On Scheduling Dags to Maximize Area

Gennaro Cordasco*
University of Salerno
cordasco@dia.unisa.it

Arnold L. Rosenberg†
Colorado State University
rsnbrg@engr.colostate.edu

Abstract

A new quality metric, called area*, is introduced for schedules that execute dags, i.e., computations having intertask dependencies. Motivated by the temporal unpredictability encountered when computing over the Internet, the goal under the new metric is to maximize the* average *number of tasks that are eligible for execution at each step of a computation. Area-maximization is a weakening of* IC-optimality, *which strives to maximize the number of eligible tasks at* every *step of the computation. In contrast to IC-optimal schedules, area-maximizing schedules exist for* every *dag. For dags that admit IC-optimal schedules, all area-maximizing schedules are IC-optimal, and vice versa. The basic properties of this metric are derived in this paper, and tools for efficiently crafting area-maximizing schedules for large classes of computationally significant dags are developed. Several of these results emerge from a close connection between area-maximizing scheduling and the* MAX Linear-Arrangement Problem for Dags.

1. Introduction

This paper studies a new quality metric, *area*, for schedules that execute computation-dags (i.e., computations having intertask dependencies). The metric is motivated by the *temporal unpredictability* encountered when computing over the Internet, which precludes the accurate use of classical dag-scheduling metrics.

Advancing technology has rendered the Internet a viable medium for enlisting multiple independent computers to collaboratively solve a single computational problem. Several mechanisms have been developed for Internet-based Computing (IC); Volunteer computing [10], Peer-to-Peer computing (P2P) [2, 17], and Grid computing [7, 8] are among the most popular. Most forms of IC employ a server-client metaphor: remote client computers compute one task at a time from a massive collection of compute-intensive tasks; a client receives a new task from the server when it returns the results from its previous one.

As with all new computing technologies, IC engenders novel scheduling challenges, even while enabling a large variety of computations that could not be handled efficiently by any fixed assemblage of dedicated computers (e.g., multiprocessors or clusters). The dominant challenge that the current study addresses is how to schedule IC computations given the *temporal unpredictability* of all modalities of IC, which has two main sources: (1) Clients typically become available (to receive work) at unpredictable times; moreover, they are typically not dedicated to the server's IC computation. One can, therefore, not predict the computation time of remote tasks precisely. (In Volunteer computing, a client may *never* return its results.) (2) Communication is over the Internet, so its speed is impossible to predict precisely. When the tasks being computed are mutually independent, (finite) computation or communication delays are just an annoyance: they never deprive the server of tasks to allocate to a newly available client. However, when the tasks have interdependencies that constrain their order of execution, such delays may restrict the supply of tasks eligible for allocation, because (almost) all tasks that are eligible for execution have already been allocated. This could prevent one from exploiting all of the computing power that one has access to if more remote clients are available than one has tasks for. Indeed, in the limit, one's computation could stall for lack of tasks that are eligible for allocation. The current strives to mitigate the negative impact of temporal unpredictability in IC.

In common with most sources in the scheduling literature, we represent a computation having intertask dependencies as a *dag*. Our goal is an algorithmic theory of how to craft dag-executing schedules that allocate eligible tasks to clients *in an order that maximizes the number of tasks that are eligible for execution*. We have studied one variant of this goal that seeks to maximize this number *at every step of the computation*; here we study one that aims to maximize the *average* number of eligible tasks; cf. [9]. An

*Dip. di Informatica ed Applicazioni, Univ. of Salerno, Fisciano 84084, ITALY

†Dept. of Electrical and Computer Engineering, Colorado State Univ., Fort Collins, CO 81523, USA

