On Scheduling Expansive and Reductive Dags for Internet-Based Computing

Gennaro Cordasco* Univ. of Salerno

Grzegorz Malewicz† Univ. of Alabama

Arnold L. Rosenberg‡ Univ. of Massachusetts

February 9, 2006

Abstract

Earlier work has developed the underpinnings of a theory of scheduling computations having intertask dependencies—modeled via dags—for Internet-based computing. The goal of the schedules produced is to render tasks eligible for execution at the maximum possible rate. This goal aims: (*a*) to utilize remote clients' computational resources well, by always having work to allocate to an available client; (*b*) to lessen the likelihood of the "gridlock" that ensues when a computation stalls for lack of eligible tasks. The dags handled by the theory thus far are those that can be constructed from a given collection of bipartite *building-block dags* via the operation of *dag-composition*. The current paper extends the range of applicability of the theory by significantly expanding the repertoire of building-block dags that the scheduling algorithms can handle. Thereby, the theory can now schedule large classes of "expansive" and "reductive" dags optimally.

1 Introduction

Earlier work [11, 12, 13] has developed a formal framework for studying the problem of scheduling computations having intertask dependencies for the several modalities of Internet-based computing (IC, for short)—including Grid computing (cf. [1, 6, 5]), global

*Dipt. di Informatica e Applicazioni, Univ. di Salerno, Baronissi (SA) 84081, ITALY, cordasco@dia.unisa.it

†Dept. of Computer Science, Univ. of Alabama, Tuscaloosa, AL 35487, USA, greg@cs.ua.edu

‡Dept. of Computer Science, Univ. of Massachusetts Amherst, Amherst, MA 01003, USA, rsnbrg@cs.umass.edu

computing (cf. [2]), and Web computing (cf. [9]). The goal is to craft schedules that maximize the rate at which tasks are rendered eligible for allocation to remote clients (hence for execution), with the dual aim of: (a) enhancing the effective utilization of remote clients, by always having work to allocate to an available client; (b) lessening the likelihood of the "gridlock" that can arise when a computation stalls pending computation of already-allocated tasks.

The framework of [11, 12, 13] idealizes the problem of scheduling computation-dags[1] for IC, via the assumption that tasks are executed in the order of their allocation. (This assumption idealizes the hope that the monitoring of clients' past behaviors and current capabilities prescribed in, say, [1, 8, 14] can render the desired order likely, if not certain.) Building on the case studies of [12, 13], in [11] we developed what we hope will be the underpinnings of a theory of scheduling computation-dags for IC. The development in [11] begins with any collection of *building-block dags* that we know how to schedule optimally. It develops two conceptual/algorithmic notions that allow us to schedule complex computation-dags built from these building blocks.

1. *The* priority *relation* $\triangleright$ *on pairs of dags.* The assertion "$\mathcal{G}_1 \triangleright \mathcal{G}_2$" asserts that the schedule Σ that entirely executes $\mathcal{G}_1$ and then entirely executes $\mathcal{G}_2$ is at least as good (relative to our quality metric) as any other schedule that executes both $\mathcal{G}_1$ and $\mathcal{G}_2$.

2. *The operation of* composition *on pairs of dags.* If one uses composition to construct a complex computation-dag $\mathcal{G}$ from a set of building blocks that are pairwise comparable under the relation $\triangleright$, then we can often compute an optimal schedule for $\mathcal{G}$ from optimal schedules for the building blocks.

Fig. 1 depicts four familiar dags that yield to the algorithmic framework of [11].

The current paper extends the range of dags that yield to the framework of [11], by significantly expanding the repertoire of building-block dags that we know how to schedule optimally and $\triangleright$-prioritize. We focus here on building blocks that compose to yield "expansive" dags and "reductive" dags, as exemplified in Fig. 1, in addition to compositions of "expansive" dags with "reductive" dags, as exemplified in Fig. 2.

A roadmap. We present the formal framework of our study in Section 2, defining all technical terms and presenting required details from [11]. We introduce the building-block dags of interest in Section 3. In addition to the "expansive" and "reductive" building blocks that are our primary focus, we introduce the class of "T-dags" that generalize both of the preceding classes. In Section 4, we show how to schedule an arbitrary "T-dag" building block optimally; by specialization, we thereby show how to schedule an arbitrary "expansive" or "reductive" building block optimally. The framework of [11] then allows

[1] All technical terms are defined in Section 2.2.

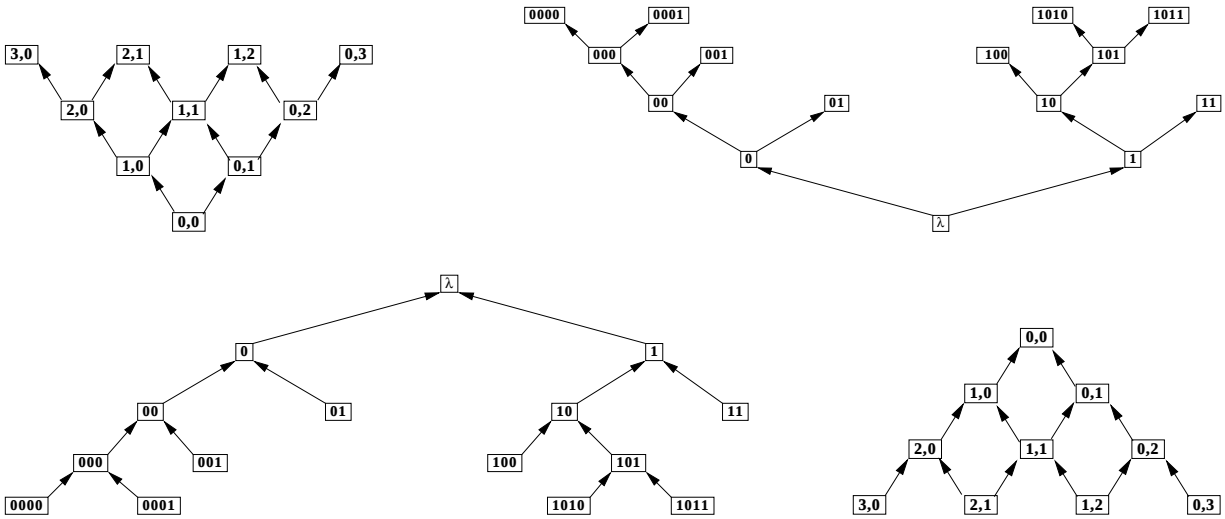

Figure 1: (Top) "expansive" dags: *a 2-dimensional mesh and a binary tree.* (Bottom) "reductive" dags: *a binary tree and a 2-dimensional mesh.*

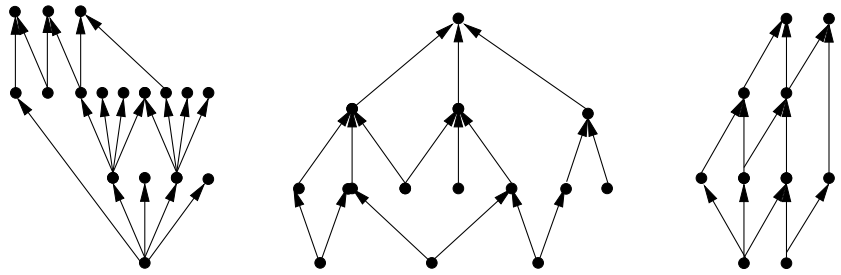

Figure 2: *Three composite dags that the theory of [11] can schedule optimally.*

one to compute, for any "expansive" dag $\mathcal{G}_1$ that admits an optimal schedule Σ_1 and any "reductive" dag $\mathcal{G}_2$ that admits an optimal schedule Σ_2, an optimal schedule for the composition of $\mathcal{G}_1$ with $\mathcal{G}_2$. In Section 5, we prove that "expansive" building blocks and "reductive" building blocks are *dual* to one another in two strong senses. (*a*) Given a building block $\mathcal{B}$ that is either "expansive" or "reductive," and given an optimal schedule Σ for $\mathcal{B}$, one can algorithmically derive from Σ an optimal schedule for the building block $\widehat{\mathcal{B}}$ that is obtained by reversing all arcs of $\mathcal{B}$. (*b*) Given two building blocks, $\mathcal{B}_1$ and $\mathcal{B}_2$ that are both either "expansive" or "reductive," if $\mathcal{B}_1 \rhd \mathcal{B}_2$, then $\widehat{\mathcal{B}}_2 \rhd \widehat{\mathcal{B}}_1$.

Related work. Most closely related to our study are its companions in this nascent scheduling theory: [11], whose contributions we have just described; [12, 13], which characterize and specify optimal schedules for uniform dags typified by those in Fig. 1. A companion to these sources, [10]—which is motivated by the fact that many dags do not admit an optimal schedule in the sense of [11]—pursues an orthogonal regimen for schedul-

ing dags for IC, in which a server allocates *batches* of tasks periodically, rather than allocating individual tasks as soon as they become eligible. Optimality is always possible within the batched framework, but achieving it may entail a prohibitively complex computation. In less directly related work, [7] presents a probabilistic approach to the problem of executing tasks on unreliable clients. Finally, the impetus for our study derives from the many exciting systems- and/or application-oriented studies of IC, in sources such as [1, 2, 5, 6, 8, 9, 14].

