority Relation
- Dags Composition

Expanding the repertoire of building-block *dags*

- Planar Bipartite Trees
- Exploiting Duality when Scheduling Dags
- The ICO-Sweep Algorithm

Familiar Computations with IC-Optimal Schedules

Project in Progress

Motivation

- Internet-Based computing platform
  - Web Computing
  - Grid Computing
  - P2P Computing

  Why Internet-Based Computing?
  - Remuneration (Commercial computational grids)
  - Reciprocation (Open computational grids)
  - Altruism (e.g. FightAids@home)
  - Curiosity (e.g. Seti@home)

Internet-Based Computing (IC)

The "*owner*" of a massive job enlists the aid of remote *clients* to compute the job's tasks.

The owner (server) allocates tasks to clients one task at a time.

A client receives its $(k+1)$-th task after returning the results from its k-th task.

Challenges in Internet-Based Computing

Focus on jobs that have inter-task dependencies (modeled as *dags*) we want to enhance their utilization.

Unfortunately, IC platforms are characterized by a temporal unpredictability:

- communication takes place over the internet
- remote clients are not dedicated, hence can be unexpectedly slow

Temporal unpredictability precludes use of "standard" strategies that were developed for older platform.

An Avenue of Idealization

- <u>Fact</u>

  Without further assumption, adversarial Clients can confute any strategy the Server adopts.

- <u>Fact</u> (Buyya-Abramson-Giddy, Kondo-Casanova-Wing-Berman, Sun-Wu)

  Monitoring client's past performance and present resources allows one to:
  - mitigate the degree of temporal unpredictability;
  - match task complexity to client resources.

- <u>Idealization</u>

  Via monitoring, one can "approximately" ensure the temporal unpredictability of clients affects <u>the timing</u>, but <u>not the order</u> of task executions.

The Formal Idealization

- <u>Assumption:</u> Tasks are executed in the order of allocation.

  This assumption allows us to:
  - let “time” be event-driven (execute a node at each “step”)
  - derive scheduling guidelines that are totally under control of the Server.

The Computation-*dag* $\mathcal{G}$

A *dag* $\mathcal{G}$=($\mathcal{N}$,$\mathcal{A}$) is used to model a computation (computation-*dag*):

– each node $v \in \mathcal{N}$ represents a task in the computation;

– an arc $(u \rightarrow v) \in \mathcal{A}$ represents the dependence of task v on task u: v cannot be executed until u is.

– Given an arc $(u \rightarrow v) \in \mathcal{A},$ u is *parent* of $v,$ and v is *child* of u in $\mathcal{G}.$ Each parentless node of $\mathcal{G}$ is a *source* (node), and each childless node is a *sink* (node).

Our Overall Goal

Determine how to schedule a *DAG* of tasks is such a way that

Informally

- the danger of gridlock is lessened
- the utilization of available client resources is enhanced

Formally

- the number of tasks that are eligible for allocation is maximized at every step of the computation

The IC Pebble Game

The Players

- A single Server, S (the owner)
- A (finite or infinite) set of Clients, C_1, C_2, …

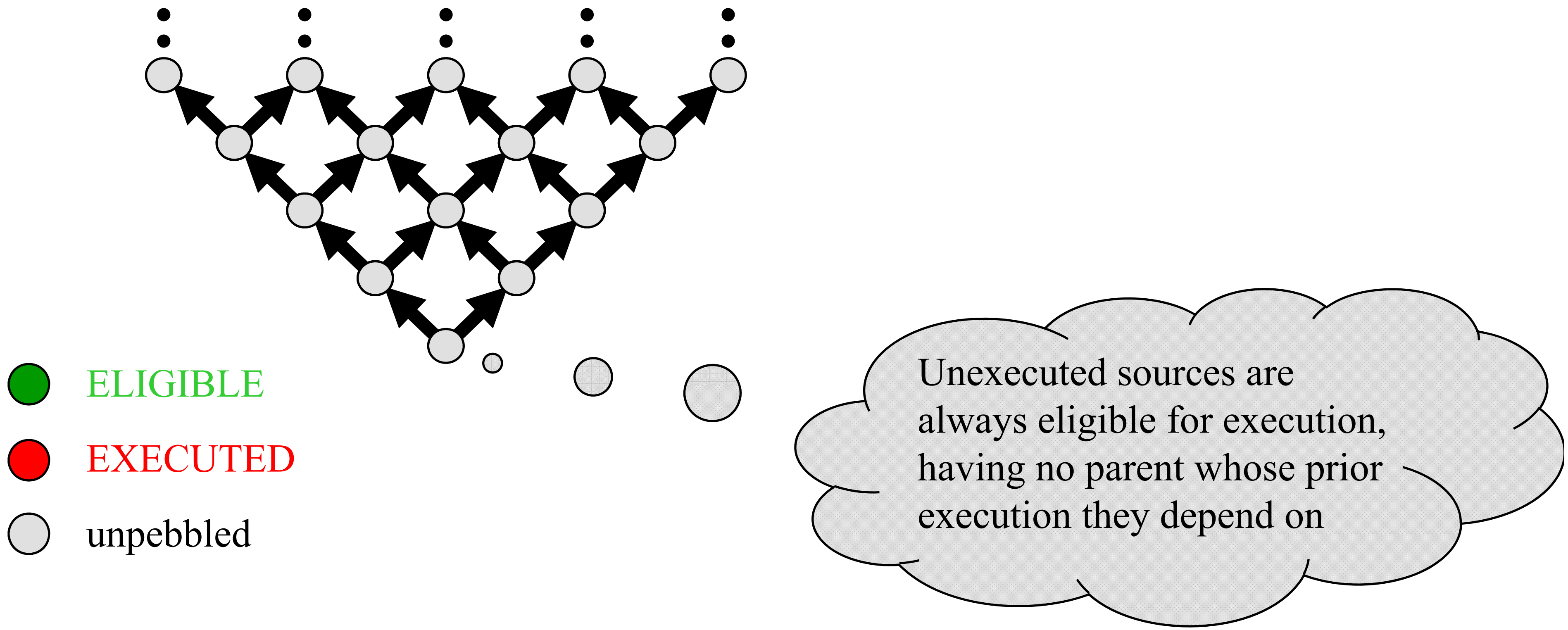
S has unlimited supplies of two types of *Pebbles:*