2 A Basis for a Scheduling Theory

2.1 Computation-Dags

A *directed graph* $\mathcal{G}$ is given by a set of *nodes* $N_{\mathcal{G}}$ and a set of *arcs* (or, *directed edges*) $A_{\mathcal{G}}$, each of the form $(u \to v)$, where $u, v \in N_{\mathcal{G}}$. A *path* in $\mathcal{G}$ is a sequence of arcs that share adjacent endpoints, as in the following path from node u_1 to node u_n: $(u_1 \to u_2)$, $(u_2 \to u_3)$, $\ldots$, $(u_{n-2} \to u_{n-1})$, $(u_{n-1} \to u_n)$. A *dag* (*directed acyclic graph*) $\mathcal{G}$ is a directed graph that has no cycles—so that no path of the preceding form has $u_1 = u_n$. When a dag $\mathcal{G}$ is used to model a computation, i.e., is a *computation-dag:*

- each node $v \in N_{\mathcal{G}}$ represents a task in the computation;
- an arc $(u \to v) \in A_{\mathcal{G}}$ represents the dependence of task v on task u: v cannot be executed until u is.

For any arc $(u \to v) \in A_{\mathcal{G}}$, u is a *parent* of v, and v is a *child* of u in $\mathcal{G}$. The *indegree* (resp., *outdegree*) of node u is its number of parents (resp., children). A parentless node of $\mathcal{G}$ is a *source*, and a childless node is a *sink*. $\mathcal{G}$ is *bipartite* if $N_{\mathcal{G}}$ can be partitioned into sets X and Y such that, for every arc $(u \to v) \in A_{\mathcal{G}}$, $u \in X$ and $v \in Y$. $\mathcal{G}$ is *connected* if, when one ignores the orientation of its arcs, there is a path connecting every pair of distinct nodes.

2.2 A Model for Executing Dags on the Internet

"Pebble games" on dags have yielded elegant formalizations of a variety of problems related to scheduling computation-dags. Such games use tokens, *pebbles*, to model the progress of a computation on a dag: the placement or removal of the various available types of pebbles—which is constrained by the dependencies modeled by the dag's arcs—represents the changing (computational) status of the dag's task-nodes. Our study is based on the *Internet-Computing* (*IC*, for short) *Pebble Game* of [12]. Based on studies of IC in, e.g.,

[1, 8, 14], arguments are presented in [12, 13] (q.v.) that justify the simplified form of the Game that we study here.

A. The rules of the Game. The IC Pebble Game on a dag $\mathcal{G}$ involves one player S, the *Server*, who has access to unlimited supplies of two types of pebbles: ELIGIBLE pebbles, whose presence indicates a task's eligibility for execution, and EXECUTED pebbles, whose presence indicates a task's having been executed. The Game is played as follows.

The IC Pebble Game

- S begins by placing an ELIGIBLE pebble on each unpebbled source of $\mathcal{G}$.
  /*Unexecuted sources are always eligible for execution, having no parents whose prior execution they depend on.*/
- 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.
- S's goal is to allocate nodes in such a way that every node v of $\mathcal{G}$ *eventually* contains an EXECUTED pebble.
  /*This modest goal is necessitated by the possibility that $\mathcal{G}$ is infinite.*/

A *schedule* for the IC Pebble Game is a rule for selecting which ELIGIBLE pebble to execute at each step of a play of the Game. For brevity, we henceforth call a node ELIGIBLE (resp., EXECUTED) when it contains an ELIGIBLE (resp., an EXECUTED) pebble. For uniformity, we henceforth talk about executing nodes rather than tasks.

B. The quality of a play of the Game. Our goal is to play the IC Pebble Game in a way that maximizes the production rate of ELIGIBLE pebbles. When $\mathcal{G}$ is bipartite, it suffices to focus on maximizing the production rate of ELIGIBLE *sinks*. For each step t of a play of the Game on a dag $\mathcal{G}$ under a schedule Σ, let $E_\Sigma(t)$ denote the number of ELIGIBLE pebbles on $\mathcal{G}$'s *nonsource* nodes at step t.

We measure the **IC quality** *of a play of the IC Pebble Game on a dag $\mathcal{G}$ by the size of $E_\Sigma(t)$ at each step t of the play—the bigger, the better. Our goal is an* **IC-optimal** *schedule Σ, in which $E_\Sigma(t)$ is as big as possible for all steps t.*

The significance of IC quality—hence of IC optimality—stems from the following scenarios. (1) Schedules that produce ELIGIBLE nodes more quickly may reduce the chance of the "gridlock" that could occur when remote clients are slow—so that new tasks cannot be allocated pending the return of already allocated ones. (2) If the IC Server receives a batch

of requests for tasks at (roughly) the same time, then having more ELIGIBLE tasks available allows the Server to satisfy more requests.

2.3 A Framework for Crafting IC-Optimal Schedules

Simplifying the search for schedules. We lose no IC quality by executing all sources of a bipartite dag before any sinks.

Lemma 2.1 ([11]). *If a schedule Σ for a dag $\mathcal{G}$ is altered to execute all of $\mathcal{G}$'s nonsinks before any of its sinks, then the IC quality of the resulting schedule is no less than Σ's.*

The priority relation $\triangleright$. For $i = 1, 2$, let the bipartite dag $\mathcal{G}_i$ have s_i sources and admit the IC-optimal schedule Σ_i. If the following inequalities hold:[2]

$$\begin{aligned}&(\forall x \in [0, s_1])\ (\forall y \in [0, s_2]) : \\ &\quad E_{\Sigma_1}(x) + E_{\Sigma_2}(y) \ \leq\ E_{\Sigma_1}(\min\{s_1, x+y\}) + E_{\Sigma_2}(\max\{0, x+y-s_1\}),\end{aligned} \tag{2.1}$$

then $\mathcal{G}_1$ *has priority over* $\mathcal{G}_2$, denoted $\mathcal{G}_1 \triangleright \mathcal{G}_2$. Informally, one never decreases IC quality by executing a source of $\mathcal{G}_1$ whenever possible.

Lemma 2.2. (a) ([11]) *The relation $\triangleright$ is transitive.* **(b)** *One can decide in time proportional to $s_1 s_2$ whether or not $\mathcal{G}_1 \triangleright \mathcal{G}_2$.*

The proof of Lemma 2.2(a) appears in [11]; the proof of Lemma 2.2(b) is immediate from the definition of $\triangleright$.

A framework for scheduling complex dags. The operation of *composition* is defined inductively as follows.

- Start with a set $\mathcal{B}$ of base dags.[3]
- One composes dags $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{B}$—which could be the same dag with nodes renamed to achieve disjointness—to obtain a composite dag $\mathcal{G}$, as follows.
  - Let $\mathcal{G}$ begin as the *sum* (or, disjoint union), $\mathcal{G}_1 + \mathcal{G}_2$, of the dags $\mathcal{G}_1, \mathcal{G}_2$. Rename nodes to ensure that $N_{\mathcal{G}}$ is disjoint from $N_{\mathcal{G}_1}$ and $N_{\mathcal{G}_2}$.
  - Select some set S_1 of sinks from the copy of $\mathcal{G}_1$ in the sum $\mathcal{G}_1 + \mathcal{G}_2$, and an equal-size set S_2 of sources from the copy of $\mathcal{G}_2$ in the sum.

[2] $[a, b]$ denotes the set of integers $\{a, a+1, \ldots, b\}$.

[3] Continuing the practice of [11], our base dags here will be connected bipartite dags.

– Pairwise identify (i.e., merge) the nodes in the sets S_1 and S_2 in some way. The resulting set of nodes is $\mathcal{G}$'s node-set; the induced set of arcs is $\mathcal{G}$'s arc-set.[4]

- Add the dag $\mathcal{G}$ thus obtained to the base set $\mathcal{B}$.

We denote the composition operation by $\Uparrow$ and say that $\mathcal{G}$ is *composite of type* $[\mathcal{G}_1 \Uparrow \mathcal{G}_2]$.

Lemma 2.3 ([11]). *The composition operation is associative. That is, $\mathcal{G}$ is composite of type $[[\mathcal{G}_1 \Uparrow \mathcal{G}_2] \Uparrow \mathcal{G}_3]$ if, and only if, it is composite of type $[\mathcal{G}_1 \Uparrow [\mathcal{G}_2 \Uparrow \mathcal{G}_3]]$.*

The dag $\mathcal{G}$ is a $\triangleright$-*linear composition* of the connected bipartite dags $\mathcal{G}_1, \ldots, \mathcal{G}_n$ if: (a) $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \cdots \Uparrow \mathcal{G}_n$; (b) each $\mathcal{G}_i \triangleright \mathcal{G}_{i+1}$, for all $i \in [1, n-1]$.

Theorem 2.1 ([11]). *Let $\mathcal{G}$ be a $\triangleright$-linear composition of $\mathcal{G}_1, \ldots, \mathcal{G}_n$, where each $\mathcal{G}_i$ admits an IC-optimal schedule Σ_i. The schedule Σ for $\mathcal{G}$ that proceeds as follows is IC optimal.*

1. *For $i = 1, \ldots, n$, in turn, Σ executes the nodes of $\mathcal{G}$ that correspond to sources of $\mathcal{G}_i$, in the order mandated by Σ_i.*
2. *Σ finally executes all sinks of $\mathcal{G}$ in any order.*

One finds in [11] a suite of algorithms that determine whether or not a given computation-dag $\mathcal{G}$ can be decomposed into a set of bipartite building blocks $\mathcal{G}_i$ that satisfy Theorem 2.1.

The nascent scheduling theory of [11] is illustrated there via a small repertoire of bipartite building blocks that lead, via Theorem 2.1, to a rich family of complex dags that we know how to schedule IC optimally (including the dags in Fig. 1). This early success leads to the current challenge: to expand the repertoire of building blocks that the theory can handle.

3 M-Strands, W-Strands, and T-Strands

We strive for an extensive repertoire of bipartite building-block dags: (a) that compose into dags that one might encounter in real computations; (b) that our theory knows how to schedule optimally and $\triangleright$-prioritize. Although our main focus is on dags that are either *expansive*—growing outward from their sources—or *reductive*—growing inward toward their sinks (cf. Figs. 1 and 3), we gain a technical advantage by considering also a building block that is a combination of expansive and reductive. Specifically, by demonstrating how to schedule any such combined building block IC optimally, we demonstrate in one fell swoop how to schedule IC optimally any building block that is either expansive or reductive.

T-dags. A sequence of positive integers is *heavy* if all integers, save perhaps the first, exceed 1. For any heavy sequence $\hat{\delta}$, the $\hat{\delta}$-**strand of T-dags**, denoted $\mathcal{T}[\hat{\delta}]$, is defined inductively as follows.

[4] An arc $(u \to v)$ is *induced* if $\{u, v\} \subseteq N_{\mathcal{G}}$.

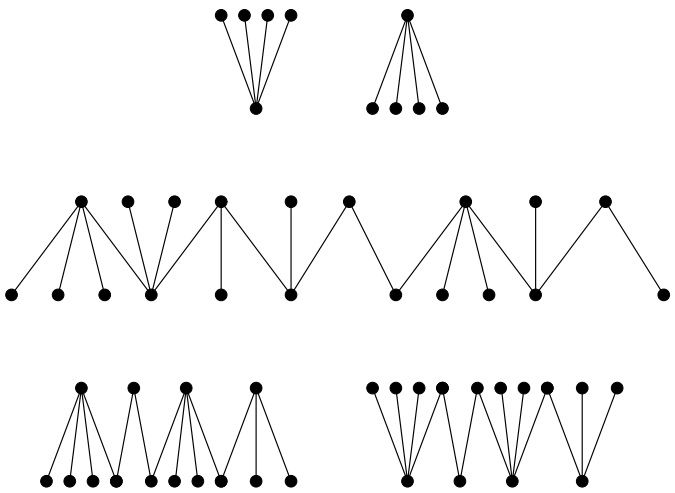

Figure 3: *Top row:* $\mathcal{T}[4] = \mathcal{W}[4]$ *and* $\mathcal{T}[1,4] = \mathcal{M}[4]$*; middle row:* $\mathcal{T}[1,4,4,3,3,2,2,4,3,2]$*; bottom row:* $\mathcal{W}[4,2,4,3]$ *and* $\mathcal{M}[4,2,4,3]$*. All arcs point upward in the figure.*

The base cases. For each $d > 1$:

- $\mathcal{T}[d]$, the (single-source) *outdegree-d* **T-dag**, is the bipartite dag that has one source, d sinks, and d arcs connecting the source to each sink. (Note that $\mathcal{T}[d] = \mathcal{W}[d]$.)
- $\mathcal{T}[1,d]$, the (single-sink) *indegree-d* **T-dag**, is the bipartite dag that has one sink, d sources, and d arcs connecting each source to the sink. (Note that $\mathcal{T}[1,d] = \mathcal{M}[d]$.)

The inductive extensions. For each heavy sequence $\hat{\delta}$ and each $d > 1$, the dag $\mathcal{T}[\hat{\delta}, d]$ is obtained as follows.[5]

If $|\hat{\delta}|$ is even (resp., odd), then identify (or, merge) the rightmost source-to-sink arc of $\mathcal{T}[\hat{\delta}]$ with the leftmost source-to-sink arc of $\mathcal{T}[d]$ (resp., of $\mathcal{T}[1,d]$).

W-dags and M-dags. We now present the two important subclasses of T-dags that are our main focus. For any finite sequence $\hat{\delta}$ of integers, each > 1: the $\hat{\delta}$**-strand of W-dags**, denoted $\mathcal{W}[\hat{\delta}]$, and the $\hat{\delta}$**-strand of M-dags**, denoted $\mathcal{M}[\hat{\delta}]$ are defined inductively as follows.

1. For each integer $d > 1$, $\mathcal{W}[d]$ is the (single-source) *degree-d* **W-dag**. It has one source and d sinks; its d arcs connect the source to each sink.

[5] $|\hat{\delta}|$ denotes the length of, i.e., number of integers in, sequence $\hat{\delta}$.

2. For each sequence $\hat{\delta}$ of integers, each > 1, and each integer $d > 1$, $\mathcal{W}[\hat{\delta}, d]$ is obtained from $\mathcal{W}[\hat{\delta}]$ and $\mathcal{W}[d]$ by identifying (or, merging) the rightmost sink of the former dag with the leftmost sink of the latter.

Clearly, every M-strand and every W-strand is a T-strand:

> Every W-strand $\mathcal{W}[d_1, d_2, \ldots, d_k]$ is equal to $\mathcal{T}[d_1, 2, d_2, 2, \ldots, 2, d_k]$; every M-strand $\mathcal{M}[d_1, d_2, \ldots, d_k]$ is equal to $\mathcal{T}[1, d_1, 2, d_2, 2, \ldots, 2, d_k]$.

Every W-strand $\mathcal{W}[\hat{\delta}]$ has a *dual* **M-strand** $\mathcal{M}_{\mathcal{W}}[\hat{\delta}]$ that is obtained by "reversing" all arcs. Conversely, each M-strand $\mathcal{M}[\hat{\delta}]$ has a *dual* W-strand $\mathcal{W}_{\mathcal{M}}[\hat{\delta}]$ that is obtained by "reversing" all arcs. In Fig. 3, e.g., the W-strands in the top row and the M-strands in the bottom row are (from left to right) dual to one another. We refer generically to $\mathcal{M}[\hat{\delta}]$ as an **M-strand** and to $\mathcal{W}[\hat{\delta}]$ as a **W-strand**. *Note that every strand is, by definition, connected, hence has no isolated nodes.*

Fig. 1 illustrates why we view W-strands and M-strands as the basic building blocks of expansive and reductive computation-dags. The expansive mesh is composite of type $\mathcal{W}[2] \Uparrow \mathcal{W}[2,2] \Uparrow \mathcal{W}[2,2,2]$; the expansive tree is composed of seven instances of $\mathcal{W}[2]$. Dually, the reductive mesh is composite of type $\mathcal{M}[2,2,2] \Uparrow \mathcal{M}[2,2] \Uparrow \mathcal{M}[2]$; the reductive tree is composed of seven instances of $\mathcal{M}[2]$.

The informal duality noted in the preceding paragraph has technical, as well as intuitive significance. The arc "reversal" that dualizes a strand exchanges the roles of source and sink and of indegree and outdegree. One might, therefore, expect that an IC-optimal schedule for $\mathcal{W}_{\mathcal{M}}$ or $\mathcal{M}_{\mathcal{W}}$ can be obtained from an IC-optimal schedule for $\mathcal{M}$ or $\mathcal{W}$, respectively, by "playing the schedule in reverse." The next section shows this to be true.

4 IC-Optimal Schedules for T-Strands

The main result of this section is the following.

Theorem 4.1. *Every sum of T-strands admits an IC-optimal schedule.*

Because M-strands and W-strands are special forms of T-strands, Theorem 4.1 immediately yields the following.

Corollary 4.1. *Every sum of M-strands and every sum of W-strands admits an IC-optimal schedule.*

Our proof of Theorem 4.1 in fact proves a stronger assertion, which we develop now.

Focus on an arbitrary sum of T-strands $\mathcal{S}$. By Lemma 2.1, we may restrict attention to schedules that execute all of $\mathcal{S}$'s sources before any of its sinks. Let $Src(\mathcal{S})$ denote the set of $\mathcal{S}$'s sources. For any $X \subseteq Src(\mathcal{S})$, denote by $e(X;\mathcal{S})$ the number of sinks of $\mathcal{S}$ that are ELIGIBLE at the step when the sources in X are precisely those that have been executed.