- ELIGIBLE pebbles, whose presence indicates a task eligible for execution
- EXECUTED pebbles, whose presence indicates an executed task

The IC Pebble Game

Rules of the Game

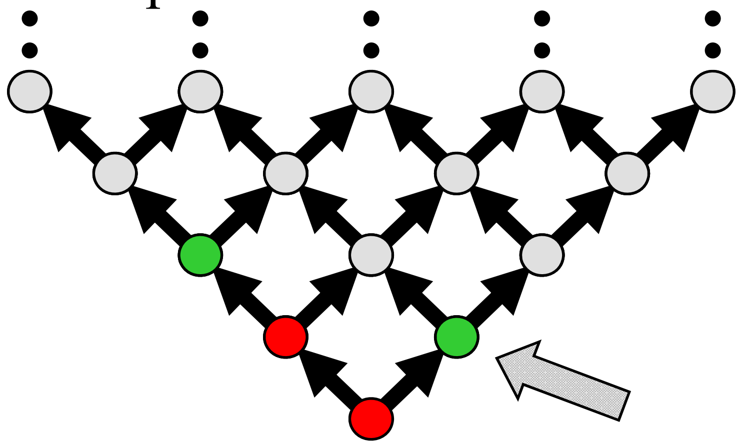
1. S begins by placing an ELIGIBLE pebble on each unpebbled source of $\mathcal{G}$.



The IC Pebble Game

Rules of the Game

1. S begins by placing an ELIGIBLE pebble on each unpebbled source of $\mathcal{G}$.
2. At each step, S
   - selects a node that contains an ELIGIBLE pebble,
   - replaces that pebble by an EXECUTED pebble,
   - places an ELIGIBLE pebble on each unpebbled node of $\mathcal{G}$ all of whose parents contain EXECUTED pebbles.



ELIGIBLE

EXECUTED

unpebbled

IC Quality/Optimality of a *Dag*-Schedule

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

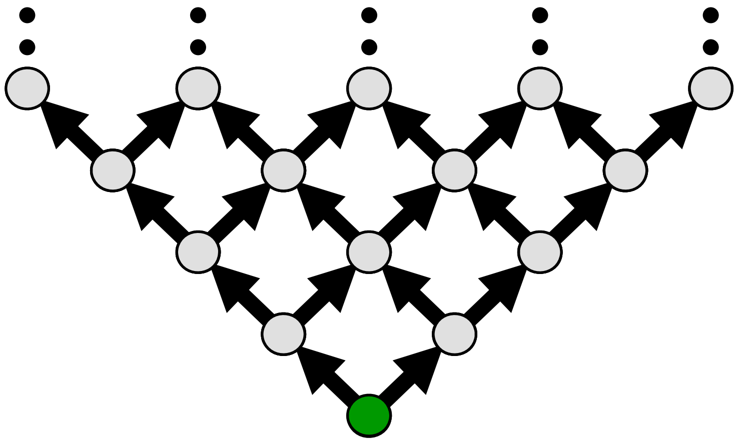
- the rate of producing ELIGIBLE nodes - the larger, the better.

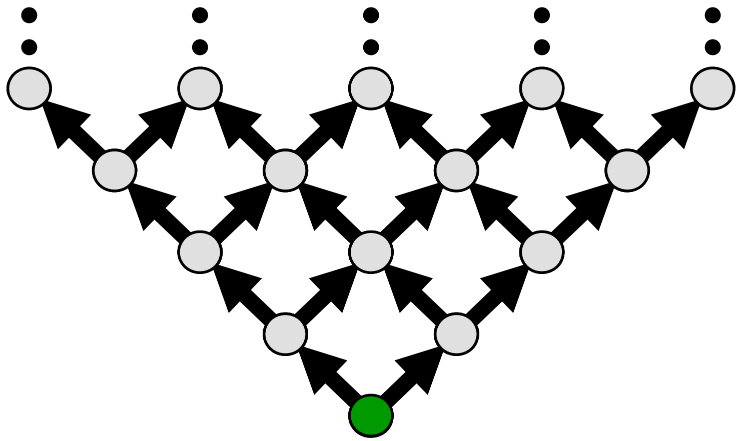
A schedule for a dag is *IC optimal* (ICO):

- It maximizes the number of ELIGIBLE nodes for all steps.

How Important is IC Quality?

- Consider the following *dag*:





Non-optimal schedule: Never more than 2 ELIGIBLE nodes

Optimal schedule: Roughly $t^{1/2}$ ELIGIBLE nodes at step t

ELIGIBLE

EXECUTED

unpebbled

Optimality is not always possible

For each step t of a play of the Game on a dag $\mathcal{G}$ under a schedule Σ: $E_{\Sigma}(t)$ denotes the number of nodes of $\mathcal{G}$ that contain ELIGIBLE pebbles at step t.

Consider the following dag:

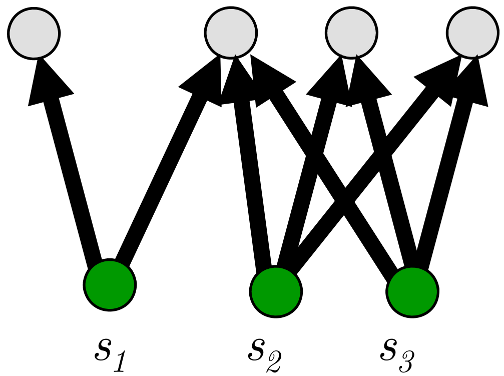
- $\forall$ schedule Σ $E_{\Sigma}(0) = 3$;
- $\max_{\Sigma} E_{\Sigma}(1)=E_{\Sigma'}(1)=3$ (where $\Sigma'=(s_1,s_2,s_3)$ or (s_1,s_3,s_2))
- but $E_{\Sigma'}(2)=2$
- However, $\max_{\Sigma} E_{\Sigma}(2)=E_{\Sigma''}(2)=3$ (where $\Sigma''=(s_2,s_3,s_1)$ or (s_3,s_2,s_1))

No schedule maximal at step 1 is maximal at step 2.

ELIGIBLE

EXECUTED

unpebbled



IC-Optimal Schedules for Common *Dags*

- *Theorem.* [MRY06]
  A schedule for $\left\{ \begin{array}{l} \textit{an evolving mesh-dag (2- or 3-D)} \\ \textit{a reduction-mesh} \\ \textit{a reduction-tree} \\ \textit{an FFT dag} \end{array} \right\}$

  is IC optimal iff is parent oriented.