Focus on a planar drawing of $\mathcal{S}$, strand by strand, so that we can identify its sources, from left to right, with the integers $1, 2, \ldots, n$, where $n = |Src(\mathcal{S})|$. For any $u \in Src(\mathcal{S})$ and any $k \in [1, n]$, denote by $u[k]$ the set $u[k] = \{u, u+1, \ldots, u+k-1\}$ comprising source u and the $k-1$ sources to its right in the drawing. To simplify exposition, we allow k to exceed $n+1-u$, so that $u[k]$ may contain numbers greater than n—*but none of these large integers denotes a node of* $\mathcal{S}$. Let $\mathcal{S}_u$ denote the strand of $\mathcal{S}$ that u belongs to, and let $S_u = Src(\mathcal{S}_u)$. Denote by $e_k(u)$ the number $e_k(u) = e(u[k] \cap S_u;\ \mathcal{S})$ of sinks of $\mathcal{S}$ that are rendered ELIGIBLE when the sources in $u[k]$ are the only EXECUTED sources of $\mathcal{S}$. Associate with each $u \in Src(\mathcal{S})$ the n-component *eligiblity vector* $V_u = \langle e_1(u), e_2(u), \ldots, e_n(u)\rangle$. Note that, since $|Src(\mathcal{S})| = n$, if $u[k]$ contains numbers larger than n, then the last u entries of V_u are identical. Finally, order the vectors $V_1, V_2, \ldots, V_n$ *in lexicographic order*, using the notation $V_v \geq_L V_w$ to indicate that source v's vector precedes source w's vector lexicographically. We call a source $s \in Src(\mathcal{S})$ *maximal* if $V_s \geq_L V_{s'}$ for all $s' \in Src(\mathcal{S})$.

A schedule $\Sigma_{\mathcal{S}}$ for the sum of T-strands $\mathcal{S}$ is $\geq_L$*-greedy* if it operates as follows.

1. $\Sigma_{\mathcal{S}}$ chooses any maximal $s \in Src(\mathcal{S})$ as the first source of $\mathcal{S}$ to execute.
2. After executing source s, $\Sigma_{\mathcal{S}}$ removes from $\mathcal{S}$ source s and all sinks (if any) that have s as their only parent. This converts $\mathcal{S}$ to a new sum of T-strands $\mathcal{S}'$ whose constituent strands are as follows.
   (a) Each T-strand of $\mathcal{S}$ other than $\mathcal{S}_s$ (i.e., each strand that does not contain s) is a constituent strand of $\mathcal{S}'$;
   (b) $\mathcal{S}_s$ contributes to $\mathcal{S}'$ all of its nodes other than s and all sinks that have s as their only parent.

   The number of constituent strands of $\mathcal{S}'$ is either one smaller than the number of constituent strands of $\mathcal{S}$ (if s is $\mathcal{S}_s$'s only source), or is the same as that number (if s is either the leftmost or rightmost source of $\mathcal{S}_s$), or is one greater than that number (in all other cases).
3. $\Sigma_{\mathcal{S}}$ recursively executes the sum $\mathcal{S}'$ using the $\geq_L$-greedy schedule $\Sigma_{\mathcal{S}'}$.

We prove Theorem 4.1 via the following more detailed result.

Theorem 4.2. *For any sum of T-strands* $\mathcal{S}$*, every* $\geq_L$*-greedy schedule* $\Sigma_{\mathcal{S}}$ *is IC optimal.*

(Intuitively the "lookahead" inherent in the eligiblity vectors prevents such an $\Sigma_{\mathcal{S}}$ from "getting stuck in" local optima that are not globally optimal.)

We prove Theorem 4.2 by induction, in the next two subsections.

4.1 $\Sigma_{\mathcal{S}}$ Chooses the Best First Source to Execute

We show in this section that any $\geq_L$-greedy schedule starts out by executing a source of $\mathcal{S}$ that is most advantageous (with respect to IC quality).

Lemma 4.1. *Let $\mathcal{S}$ be an arbitrary sum of T-strands, and let s be an arbitrary maximal source of $\mathcal{S}$. For each set $X \subseteq Src(\mathcal{S})$, there is a set $X' \subseteq Src(\mathcal{S})$ of equal cardinality such that: (a) $s \in X'$, and (b) $e(X'; \mathcal{S}) \geq e(X; \mathcal{S})$.*

Proof. For any maximal source s of $\mathcal{S}$, focus on an arbitrary $X \subseteq Src(\mathcal{S})$ that does not contain s. Let $w \in X$ be any source and $\ell \in [1, n]$ any integer such that $w[\ell]$ is a maximal cardinality sequence of consecutive numbers from $X \cap S_w$ (using the integer names of sources). The maximality of source s ensures that $V_s \geq_L V_w$; i.e., either $V_s = V_w$, or there exists $h \in [0, n-1]$ such that $s(1) = w(1)$, $s(2) = w(2), \ldots,$ $s(h) = w(h)$, and $s(h+1) > w(h+1)$.

We now investigate the impact on the number of ELIGIBLE sinks at step $|X|$ of the execution of $\mathcal{S}$, of "un-executing" certain sources from set X and executing an equal number of other sources in their stead. Specifically, we "un-execute" some EXECUTED sources near to, and including w, and execute an equal number of un-EXECUTED sources near to, and including s. (This substitution for certain nodes in X will yield the set X'.)

Focus on the following two quantities.

- $h^\star \in [0, n-1]$ is the largest value such that $e_1(s) = e_1(w), e_2(s) = e_2(w), \ldots, e_{h^\star}(s) = e_{h^\star}(w)$. It is, thus, the maximum number of consecutive sources starting from s whose execution *does not* improve on the "yield" produced by starting from w.
- $k^\star \in [1, \ell]$ is the largest value such that $s[k^\star] \cap X = \emptyset$. It is, thus, the maximum number of consecutive sources starting from s that are *not* EXECUTED at the start of our modification of X.

We branch on the relative sizes of $h^\star$ and $k^\star$.

Case 1: $h^\star < k^\star$.
Consider the impact of "unexecuting" the sources in $w[h^\star + 1]$ and executing, in their steads, the sources in $s[h^\star + 1]$ (which, in this case, is a subset of S_s). This replaces the set X by $X' = X \setminus w[h^\star + 1] \cup s[h^\star + 1]$. The "unexecution" loses us at most $e_{h^\star+1}(w) + 1$

ELIGIBLE sinks. (The possible extra sink may be lost if the rightmost source in $w[h^\star + 1]$ has a righthand neighbor in $\mathcal{S}_w \cap X$ with which it shares a child.) In compensation, the execution gains us at least $e_{h^\star+1}(s) \geq e_{h^\star+1}(w) + 1$ new ELIGIBLE sinks. We are, thus, not worse off for the exchange.

Case 2: $h^\star \geq k^\star$.
We have two subcases to consider.

2a. $e_{k^\star}(w) = e_\ell(w)$.
In this case, we know two significant pieces of information.

1. Once having executed nodes $w, w+1, \ldots, w+k^\star-1$, no further sinks were rendered ELIGIBLE by executing any source in $\{w+k^\star, \ldots, w+\ell-1\}$. (Recall that $w[\ell] \subseteq X$.)

2. All of nodes $s, s+1, \ldots, s+k^\star-1$ belong to S_s, and none belongs to X.

This information tells us that "unexecuting" the sources in $w[k^\star]$ loses us *exactly* $e_{k^\star}(w)$ ELIGIBLE sinks, while executing the sources in $s[k^\star]$ gains us at least $s_{k^\star}(w) = e_{k^\star}(w)$ new ELIGIBLE sinks. This means that replacing the set X by $X' = X \setminus w[k^\star] \cup s[k^\star]$ leaves us with at least as many ELIGIBLE sinks. As in Case 1, we are not worse off for the exchange.

2b. $e_{k^\star}(w) < e_\ell(w)$.
In this case, even having executed nodes $w, w+1, \ldots, w+k^\star-1$, some new sinks *are* rendered ELIGIBLE by executing the sources in $\{w+k^\star, \ldots, w+\ell-1\}$. This means that $s+k^\star \in S_s$, i.e., that the integer $s+k^\star$ is the name of a real source of $\mathcal{S}$. To wit, were this not true, we would have $e_{k^\star+1}(s) = e_{k^\star}(s)$ for all $i > 1$, so that, in particular, $e_\ell(s) < e_\ell(w)$. This would contradict the fact that $V_s \geq_L V_w$.

Let t be the leftmost child of $s+k^\star$ (using the planar drawing of $\mathcal{S}$ to specify "leftmost"). Since source $s+k^\star-1 \notin X$, t was not rendered ELIGIBLE by the execution of the sources in X. Further, since $s+k^\star \notin s[k^\star]$, t will not be rendered ELIGIBLE by the execution of the sources in $s[k^\star]$.

We digress to introduce a useful notion. A source (resp., sink) u of a T-strand is a *backbone* source (resp., sink) if u is either extremal in the strand (i.e., leftmost or rightmost), or u has at least two children (resp., at least two parents).

We now note the obvious, yet important, fact that *every maximal source is a backbone source*. To wit, at least one of the two neighboring (in the planar drawing) backbone sources of each non-backbone source s has a lexicographically larger eligibility vector than s.

Our analysis now branches on whether or not $s+k^\star$ is a backbone source of $\mathcal{S}$.

2b(i). If $s+k^\star$ is a backbone source, then we know four significant facts.

1.	$s + k^\star \in X$	because $h^\star \geq k^\star$;
2.	$s + k^\star$ is a backbone source	by assumption;
3.	s is a backbone source	because it is maximal;
4.	$s + k^\star \notin w[k^\star]$	because $V_s \geq_L V_w$.

It follows that executing the nodes in $X \cup s[k^*]$ renders sink t ELIGIBLE. Therefore, "unexecuting" the sources in $w[k^\star]$ loses us at most $e_{k^\star}(w) + 1$ ELIGIBLE sinks. In compensation, executing the sources in $s[k^\star]$ gains us at least $e_{k^\star}(s)+1$ ELIGIBLE sinks—because t becomes ELIGIBLE when source $s+k^\star-1$ is executed. Thus, replacing X with $X' = X \setminus w[k^\star] \cup s[k^\star]$ leaves us with at least as many ELIGIBLE sinks.

2b(ii). If $s + k^\star$ is not a backbone source, then in order to render sink t ELIGIBLE, we must execute not only $s + k^\star$, but also the set $R \subset S_s$ that comprises all of the sources to $s + k^\star$'s right (in the drawing), until we reach the next backbone source.

Let $j^\star \in [0, n]$ be the largest integer such that $e_{k^\star}(w) = e_{k^\star+j^\star}(w)$; i.e., $e_{k^\star}(w) = e_{k^\star+1}(w) = e_{k^\star+2}(w) = e_{k^\star+j^\star}(w) < e_{k^\star+j^\star+1}(w)$. Since the current case is defined by the condition "$e_{k^\star}(w) < e_\ell(w)$," we know that *every element of* $w[k^\star + j^\star]$ *belongs to* X.

Easily, $|R| \leq j^\star$. Were this not the case, then we would have $e_{k^\star+j^\star+1}(s) = e_{k^\star+j^\star}(s) = \cdots = e_{k^\star}(s) = e_{k^\star}(w) < e_{k^\star+j^\star+1}(w)$. This would contradict the fact that $V_s \geq_L V_w$.

Note now that "unexecuting" the $k^\star + j^\star$ sources in $w[k^\star + j^\star]$ would lose us no more than $e_{k^\star}(w) + 1$ ELIGIBLE sinks. In compensation, executing the $k^\star$ sources in $s[k^\star]$, plus the at-most $j^\star$ sources in R gains us at least $e_{k^\star}(s) + 1$ ELIGIBLE sinks (because sink t becomes ELIGIBLE). Thus, replacing the set X with the (possibly smaller) set $X' = X \setminus w[k^\star + j^\star] \cup (s[k^\star] \cup R)$ leaves us with at least as many ELIGIBLE sinks.

The sequences of cases we have considered have established the following. By executing a sequence of sources starting with s, instead of a like-numbered sequence starting with w, we can only increase the total number of ELIGIBLE sinks. The lemma follows. □

4.2 $\Sigma_{\mathcal{S}}$ Continues to Make Good Choices

Assume, for induction, that every $\geq_L$-greedy schedule in IC optimal for all sums of T-strands having n or fewer sources. Lemma 4.1 verifies the $(n = 1)$ case of this assertion. Focus, therefore, on an arbitrary sums of T-strands having $n+1$ sources. Lemma 4.1 shows that every $\geq_L$-greedy schedule, $\Sigma_{\mathcal{S}}$, chooses an optimal source $s \in Src(\mathcal{S})$ to execute in its first step. Hence, if we seek a set $X \subseteq Src(\mathcal{S})$ that renders maximally many sinks of $\mathcal{S}$ ELIGIBLE, among cardinality-$|X|$ subsets of $Src(\mathcal{S})$, we can, with no loss of generality, choose X to contain s. Let $\mathcal{S}'$ be the sum of T-strands obtained by removing from $\mathcal{S}$ source s plus all sinks that have s as their only parent.

Let the *yield*, $Yld(v)$, of a source v be the number of sinks that are rendered ELIGIBLE by executing just v. Removing s and its single-parent children effectively increases by 1 the yields of the sources that neighbor s in $\mathcal{S}_s$, if any.

Easily $e(X;\mathcal{S}) = Yld(s)+e(X\setminus\{s\}\,;\,\mathcal{S}')$. Since $\mathcal{S}'$ has fewer sources than $\mathcal{S}$, our inductive assumption asserts that schedule $\Sigma_{\mathcal{S}'}$ is IC optimal for $\mathcal{S}'$. It follows that schedule $\Sigma_{\mathcal{S}}$, which executes s and then mimics schedule $\Sigma_{\mathcal{S}'}$, is IC optimal for $\mathcal{S}$.

This completes the proof of Theorem 4.2, hence, also, of Theorem 4.1.

5 Dualities between W-Strands and M-Strands

In this section, we demonstrate that W-strands and M-strands are *dual* to one another, in the sense that knowing how to schedule and/or $\triangleright$-prioritize either class of dags allows one to schedule (Section 5.1) and/or $\triangleright$-prioritize (Section 5.2) the other. While the scheduling-based duality is not really needed, given the algorithm in Section 4 that schedules every T-dag IC optimally, the duality is convenient to know about, because it allows one to "read off" an IC-optimal schedule for either a W-strand or an M-strand from an an IC-optimal schedule for its dual.

5.1 Scheduling-Based Duality of W-strands and M-strands

5.1.1 Scheduling M-strands via their dual W-strands

Let $\mathcal{W}$ be a W-strand that has n sources, $\{u_1,\ldots,u_n\}$, and m sinks, $\{v_1,\ldots,v_m\}$. Let Σ be a schedule for $\mathcal{W}$ that executes $\mathcal{W}$'s sources in the order

$$u_{k_1},\ u_{k_2},\ \ldots,\ u_{k_{n-1}},\ u_{k_n}. \tag{5.1}$$

Easily, Σ renders $\mathcal{W}$'s sinks ELIGIBLE in a sequence of "packets,"

$$P_1 = \{v_{1,1},\ldots,v_{1,i_1}\},\ P_2 = \{v_{2,1},\ldots,v_{2,i_2}\},\ \ldots,\ P_n = \{v_{n,1},\ldots,v_{n,i_n}\}; \tag{5.2}$$

for each $\ell \in [1,n]$, the set P_ℓ is the (possibly empty) "packet" of sinks that become ELIGIBLE when Σ executes source u_{k_ℓ}. (Clearly, there must be an arc from u_{k_ℓ} to each sink in P_ℓ.) A schedule for $\mathcal{M}_{\mathcal{W}}$ is *dual to* Σ if it executes $\mathcal{M}_{\mathcal{W}}$'s sources in an order of the form

$$[[v_{n,1},\ldots,v_{n,i_n}]],\ [[v_{n-1,1},\ldots,v_{n-1,i_{n-1}}]],\ \ldots,\ [[v_{1,1},\ldots,v_{1,i_1}]], \tag{5.3}$$

where $[[a,b,\ldots,c]]$ denotes a fixed, but unspecified, permutation of the set $\{a,b,\ldots,c\}$. Note that $\mathcal{M}_{\mathcal{W}}$ will generally admit many schedules that are dual to Σ.

Theorem 5.1. *Let the W-strand $\mathcal{W}$ admit the IC-optimal schedule $\Sigma_{\mathcal{W}}$. Any schedule for $\mathcal{M}_{\mathcal{W}}$ that is dual to $\Sigma_{\mathcal{W}}$ is IC optimal.*

Proof. Let $\Sigma_{\mathcal{W}}$ execute $\mathcal{W}$'s sources in the order (5.1) and render $\mathcal{W}$'s sinks ELIGIBLE according to the partial order (5.2). It follows that every schedule for $\mathcal{M}_{\mathcal{W}}$ that is dual to $\Sigma_{\mathcal{W}}$ executes $\mathcal{M}_{\mathcal{W}}$'s m sources in an order of the form (5.3). Assume, for contradiction, that some schedule, $\Sigma_{\mathcal{M}_{\mathcal{W}}}$, for $\mathcal{M}_{\mathcal{W}}$ is dual to $\Sigma_{\mathcal{W}}$ but is *not* IC optimal. There is then a schedule Σ for $\mathcal{M}_{\mathcal{W}}$ and a step $t \in [1, m]$ such that $E_{\Sigma}(t) > E_{\Sigma_{\mathcal{M}_{\mathcal{W}}}}(t)$. We show now that the existence of Σ refutes the alleged IC optimality of $\Sigma_{\mathcal{W}}$.

Some notation will facilitate our analysis. Let A_t denote the sources of $\mathcal{M}_{\mathcal{W}}$ that $\Sigma_{\mathcal{M}_{\mathcal{W}}}$ executes during its first t steps, and let B_t denote the sinks of $\mathcal{M}_{\mathcal{W}}$ that these t executions render ELIGIBLE. Let A'_t and B'_t denote, respectively, the analogous sets for schedule Σ. Note that $|A_t| = |A'_t| = t$, and that, by assumption,

$$b \stackrel{\text{def}}{=} |B_t| \;=\; E_{\Sigma_{\mathcal{M}_{\mathcal{W}}}}(t) \;<\; E_{\Sigma}(t) \;=\; |B'_t| \stackrel{\text{def}}{=} b'.$$

For each set S, we denote by $\overline{S}$ its "complement;" e.g., $\overline{B_t} = U - B_t$ comprises the sinks of $\mathcal{M}_{\mathcal{W}}$ that *do not* become ELIGIBLE when the sources in A_t are executed.