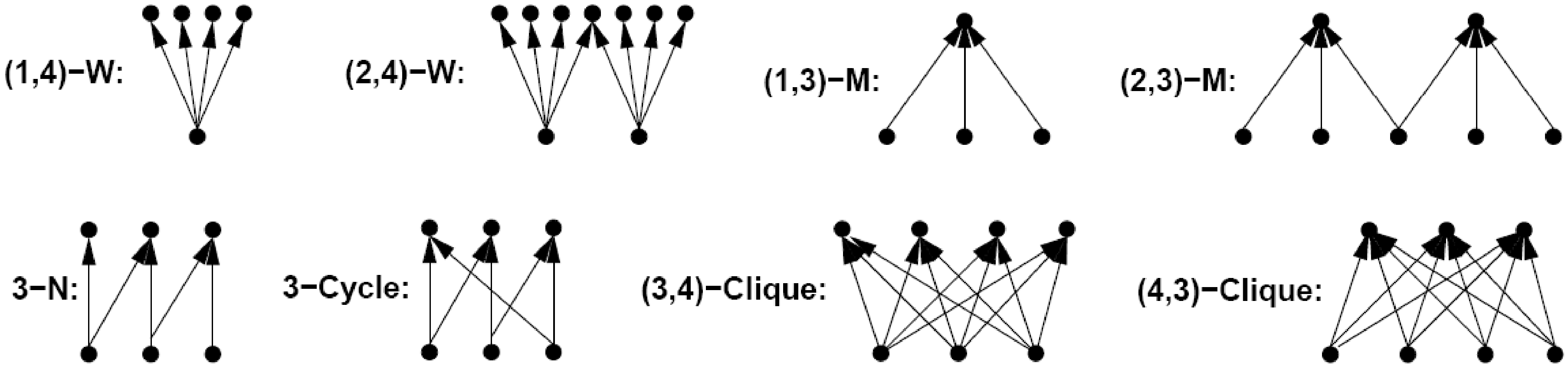
Parent oriented: A node's parents are executed sequentially.

- Meshes: level by level, sequentially across each level
- Tree-dags: in "sibling pairs" (nodes that share a child)
- FFT-dags: in "butterfly pairs" (nodes that share two children)

Decomposition-Based Scheduling Theory

Construct *Dags* from schedulable "building blocks"

1. Choose *bipartite* "building block" *dags* that have optimal schedules.



<u>*Theorem.*</u> [MRY06]

Any schedule for these building blocks that executes all sources sequentially is IC optimal

The Priority relation

Let two *dag* $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively admit an IC optimal schedule Σ_1 and Σ_2 then

$\mathcal{G}_1 \triangleright \mathcal{G}_2$ means that the schedule Σ that entirely execute $\mathcal{G}_1$'s *non-sinks* and then entirely execute $\mathcal{G}_2$'s *non-sinks* is at least as good as any other schedule that execute both $\mathcal{G}_1$ and $\mathcal{G}_2$ ($\mathcal{G}_1+\mathcal{G}_2$).

<u>*Lemma.*</u> [MRY06] *The relation* $\triangleright$ *is transitive*

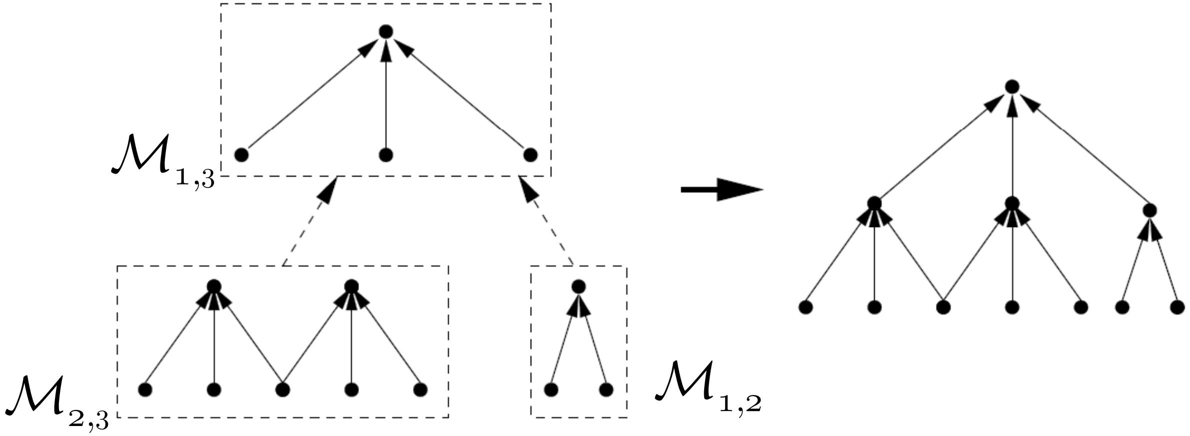
Dag Composition

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ two *dags*, the composition of $\mathcal{G}_1$ and $\mathcal{G}_2$ is obtained by merging some k *sources* of $\mathcal{G}_2$ with some k *sinks* of $\mathcal{G}_1$.

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

Composition is associative

Example $\mathcal{M}_{1,2} \Uparrow \mathcal{M}_{2,3} \Uparrow \mathcal{M}_{1,3}$



Dag Composition

<u>*Theorem.*</u> [MRY06] *If the dag* $\mathcal{G}$ *is a composition of connected bipartite dags* $\{\mathcal{G}_{\mathrm{i}}\}_{1 \le i \le n}$, *of type*

$$\mathcal{G}_1 \Uparrow \mathcal{G}_2 \Uparrow \ldots \Uparrow \mathcal{G}_n$$

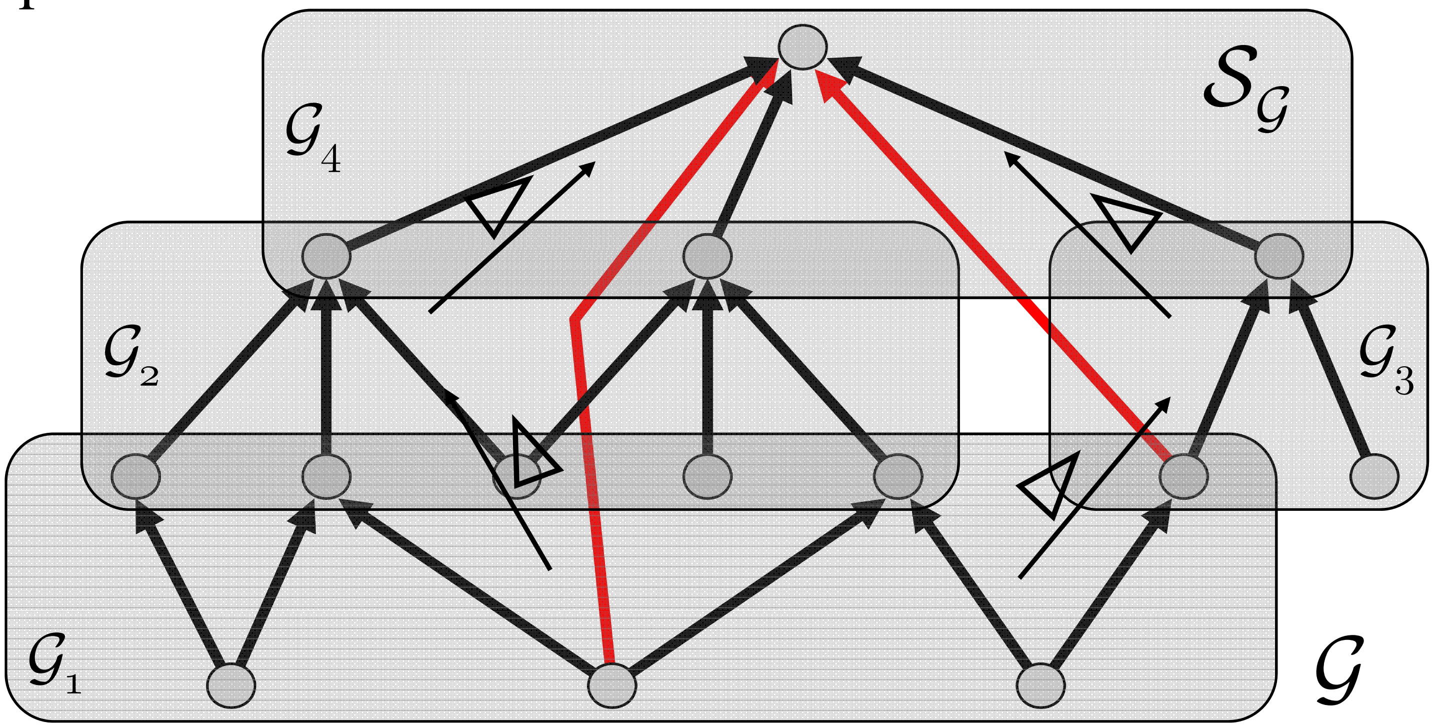
and if

$$\mathcal{G}_1 \rhd \mathcal{G}_2 \rhd \ldots \rhd \mathcal{G}_n \ (\mathcal{G} \textit{ is a } \rhd\text{–linear composition})$$

then executing $\mathcal{G}$ *by executing the* $\mathcal{G}_i$ *in* $\rhd$*–order is IC optimal.*

IC-Optimality via Dag-Decomposition

- The "real" problem is not to *build* a computation-*dag* but rather to execute a given one
- In [MRY06] a framework which allows to convert a "real" *dag* into a *simplified* and *decomposed* one has been provided.



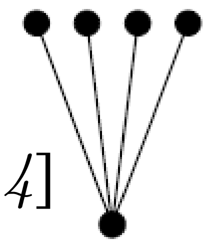
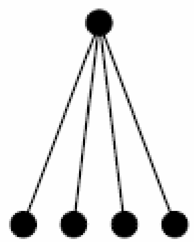
Expanding the repertoire of building-block *dags*

Planar Bipartite Trees (PBT)

A sequence of positive numbers is zigzagged if it alternates integers and reciprocals of integers, with all integers exceeding 1. For any zigzagged sequence δ , the δ-Planar Bipartite Tree (PBT, for short), denoted $\mathcal{P}[\delta]$, is denoted inductively as follows:

For each $\mathrm{d} > 1$:

- $\mathcal{P}[\mathrm{d}]$ is the (single-source) out-degree-d W-dag $\mathcal{W}[\mathrm{d}]$, i.e., the bipartite dag that has one source, d sinks, and d arcs connecting the source to each sink.
- $\mathcal{P}[1/\mathrm{d}]$ is the (single-sink) in-degree-d M-dag $\mathcal{M}[\mathrm{d}]$, i.e., the bipartite dag that has one sink, d sources, and d arcs connecting each source to the sink.

$\mathcal{P}[4] = \mathcal{W}[4]$  $\mathcal{P}[1/4] = \mathcal{M}[4]$ 

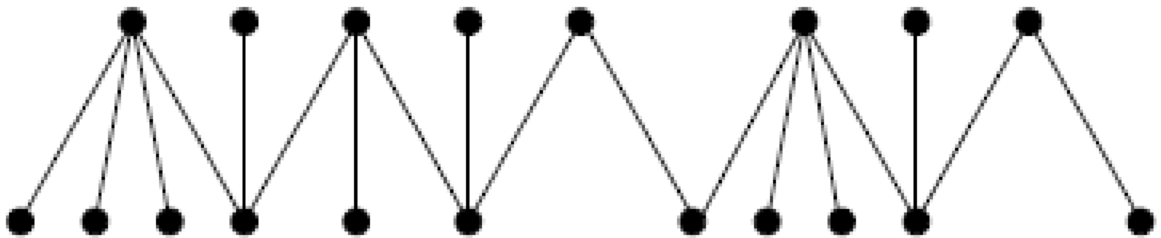
Expanding the repertoire of building-block *dags*

Planar Bipartite Trees (PBT)

For each zigzagged sequence d and each $d > 1$:

If δ ends with a reciprocal, then $\mathcal{P}[\delta, d]$ is obtained by giving d new sinks to $\mathcal{P}[\delta]$, with $\mathcal{P}[\delta]$'s rightmost source as their common parent.

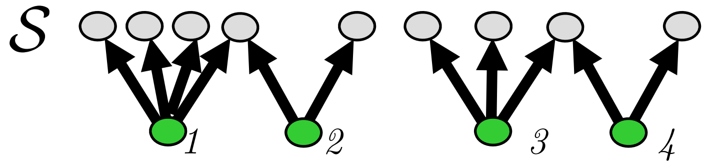
If δ ends with an integer, then $\mathcal{P}[\delta, 1/d]$ is obtained by giving d new sources to $\mathcal{P}[\delta]$, with $\mathcal{P}[\delta]$'s rightmost sink as their common child.