Focus on a schedule $\Sigma'_{\mathcal{W}}$ for $\mathcal{W}$ that executes $\mathcal{W}$'s sources in the order $[[\overline{B'_t}]],\ [[B'_t]]$; i.e.,

1. $\Sigma'_{\mathcal{W}}$ begins by executing, in some order, the $n - b'$ sources of $\mathcal{W}$ in the set $\overline{B'_t}$;
2. $\Sigma'_{\mathcal{W}}$ then executes, in some order, the b' sources of $\mathcal{W}$ in the set B'_t.

Consider the numbers of sinks of $\mathcal{W}$ that become ELIGIBLE at various steps under schedules $\Sigma_{\mathcal{W}}$ and $\Sigma'_{\mathcal{W}}$.

Claim 1. $E_{\Sigma'_{\mathcal{W}}}(n - b') \;\geq\; m - t$.

Proof of Claim 1. No sink of $\mathcal{W}$ in $\overline{A'_t}$ can have a parent in B'_t, for such a dependency would deny that all sinks of $\mathcal{M}_{\mathcal{W}}$ in B'_t are rendered ELIGIBLE by executing the sources of $\mathcal{M}_{\mathcal{W}}$ in A'_t. (Note the role of duality here.) It follows that all $m - t$ sinks of $\mathcal{W}$ in $\overline{A'_t}$ are rendered ELIGIBLE by the execution of the $n - b'$ sources of $\mathcal{W}$ in $\overline{B'_t}$. □

Claim 2. $E_{\Sigma_{\mathcal{W}}}(n - b) \;>\; E_{\Sigma_{\mathcal{W}}}(n - b')$.

Proof of Claim 2. Immediate from the assumption that $b < b'$, plus the fact that, for any IC-optimal schedule Σ, $E_{\Sigma}(t)$ is strictly increases with t. To wit, if T is a set of $t < n$ sources whose execution renders maximally many sinks ELIGIBLE, then executing any source that neighbors a member of T must render at least one more sink ELIGIBLE. □

It follows from Claim 2 that no packet can be empty. Note that A_t consists of the first b "packets" in (5.3), plus, *possibly*, part—but not all—of the $(b+1)$th "packet." We consider two cases, based on this possibility.

Case 1. A_t does not contain any nodes from the $(b+1)$th "packet" in (5.3).

Claim 3. *In this case,* $E_{\Sigma_{\mathcal{W}}}(n-b) = m-t$.

Proof of Claim 3. An argument parallel to that of Claim 1 shows that $E_{\Sigma_{\mathcal{W}}}(n-b) \geq m-t$ (there can be no arc from B_t to $\overline{A_t}$); we need, therefore, show only that $E_{\Sigma_{\mathcal{W}}}(n-b) \leq m-t$. We verify this as follows. If executing the sources of $\mathcal{W}$ in $\overline{B_t}$ rendered any sink $v \in A_t$ ELIGIBLE, then there would exist no arc in $\mathcal{W}$ of the form $(u \to v)$ for any $u \in B_t$. It follows that v would belong to some packet beyond the first b in (5.3), contrary to assumption. □

Claims 1–3 show that, in this case, $E_{\Sigma'_{\mathcal{W}}}(n-b') \geq m-t = E_{\Sigma_{\mathcal{W}}}(n-b) > E_{\Sigma_{\mathcal{W}}}(n-b')$, which refutes the alleged IC optimality of $\Sigma_{\mathcal{W}}$.

Case 2. A_t contains at least one node from the $(b+1)$th "packet" in (5.3).

Claim 4. *In this case,* $E_{\Sigma_{\mathcal{W}}}(n-b-1) < m-t$.

Proof of Claim 4. By assumption, A_t contains b full packets, $P_n, \ldots, P_{n-b+1}$, plus a portion of packet P_{n-b} that is neither empty nor full. It follows that when only the sources in A_t are EXECUTED, the sink $u_{k_{n-b}}$ that corresponds to P_{n-b} cannot be ELIGIBLE; to wit, that sink has an arc from every source in P_{n-b}, while not all sources in that packet are EXECUTED. Hence, the set B_t comprises precisely the b sinks $u_{k_n}, \ldots, u_{k_{n-b+1}}$. Consider now the effect of executing only the $n-b-1$ sources $u_{k_1}, \ldots, u_{k_{n-b-1}}$ under Σ. We clearly render ELIGIBLE only the sinks from packets $P_1, \ldots, P_{n-b-1}$—and none of those in A_t. Since A_t has strictly fewer sinks than do packets $P_n, \ldots, P_{n-b}$, the claimed strict inequality follows. □

Claims 1, 2, 4 show that, in this case, $E_{\Sigma'_{\mathcal{W}}}(n-b') \geq m-t > E_{\Sigma_{\mathcal{W}}}(n-b-1) \geq E_{\Sigma_{\mathcal{W}}}(n-b')$, which again refutes the alleged IC optimality of $\Sigma_{\mathcal{W}}$.

We conclude that any schedule for $\mathcal{M}_{\mathcal{W}}$ that is dual to $\Sigma_{\mathcal{W}}$ is IC optimal for $\mathcal{M}_{\mathcal{W}}$ whenever $\Sigma_{\mathcal{W}}$ is IC optimal for $\mathcal{W}$. □

5.1.2 Scheduling W-strands via their dual M-strands

We turn now to the intuition about M-strands and their dual W-strands. Let $\mathcal{M}$ be an M-strand with m sources, $\{v_1, \ldots, v_m\}$, and n sinks, $\{u_1, \ldots, u_n\}$. Let Σ be a schedule for $\mathcal{M}$ that renders $\mathcal{M}$'s sinks ELIGIBLE in the order (5.1), by executing sources packet by

packet, in the order (5.2). The (unique) *dual schedule to* Σ is the schedule for $\mathcal{M}$'s dual W-strand $\mathcal{W}_{\mathcal{M}}$ that executes $\mathcal{W}_{\mathcal{M}}$'s sources in the order

$$u_{k_n},\ u_{k_{n-1}},\ \ldots,\ u_{k_2},\ u_{k_1}. \tag{5.4}$$

Lemma 5.1. *Every IC-optimal schedule Σ for $\mathcal{M}$ operates in n stages: For each $l \in [1,n]$ in turn, Σ executes all un-*EXECUTED *sources of sink u_{k_l}. Consequently, for all $t \in [1, n-1]$, $E_\Sigma(t+1) \leq E_\Sigma(t) + 1$.*

Proof. We focus first on Σ's claimed execution regimen. Were Σ to deviate from this regimen, there would be a step t such that, Σ could have rendered a sink ELIGIBLE at step t only if it had executed all of some sink u's parents as fast as possible.

We turn next to the claim that Σ renders at most one new sink ELIGIBLE per step. Say, for contradiction, that there is a step t when executing a source x renders two or more sinks ELIGIBLE. The topology of $\mathcal{M}$ implies that these must be two neighboring sinks, y and y'. Just before step t, all parents of y and y', save for their shared parent, x, must be EXECUTED. Let P_y and $P_{y'}$ be, respectively, the parents of y and y', other than x.

Case 1. Some source $v \in P_y \cup P_{y'}$ has just one child. Say, for definiteness, that $v \in P_y$. Let $t' < t$ be the step when v gets EXECUTED. Let us create a "gap" in schedule Σ by not executing v at step t' (so Σ "idles" then). Clearly, the number of ELIGIBLE sinks at step $t-1$ does not change. Therefore, we can accelerate by one step each source-execution by Σ in steps $t'+1, \ldots, t-1$. The resulting "shifted" schedule has the same number of ELIGIBLE sinks at step $t-1$ as does Σ. Moreover, by step $t-1$, every parent of y', except for x, is EXECUTED. So, if we execute x during the gap that now exists at step $t-1$, we increase the number of ELIGIBLE sinks at that step, thus contradicting Σ's alleged IC optimality.

Case 2. Every source in $P_y \cup P_{y'}$ has two children. Lemma 5.1 and $\mathcal{M}$'s topology imply that, in this case, both y and y' have only two parents: x, and v for y, and v' for y'. Both v and v' must be EXECUTED before step t. Say, for definiteness, that v' is to the right of x. Pick the rightmost source w that is EXECUTED at some step $t'' < t$. We claim that had Σ executed x instead of w at step t'', this would have increased the number of ELIGIBLE sinks at step $t-1$, contradicting Σ's alleged IC optimality. To wit, if a sink is rendered ELIGIBLE when Σ executes w, then w must have had an EXECUTED lefthand neighbor at step t''—whence $w \neq v'$. Hence, executing w renders only one sink ELIGIBLE, while executing x renders two sinks ELIGIBLE. If, alternatively, no sink is rendered ELIGIBLE when Σ executes w, then the benefit of executing x at step t'', instead of w, is even greater. □

Lemma 5.1 implies that every packet is nonempty. To wit, if some packet P_ℓ were empty, then there would be a packet P_r, $r < \ell$, such that the execution of the last source in P_r renders both u_{k_ℓ} and u_{k_r} ELIGIBLE.