$\mathcal{P}[1/4, 3, 1/3, 3, 1/2, 2, 1/4, 3, 1/2]$

On Scheduling Strands IC-Optimally

<u>Theorem.</u> Every sum of PBT-Strands *admits an IC-optimal schedule.*



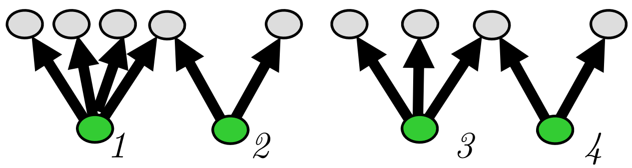
- Let $\mathrm{Src}(\mathcal{S})$ denote $\mathcal{S}$'s sources
- For X $\in$ $\mathrm{Src}(\mathcal{S})$, $\mathrm{e}(\mathrm{X};\mathcal{S})$ denotes the number of sinks of $\mathcal{S}$ that are rendered ELIGIBLE when precisely the sources in X are EXECUTED.
- For each $u \in \mathrm{Src}(\mathcal{S})$
  - For any $k \in [1,n]$, $u(k) = \mathrm{e}(\{u,\ldots, u + k - 1\}; \mathcal{S})$
  - $\mathrm{V}_u = \langle u(1),\ldots,u(n)\rangle$
- Order the vectors $\mathrm{V}_1,\ldots,\mathrm{V}_n$ lexicographically, using the notation $\mathrm{V}_a \geq_{\mathrm{L}} \mathrm{V}_b$ to denote this order.
- A source $s \in \mathrm{Src}(\mathcal{S})$ is *maximum* if $\mathrm{V}_s \geq_{\mathrm{L}} \mathrm{V}_{s'}$ for all $s' \in \mathrm{Src}(\mathcal{S})$.

The greedy schedule $\Sigma_{\mathcal{S}}$

The greedy schedule $\Sigma_{\mathcal{S}}$ for $\mathcal{S}$ operates as follows.

1. $\Sigma_{\mathcal{S}}$ executes any *maximum* $s \in \mathrm{Src}(\mathcal{S})$.
2. $\Sigma_{\mathcal{S}}$ removes from $\mathcal{S}$ the just-executed source s and all sinks having s as their only parent. This converts $\mathcal{S}$ to the sum of PBT-strands $\mathcal{S}$'
3. $\Sigma_{\mathcal{S}}$ recursively executes $\mathcal{S}$' using schedule $\Sigma_{\mathcal{S}'}$.

The schedule $\Sigma_{\mathcal{S}}$ is IC optimal for $\mathcal{S}$

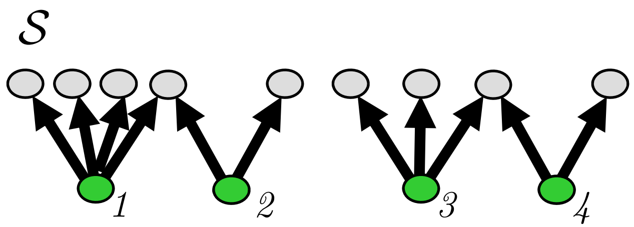


The greedy schedule $\Sigma_{\mathcal{S}}$: An example

Consider the dag $\mathcal{S}=\mathcal{P}[4,1/2,2]+\mathcal{P}[3,1/2,2]$

$V_1=\langle 3,5,5,5\rangle$, $V_2=\langle 1,1,1,1\rangle$, $V_3=\langle 2,4,4,4\rangle$, $V_4=\langle 1,1,1,1\rangle$, hence 1 is *maximum*

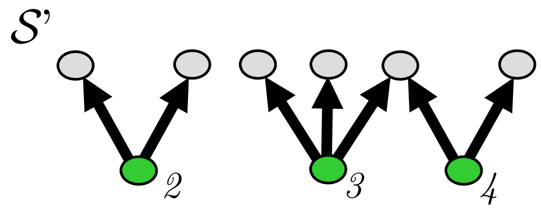
$\Sigma_{\mathcal{S}}$ executes 1



$\mathcal{S}' = \mathcal{P}[2]+P[3,1/2,2]$

$V_2=\langle 2,2,2\rangle$, $V_3=\langle 2,4,4\rangle$, $V_4=\langle 1,1,1\rangle$, hence 3 is *maximum*

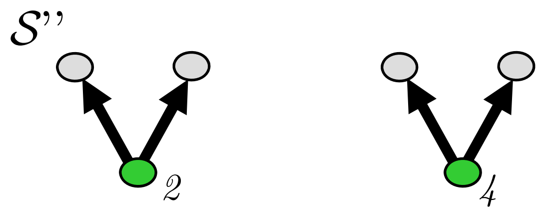
$\Sigma_{\mathcal{S}'}$ executes 3



$\mathcal{S}'' = \mathcal{P}[2]+\mathcal{P}[2]$

$V_2=\langle 2,2\rangle$, $V_4=\langle 2,2\rangle$, hence 2 and 4 are *maximum*

$\Sigma_{\mathcal{S}''}$ executes 2 and then 4 *(or viceversa)*



Scheduling-Based Duality

The dual of a dag $\mathcal{G}$ is the dag $\mathcal{G}^{\mathcal{D}}$ that is obtained by reversing all of $\mathcal{G}$'s arcs

The following results holds:

1. For any dag $\mathcal{G}$, given an optimal schedule Σ for $\mathcal{G}$, one can algorithmically derive from Σ an optimal schedule $\Sigma^{\mathcal{D}}$ for $\mathcal{G}^{\mathcal{D}}$
2. Given two dags, $\mathcal{G}_1$ and $\mathcal{G}_2$, if $\mathcal{G}_1 \triangleright \mathcal{G}_2$, then $\mathcal{G}_2^{\mathcal{D}} \triangleright \mathcal{G}_1^{\mathcal{D}}$

Scheduling-Based Duality

Let $\mathcal{G}$ be a dag having n non-sinks and m non-source nodes.

Let Σ be a schedule for $\mathcal{G}$ that execute its non-sink in the order $u_1, u_2, \ldots, u_n$.

Σ renders $\mathcal{G}$'s sinks ELIGIBLE in a sequence of "packets" $P_1, P_2, \ldots, P_n$ (where P_i is the set of non-sources that become eligible when Σ executes u_i).

A schedule for $\mathcal{G}^{\mathcal{D}}$ *is dual to* Σ if it executes $\mathcal{G}^{\mathcal{D}}$'s sources in an order of the form $[[P_n]], [[P_{n-1}]], \ldots, [[P_1]]$

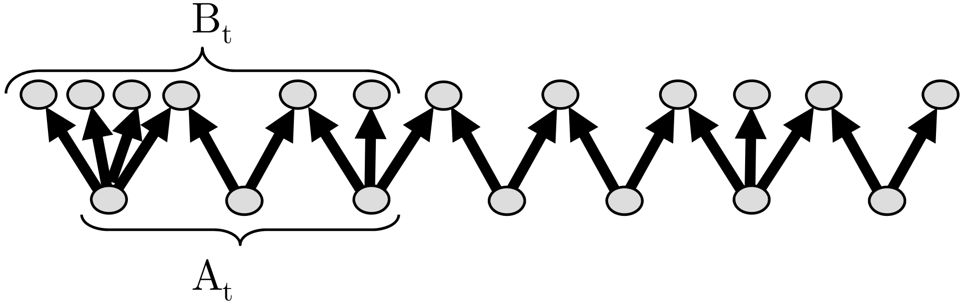
Where $[[a, b, .. c]]$ denotes a fixed, but unspecified permutation of $\{a, b, \ldots, c\}$

Scheduling-Based Duality

Theorem.

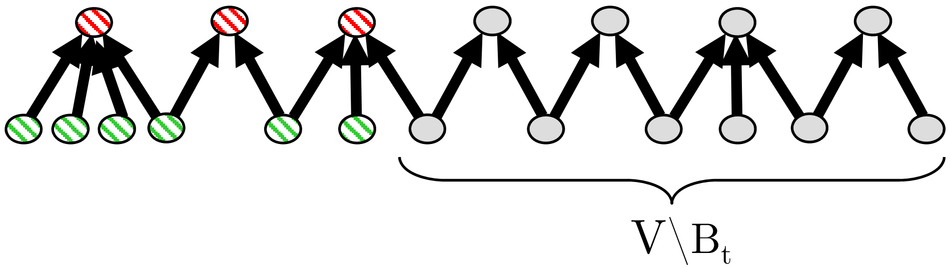
Let the $\mathcal{G}$ dag admit the IC-optimal schedule Σ. Any schedule for $\mathcal{G}^{\mathcal{D}}$ that is dual to Σ *is IC-optimal*

$\mathcal{G}$



A_t = set of sources executed in the first t steps of Σ
B_t = set of sinks ELIGIBLE after step t of Σ

$\mathcal{G}^{\mathcal{D}}$



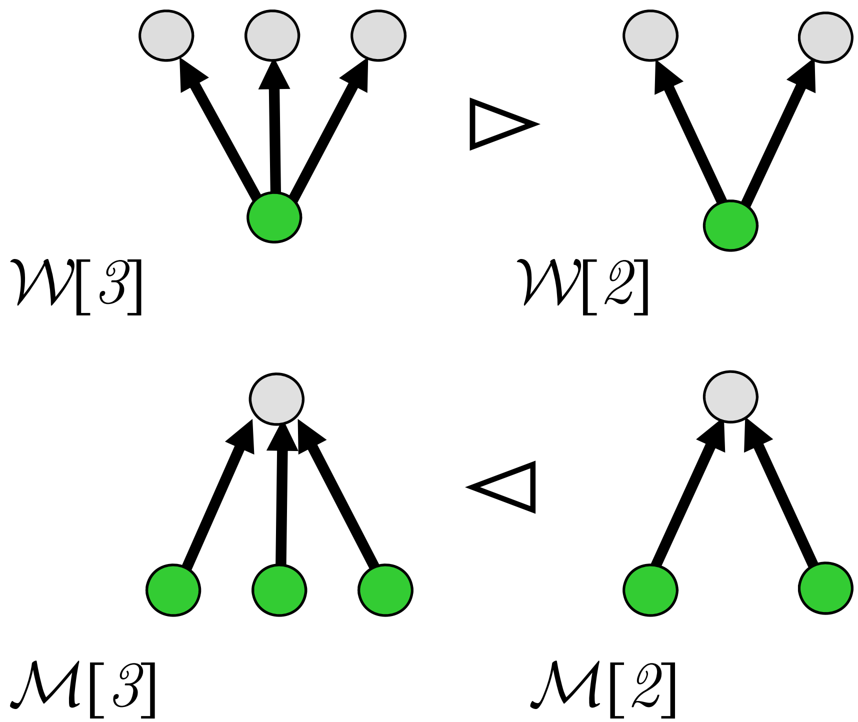
A_t is ICO for $\mathcal{G}$ at step $|A_t|$

$V \backslash B_t$ is ICO for $\mathcal{G}^{\mathcal{D}}$ at step m-$|B_t|$

Scheduling-Based Duality (2)

Theorem. *For any dag* $\mathcal{G}_1$ *and* $\mathcal{G}_2$: $\mathcal{G}_1 \triangleright \mathcal{G}_2$ *if, and only if,* $\mathcal{G}_2^{\mathcal{D}} \triangleright \mathcal{G}_1^{\mathcal{D}}$

An Example



The ICO-Sweep Algorithm

Consider a sequence of $p \geq 2$ disjoint *dags*, $\mathcal{G}_1, \ldots, \mathcal{G}_p$ having respectively $n_1, \ldots, n_p$ non-sinks.

<u>*Lemma.*</u> *If the sum* $\mathcal{G} = \mathcal{G}_1 + \mathcal{G}_2 + \cdots + \mathcal{G}_p$ *admits an IC-optimal schedule* Σ*, then, for each* $i \in [1, p]$,

1. *Each* $\mathcal{G}_i$ *admits an IC-optimal schedule* Σ_i
2. Σ *must execute the non-sinks of* $\mathcal{G}$ *that come from* $\mathcal{G}_i$ *in the same order as some IC-optimal schedule* Σ_i *for* $\mathcal{G}_i$.