Theorem 5.2. *Let the M-strand $\mathcal{M}$ admit the IC-optimal schedule $\Sigma_{\mathcal{M}}$. The schedule $\Sigma_{\mathcal{W}_{\mathcal{M}}}$ for $\mathcal{W}_{\mathcal{M}}$ that is dual to $\Sigma_{\mathcal{M}}$ is IC optimal.*

Proof. Let $\Sigma_{\mathcal{M}}$ render $\mathcal{M}$'s sinks ELIGIBLE in the order (5.1), so that $\Sigma_{\mathcal{W}_{\mathcal{M}}}$ executes $\mathcal{W}_{\mathcal{M}}$'s sources in the order (5.4). Assume, for contradiction, that $\Sigma_{\mathcal{W}_{\mathcal{M}}}$ is not IC optimal for $\mathcal{W}_{\mathcal{M}}$. There must then be a schedule Σ for $\mathcal{W}_{\mathcal{M}}$ and a step $t \in [1, n]$ such that $E_{\Sigma}(t) > E_{\Sigma_{\mathcal{W}_{\mathcal{M}}}}(t)$. We show now that the existence of Σ refutes the alleged IC optimality of $\Sigma_{\mathcal{M}}$.

Some notation will facilitate our analysis. Let C_t denote the sources $u_{k_n}, \ldots, u_{k_{n-t+1}}$ of $\mathcal{W}_{\mathcal{M}}$ that $\Sigma_{\mathcal{W}_{\mathcal{M}}}$ executes during its first t steps, and let D_t denote the sinks of $\mathcal{W}_{\mathcal{M}}$ that these t executions render ELIGIBLE. Recall that every source from packet P_ℓ has an arc in $\mathcal{M}$ to sink u_{k_ℓ}, so that all sinks in D_t come from packets $P_n, \ldots, P_{n-t+1}$. (Sinks from other packets cannot become ELIGIBLE.) Let C'_t and D'_t denote, respectively, the analogous sets for schedule Σ. Note that $|C_t| = |C'_t| = t$, and that, by assumption,

$$d \stackrel{\text{def}}{=} |D_t| = E_{\Sigma_{\mathcal{W}_{\mathcal{M}}}}(t) < E_{\Sigma}(t) = |D'_t| \stackrel{\text{def}}{=} d'.$$

Back to the proof: Consider any schedule $\Sigma'_{\mathcal{M}}$ for $\mathcal{M}$ that executes the sources of $\mathcal{M}$ in the order $[[\overline{D'_t}]]$, $[[D'_t]]$; in other words:

1. $\Sigma'_{\mathcal{M}}$ begins by executing, in some order, the $m - d'$ sources of $\mathcal{M}$ in the set $\overline{D'_t}$;
2. $\Sigma'_{\mathcal{M}}$ then executes, in some order, the d' sources of $\mathcal{M}$ in the set D'_t.

Consider the numbers of sinks of $\mathcal{M}$ that become ELIGIBLE at various steps under schedules $\Sigma_{\mathcal{M}}$ and $\Sigma'_{\mathcal{M}}$. The following fact mirrors Claim 1 in Theorem 5.1 and is left to the reader.

Claim 5. $E_{\Sigma'_{\mathcal{M}}}(m - d') \geq n - t$.

Claim 6. $E_{\Sigma_{\mathcal{M}}}(m - d) > E_{\Sigma_{\mathcal{M}}}(m - d')$.

Proof of Claim 6. The inequality follows from three observations.

1. $\Sigma_{\mathcal{M}}$ renders all $n - t$ sinks of $\mathcal{M}$ in $\overline{C_t}$ ELIGIBLE when it executes the $m - d$ sources of $\mathcal{M}$ in $\overline{D_t}$. This is because no sink of $\mathcal{M}$ in $\overline{C_t}$ has a parent in D_t (cf. Claim 5).

2. Each source u of $\mathcal{M}$ in $\overline{D_t}$ has a child in $\overline{C_t}$. Otherwise, $u \in D_t$.

3. After $\Sigma_{\mathcal{M}}$ has executed $m - d' < m - d$ sources of $\mathcal{M}$, at least one source, $v \in \overline{D_t}$, of $\mathcal{M}$ remains un-EXECUTED. v's child in $\overline{C_t}$ thus remains un-ELIGIBLE. □

Claim 7. $E_{\Sigma_{\mathcal{M}}}(m - d) = n - t$.

Proof of Claim 7. Repeating the proof of Claim 5 for $\Sigma_{\mathcal{M}}$ establishes that $E_{\Sigma_{\mathcal{M}}}(m-d) \geq n-t$, so it suffices to show that $E_{\Sigma_{\mathcal{M}}}(m-d) \leq n-t$. We accomplish this by showing that each sink of $\mathcal{M}$ in C_t has a parent in D_t within $\mathcal{M}$. To see this, observe that, the sources from any packet P_ℓ, in $\mathcal{M}$, cannot have arcs to any earlier sink u_{k_r}, $r < \ell$, because the such an arc would prevent u_{k_r} from becoming ELIGIBLE after packet P_r has been executed. Therefore, D_t actually contains all sinks of $\mathcal{W}_{\mathcal{M}}$ from packets $P_n, \ldots, P_{n-t+1}$. But every packet P_ℓ is nonempty, and each of its sources in $\mathcal{M}$ has an arc to sink u_{k_ℓ}. Since C_t comprises the sources $u_{k_n}, \ldots, u_{k_{n-t+1}}$ of $\mathcal{W}_{\mathcal{M}}$, and D_t includes sinks from $P_n, \ldots, P_{n-t+1}$, any sink from C_t clearly has a parent in D_t within $\mathcal{M}$. □

Claims 5–7 combine to show that $E_{\Sigma'_{\mathcal{M}}}(m-d') \geq n-t = E_{\Sigma_{\mathcal{M}}}(m-d) > E_{\Sigma_{\mathcal{M}}}(m-d')$, which refutes the alleged IC optimality of $\Sigma_{\mathcal{M}}$.

We conclude that $\Sigma_{\mathcal{W}_{\mathcal{M}}}$ is IC optimal for $\mathcal{W}_{\mathcal{M}}$ whenever $\Sigma_{\mathcal{M}}$ is IC optimal for $\mathcal{M}$. □

5.2 Priority-Based Duality of W-strands and M-strands

We now derive $\triangleright$-priorities between two W-strands or two M-strands from $\triangleright$-priorities between their dual M-strands or W-strands, respectively.

Since the system of inequalities (2.1) that defines $\triangleright$ is not convenient for establishing $\triangleright$-priorities among M-strands, we invoke a more convenient dual formulation of the relation.

Let Σ be a schedule for the S-sink bipartite dag $\mathcal{G}$. For any $e \in [0, S]$, let $X_\Sigma(e)$ be the smallest number of sources of $\mathcal{G}$ that must be executed in order to render at least e sinks of $\mathcal{G}$ ELIGIBLE. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be disjoint bipartite dags that, respectively, have S_1 and S_2 sinks and IC-optimal schedules Σ_1 and Σ_2. If the following inequalities hold:

$$\begin{array}{l}(\forall e \in [0, S_1])\ (\forall f \in [0, S_2]): \\ X_{\Sigma_1}(e) + X_{\Sigma_2}(f) \ \geq\ X_{\Sigma_1}(\min\{e+f, S_1\}) + X_{\Sigma_2}(\max\{0, e+f-S_1\}),\end{array} \tag{5.5}$$

then $\mathcal{G}_1$ *has dual priority over* $\mathcal{G}_2$, denoted $\mathcal{G}_1 \widetilde{\triangleright} \mathcal{G}_2$. Clearly, relation $\widetilde{\triangleright}$ is equivalent to $\triangleright$: the former relation strives to minimize the number of EXECUTED sources for a given number of ELIGIBLE sinks; the latter strives to maximize the number of ELIGIBLE sinks for a given number of EXECUTED sources. Stated formally,

Lemma 5.2. *Let bipartite dags $\mathcal{G}_1$ and $\mathcal{G}_2$ admit IC-optimal schedules Σ_1 and Σ_2, respectively. $\mathcal{G}_1 \triangleright \mathcal{G}_2$ if, and only if, $\mathcal{G}_1 \widetilde{\triangleright} \mathcal{G}_2$.*

Lemma 5.2 affords us easy access to the following result. (Note the reversal of indices.)

Theorem 5.3. *For any W-strands $\mathcal{W}_1$ and $\mathcal{W}_2$: $\mathcal{W}_1 \triangleright \mathcal{W}_2$ if, and only if, $\mathcal{M}_{\mathcal{W}_2} \triangleright \mathcal{M}_{\mathcal{W}_1}$.*

Proof. For $i = 1, 2$, let $\mathcal{W}_i$ have s_i sources and S_i sinks, and let it admit the IC-optimal schedule Σ_i. Theorem 5.1 tells us how to construct, from Σ_i an IC-optimal schedule $\widehat{\Sigma}_i$ for $\widehat{\mathcal{W}}_i = \mathcal{M}_{\mathcal{W}_i}$. Moreover, the proof of the Theorem gives us valuable information about how Σ_i and $\widehat{\Sigma}_i$ operate. In particular, using the notation of the proof, we recall that, in order to render b sinks of $\mathcal{M}_{\mathcal{W}}$ ELIGIBLE, we must execute sources in packets $P_n, \ldots, P_{n-b+1}$. Moreover, executing $n - b$ sources of $\mathcal{W}$ renders ELIGIBLE exactly the sinks of packets $P_1, \ldots, P_{n-b}$. Hence, for all $e \in [0, s_i]$, $X_{\widehat{\Sigma}_i}(e) = S_i - E_{\Sigma_i}(s_i - e)$. Thus, for any $e \in [0, s_1]$ and $f \in [0, s_2]$, we have

$$X_{\widehat{\Sigma}_1}(e) + X_{\widehat{\Sigma}_2}(f) = (S_1 + S_2) - E_{\Sigma_1}(s_1 - e) - E_{\Sigma_2}(s_2 - f). \tag{5.6}$$

Say now that $\mathcal{W}_1 \rhd \mathcal{W}_2$. By (2.1), we then have

$$E_{\Sigma_1}(s_1 - e) + E_{\Sigma_2}(s_2 - f) \leq E_{\Sigma_1}(\min\{s_1, s_1 + s_2 - e - f\}) + E_{\Sigma_2}(\max\{0, s_2 - e - f\}).$$

Combining this inequality with (5.6), we find that

$$\begin{aligned} X_{\widehat{\Sigma}_1}(e) + X_{\widehat{\Sigma}_2}(f) &\geq (S_1 + S_2) - E_{\Sigma_1}(\min\{s_1, s_1 + s_2 - e - f\}) \\ &\quad - E_{\Sigma_2}(\max\{0, s_2 - e - f\}) \\ &= X_{\widehat{\Sigma}_1}(s_1 - \min\{s_1, s_1 + s_2 - e - f\}) \\ &\quad + X_{\widehat{\Sigma}_2}(s_2 - \max\{0, s_2 - e - f\}) \\ &= X_{\widehat{\Sigma}_1}(\max\{0, e + f - s_2\}) + X_{\widehat{\Sigma}_2}(\min\{s_2, e + f\}). \end{aligned} \tag{5.7}$$

This last inequality means that $\mathcal{M}_{\mathcal{W}_2} \widetilde{\rhd} \mathcal{M}_{\mathcal{W}_1}$, so that, by Lemma 5.2, $\mathcal{M}_{\mathcal{W}_2} \rhd \mathcal{M}_{\mathcal{W}_1}$, as claimed. The converse follows "by running the argument backwards." □

Corollary 5.1. *For any M-strands $\mathcal{M}_1$ and $\mathcal{M}_2$: If $\mathcal{M}_1 \rhd \mathcal{M}_2$, then $\mathcal{W}_{\mathcal{M}_2} \rhd \mathcal{W}_{\mathcal{M}_1}$.*

Proof. Every W-strand (resp., M-strand) is dual to its dual M-strand (resp., W-strand). □

6 Where We Are, and Where We're Going

6.1 Conclusions

The results in this paper significantly expand the range of computation-dags that the algorithmic theory of [11] can schedule IC optimally. In addition to the structurally uniform dags of Fig. 1 and the simple composite dags of [11], we are now able to schedule IC optimally any sequence of *sums of T-strands* that is linearly ordered under $\rhd$-priority: $\mathcal{T}_1 \rhd \mathcal{T}_2 \rhd \cdots \rhd \mathcal{T}_n$. We are, consequently, now able to schedule IC optimally the following three classes of computation-dags, which are reminiscent of the dags encountered in a variety of scientific computations. Let us be given:

- any sequence of *sums of W-strands* that is linearly ordered under $\triangleright$-priority: $\breve{\mathcal{S}}_1 \triangleright \breve{\mathcal{S}}_2 \triangleright \cdots \triangleright \breve{\mathcal{S}}_n$;
- any sequence of *sums of M-strands* that is linearly ordered under $\triangleright$-priority: $\widehat{\mathcal{S}}_1 \triangleright \widehat{\mathcal{S}}_2 \triangleright \cdots \triangleright \widehat{\mathcal{S}}_m$.

Then any composite dag of one of the following three types admits the IC-optimal schedule dictated by Theorem 2.1:

(1) $\breve{\mathcal{S}}_1 \Uparrow \breve{\mathcal{S}}_2 \Uparrow \cdots \Uparrow \breve{\mathcal{S}}_n$,
(2) $\widehat{\mathcal{S}}_1 \Uparrow \widehat{\mathcal{S}}_2 \Uparrow \cdots \Uparrow \widehat{\mathcal{S}}_m$,
(3) $\breve{\mathcal{S}}_1 \Uparrow \breve{\mathcal{S}}_2 \Uparrow \cdots \Uparrow \breve{\mathcal{S}}_n \Uparrow \widehat{\mathcal{S}}_1 \Uparrow \widehat{\mathcal{S}}_2 \Uparrow \cdots \Uparrow \widehat{\mathcal{S}}_m$.

Intuitively, we now have scheduling control over a quite broad family of dags that are expansive, reductive, and expansive-reductive.

6.2 Projections

Our work on this project proceeds in a number of directions.

Theory. We are engaged in investigations aimed at extending the scope of the theory of [11] in several ways.

1. We are expanding the scheduling component of our theory to move beyond its current dependence on $\triangleright$-linear compositions of building blocks. Our new work focuses on (IC-optimal) schedules that allow the *interleaved* execution of building blocks.
2. We are seeking a rigorous framework for devising schedules that are "approximately" IC optimal. This thrust is important for computational reasons—a computationally simple heuristic may be "almost as good" as a more arduously derived IC-optimal schedule—and because many dags do not admit IC-optimal schedules.
3. We are working to extend our theory so that it can optimally schedule composite dags whose building blocks are not necessarily bipartite.

Simulations and experiments. We are currently engaged in a suite of simulation experiments that seek to determine the extent to which our scheduling algorithms actually enhance the efficiency of Internet-based computations. The simulations will include real scientific dags that coauthor G. Malewicz studied during a summer, 2005, visit to Argonne National Lab., as well as artificially generated ones.

Integration with grid schedulers. Perhaps our most ambitious non-theoretical endeavor involves incorporating our suite of scheduling algorithms into a real scheduling tool. We

have taken steps in this directions by developing a tool [4] for prioritizing the jobs of a DAGMan file. The tool was implemented by coauthor G. Malewicz his visit at Argonne National Lab. An integration of the tool within the Condor project [3] is under way.

Acknowledgments. A portion of the research of G. Cordasco and G. Malewicz was done while visiting the TAPADS Group at the Univ. of Massachusetts Amherst. A portion of the research of G. Malewicz was done while visiting the Mathematics and Computer Science Division of Argonne National Laboratory. The research of G. Malewicz was supported in part by NSF Grant ITR-800864. The research of A. Rosenberg was supported in part by NSF Grant CCF-0342417. The authors are grateful to Matt Yurkewych (UMass) and Michael Wilde (Argonne) for valuable conversations at several stages of the research reported here. G. Malewicz wishes to thank Ian Foster (Argonne) for hosting his visit at Argonne and Frederica Darema (NSF) for support that facilitated that visit.

References

[1] R. Buyya, D. Abramson, J. Giddy (2001): A case for economy Grid architecture for service oriented Grid computing. *10th Heterogeneous Computing Wkshp.*

[2] W. Cirne and K. Marzullo (1999): The Computational Co-Op: gathering clusters into a metacomputer. *13th Intl. Parallel Processing Symp.*, 160–166.

[3] Condor Project, University of Wisconsin. `http://www.cs.wisc.edu/condor`

[4] I. Foster, G. Malewicz, A.L. Rosenberg, M. Wilde (2006): A tool for prioritizing DAGMan jobs for internet-based computing. In preparation.

[5] I. Foster and C. Kesselman [eds.] (2004): *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*. Morgan-Kaufmann, San Francisco.

[6] I. Foster, C. Kesselman, S. Tuecke (2001): The anatomy of the Grid: enabling scalable virtual organizations. *Intl. J. Supercomputer Applications.*

[7] L. Gao and G. Malewicz (2005): Toward maximizing the quality of results of dependent tasks computed unreliably. *Theory of Computing Systs,*, to appear. See also, *Intl. Conf. on Principles of Distributed Systems*, 2004.

[8] D. Kondo, H. Casanova, E. Wing, F. Berman (2002): Models and scheduling mechanisms for global computing applications. *Intl. Parallel and Distr. Processing Symp.*

[9] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, M. Lebofsky (2000): SETI@home: massively distributed computing for SETI. In *Computing in Science and Engineering* (P.F. Dubois, Ed.) IEEE Computer Soc. Press, Los Alamitos, CA.

[10] G. Malewicz and A.L. Rosenberg (2005): On batch-scheduling dags for Internet-based computing. *Euro-Par 2005*, 262–271.

[11] G. Malewicz, A.L. Rosenberg, M. Yurkewych (2005): Toward a theory for scheduling dags in Internet-based computing. *IEEE Trans. Comput.*, to appear. See also, *Intl. Parallel and Distr. Processing Symp.*, 2005.

[12] A.L. Rosenberg (2004): On scheduling mesh-structured computations for Internet-based computing. *IEEE Trans. Comput. 53*, 1176–1186.

[13] A.L. Rosenberg and M. Yurkewych (2005): Guidelines for scheduling some common computation-dags for Internet-based computing. *IEEE Trans. Comput. 54*, 428–438.

[14] X.-H. Sun and M. Wu (2003): GHS: A performance prediction and node scheduling system for Grid computing. *IEEE Intl. Parallel and Distributed Processing Symp.*