Algorithm ICO-Sweep on two *dags*

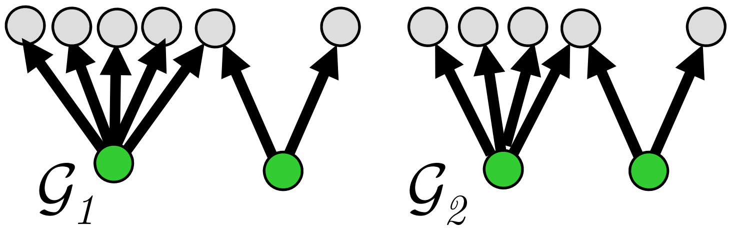
Algorithm **2-ICO-Sweep**:

1. Use the schedules Σ_1 and Σ_2 to construct an $(n_1 + 1) \times (n_2 + 1)$ table $\mathcal{E}$ such that: $(\forall i \in [0, n_1])(\forall j \in [0, n_2])$ $\mathcal{E}(i,j)$ is the maximum number of nodes of $\mathcal{G}_1 + \mathcal{G}_2$ that can be rendered ELIGIBLE by the execution of i non-sinks of $\mathcal{G}_1$ and j non-sinks of $\mathcal{G}_2$.
2. Perform a left-to-right pass along each diagonal $i+j$ of $\mathcal{E}$ in turn, and fill in the $n_1 \times n_2$ *Verification Table* $\mathcal{V}$, as follows.
   a) Initialize all $\mathcal{V}(i,j)$ to "NO"
   b) Set $\mathcal{V}(0,0)$ to "YES"
   c) for each t $\in [1, n_1 + n_2]$:
      i. for each $\mathcal{V}(i,j)$ with $i + j = t$: if ($\mathcal{V}(i\text{-}1, j)$ = "YES" OR $\mathcal{V}(i, j\text{-}1)$ = "YES") AND $\mathcal{E}(i, j) = \max_{a+b=i+j}\{\mathcal{E}(a,b)\}$ then set $\mathcal{V}(i, j)$ to "YES"
      ii. if no entry $\mathcal{V}(i, j)$ with $i + j = t$ has been set to "YES" then HALT and report "There is no IC-optimal schedule."
   d) HALT and report "There is an IC-optimal schedule."

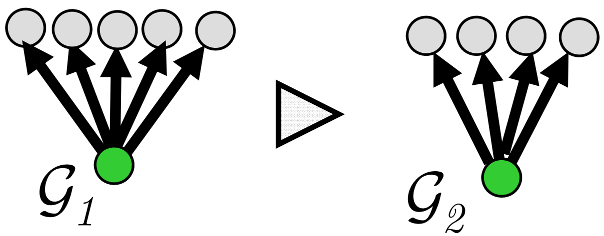
Algorithm ICO-Sweep on two *dags*

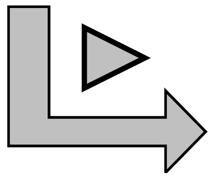
<u>*Theorem.*</u> *Given disjoint* dags, *$\mathcal{G}_1$ and $\mathcal{G}_2$, having, respectively, n_1 and n_2 non-sinks, Algorithm* **2-ICO-Sweep** *determines, within time $O(n_1 n_2)$:*

1. *whether or not the sum $\mathcal{G}_1 + \mathcal{G}_2$ admits an IC-optimal schedule; in the positive case, the Algorithm provides such a schedule;*
2. *whether or not either $\mathcal{G}_1 \triangleright \mathcal{G}_2$, or $\mathcal{G}_2 \triangleright \mathcal{G}_1$, or both.*



$\mathcal{G}_1 \downarrow$ $\mathcal{G}_2 \rightarrow$	0	1	2
0	0	3	5
1	4	7	9
2	6	9	11





$\mathcal{G}_1 \downarrow$ $\mathcal{G}_2 \rightarrow$	0	1
0	0	4
1	5	9

Algorithm ICO-Sweep on multiple *dag*

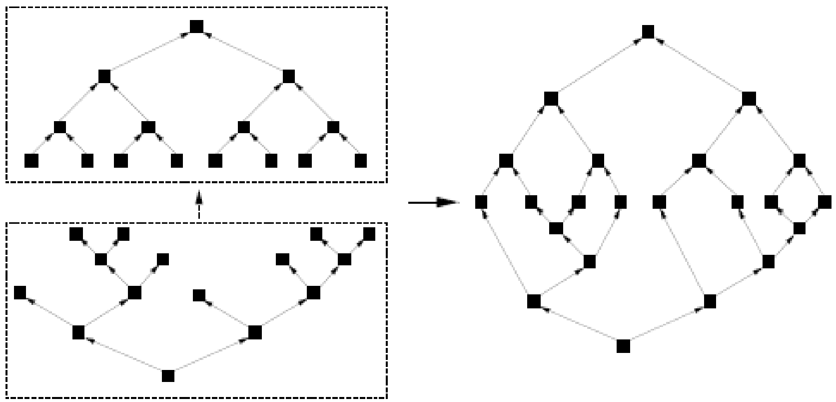
<u>*Theorem.*</u> *Given $p \geq 2$ disjoint* dags*, $\mathcal{G}_1$, $\mathcal{G}_2$,..., $\mathcal{G}_p$, where each $\mathcal{G}_i$ has n_i non-sinks and admits the IC-optimal schedule Σ_i ,*

Algorithm **ICO-Sweep** *determines, within time* $O\left(\Sigma_{1 \leq i < j \leq p} n_i n_j\right)$:

1. *whether or not the sum $\mathcal{G}_1 + \mathcal{G}_2 + \cdots + \mathcal{G}_p$ admits an IC-optimal schedule; in the positive case, the Algorithm provides such a schedule;*
2. *whether or not $\mathcal{G}_1 \triangleright \mathcal{G}_2 \triangleright \cdots \triangleright \mathcal{G}_p$.*

Familiar Computations with ICO Schedules

- Expansive-Reductive Computations



Example: Numerically integrate F via Trapezoid Rule (linear approx to F) or Simpson's Rule (quadratic approx to F)

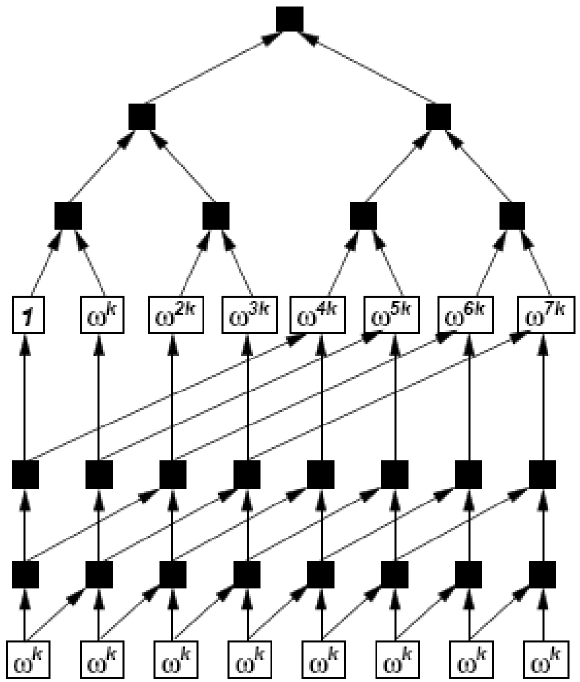
Out-tree: generate approximate area in interval; test for quality accept approximation **OR** bisect interval; recurse

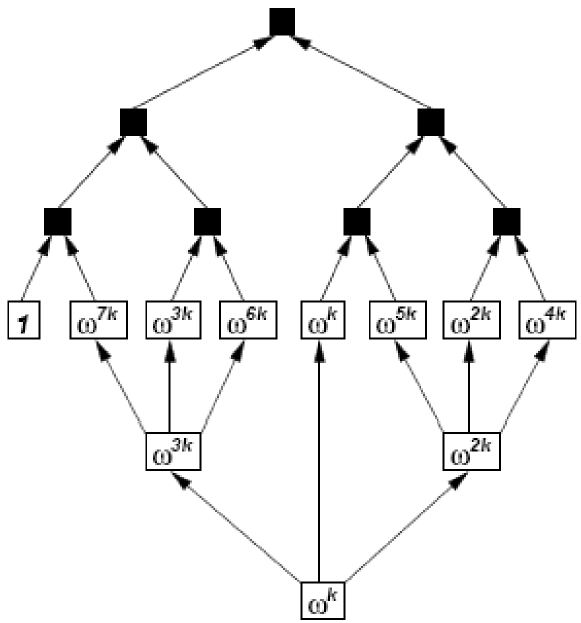
In-tree: Accumulate accepted approximate subareas.

Familiar Computations with ICO Schedules

- The Discrete Laplace Transform: Two Algorithms

kth function: $y_k(\omega) \;=\; x_0 + x_1\omega^k + x_2\omega^{2k} + \cdots + x_{n-1}\omega^{(n-1)k}$

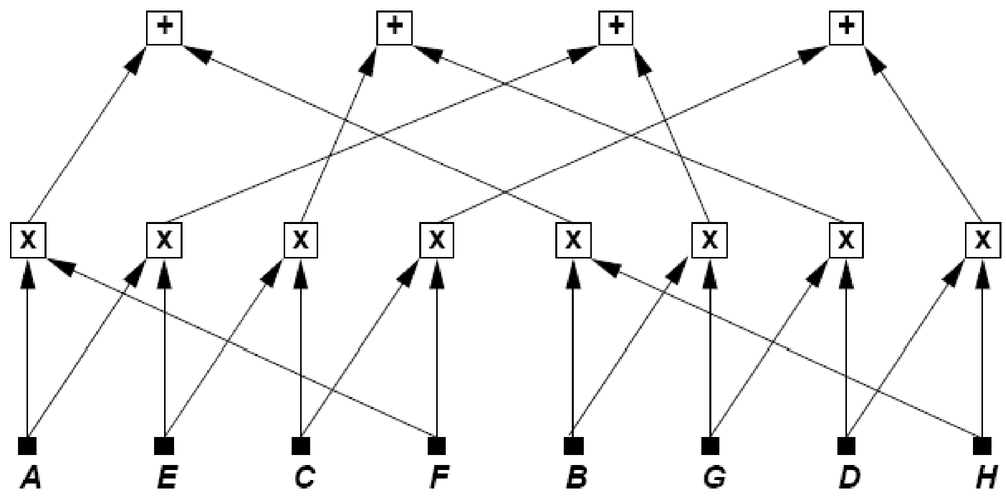




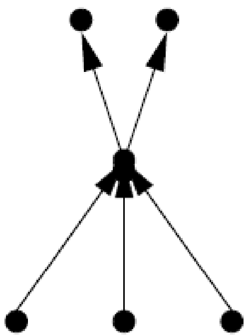
Familiar Computations with ICO Schedules

- Matrix Multiplication via Recursion

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \times \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + BH \end{pmatrix}$$



Project in Progress



1. Extend the priority relation ▷ to include topological order.
   – *Order of composition (rather than ▷-priority) may force a schedule to execute $\mathcal{G}_1$ before $\mathcal{G}_2$.*

2. Determine how to invoke schedules that execute building blocks in an interleaved – rather than sequential – fashion.
   – *Can now do this for bipartite dags.*

3. Experimentally determine the significance of IC-optimality
   – *Initial results suggest significant speedup* [MFRW06]

Thanks for your attention

Any Questions?

References

- [R04] A.L. Rosenberg (2004): "On scheduling mesh-structured computations for Internet-based computing". *IEEE Trans. Comput. 53*, 1176–1186.
- [MRY06] G. Malewicz, A.L. Rosenberg and M. Yurkewych (2006): "Toward a theory for scheduling dags in Internet-based computing". *IEEE Trans. Comput. 55*.
- [MFRW06] Grzegorz Malewicz, Ian Foster, Arnold L. Rosenberg and Michael Wilde (2006): "A Tool for Prioritizing DAGMan Jobs and Its Evaluation". *IEEE International Symposium on High Performance Distributed Computing.*
- [CMR07a] G. Cordasco, G. Malewicz and A.L. Rosenberg (2007): "Advances in IC-Scheduling Theory: Scheduling Expansive and Reductive Dags and Scheduling Dags via Duality". *IEEE Transaction on Parallel and Distributed Systems. (To appear)*
- [CMR07b] G. Cordasco, G. Malewicz and A.L. Rosenberg (2007): "Applying IC-Scheduling Theory to Familiar Classes of Computations". *In Proc. of Workshop on Large-Scale, Volatile Desktop Grids held in conjunction with the IEEE International Parallel & Distributed Processing Symposium, March 30, 2007 Long Beach, California U.S.A. (To appear)*