978-1-4244-3750-4/09/$25.00 ©2009 IEEE

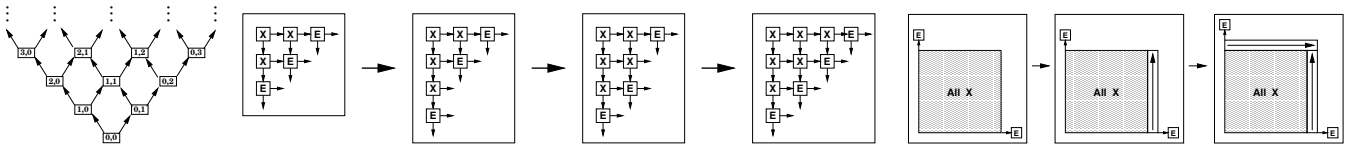

Figure 1. (Left) *The* 2*-dimensional evolving mesh* $\mathcal{M}_2$. (Center) *Computing a* diagonal *shell of* $\mathcal{M}_2$. (Right) *Computing a* square *shell of* $\mathcal{M}_2$. "X" *marks an executed node,* "E" *an eligible one.*

example suggests the significance of these goals. Say that one wants to schedule a wavefront computation whose task-dependencies have the structure of the evolving mesh $\mathcal{M}$ in Fig. 1(left). If one schedules $\mathcal{M}$ along "diagonal" levels [Fig. 1(center)], then executing x tasks leaves roughly $\sqrt{x}$ tasks eligible to be executed next. If one schedules $\mathcal{M}$ along "square" shells [Fig. 1(right)], then there are never more than *three* eligible tasks! The order of executing eligible tasks can thus have a major impact on the rate of producing new eligible tasks.

Our ongoing work on this topic [3, 5, 12, 14, 15, 16] has begun to develop *IC-Scheduling theory*, which seeks to maximize the number of eligible tasks *at every step of a computation*. Simulations in [9, 11] suggest that computations that achieve this stringent goal do complete significantly faster in a variety of typical IC situations; the case studies in [4] develop schedules for a range of important computations and computational paradigms that are optimal within the theory. The current study is motivated by the fact that many computationally significant dags *do not admit any optimal schedule* under the stringent goal [12].

In this paper, we weaken the "at every step" goal of IC-Scheduling, in favor of maximizing the *average* number of eligible tasks over the steps of the computation. The resulting *AREA* metric for a schedule also has close relations with other dag-theoretic notions. AREA-maximization subsumes the goal of IC-Scheduling (Lemma 2):

• *Every dag* $\mathcal{G}$ *admits an AREA-maximizing schedule.*

• *If* $\mathcal{G}$ *admits an optimal schedule under IC-Scheduling, then: (a) that schedule maximizes AREA; (b) every AREA-maximizing schedule is optimal under IC-Scheduling.*

Our main results. In addition to the preceding results: (1) We show how to "read off" an AREA-maximizing schedule for a dag $\mathcal{G}$ from such a schedule for $\mathcal{G}$'s *dual* (whose arcs are the reversals of $\mathcal{G}$'s). (2) We exhibit a close connection between the quest for AREA-maximizing schedules and the *Maximum Linear Arrangement problem for dags* [13]; we use these connections to bound the complexity of scheduling general dags and to derive an efficient scheduling algorithm for "monotonic" tree-dags. (3) We establish several results about scheduling dags that are compositions of bipartite cliques and cycles, two classes of dags that are the most intransigent for IC-Scheduling; several of these results can be adapted to other families of dags.

2. Computation-Dags and Schedules

2.1. Basic Definitions

A. Computation-dags. A *(computation-)dag* $\mathcal{G}$ has a set $\mathcal{N}_\mathcal{G}$, of cardinality $N_\mathcal{G}$, of *nodes*, each representing a task in a computation, and a set $\mathcal{A}_\mathcal{G}$. of cardinality $A_\mathcal{G}$, of *arcs*, each representing an intertask dependency. For arc $(u \to v) \in \mathcal{A}_\mathcal{G}$: • task v cannot be executed until task u is; • u is a *parent* of v, and v is a *child* of u in $\mathcal{G}$. The *indegree* (resp., *outdegree*) of $u \in \mathcal{N}_\mathcal{G}$ is its number of parents (resp., children). A parentless node is a *source*; a childless node is a *sink*. $\mathcal{G}$ is *bipartite* if $\mathcal{N}_\mathcal{G}$ can be partitioned into X and Y, and each arc $(u \to v)$ has $u \in X$ and $v \in Y$. $\mathcal{G}$ is *connected* if it is so when one ignores arc orientations. When $\mathcal{N}_{\mathcal{G}_1} \cap \mathcal{N}_{\mathcal{G}_2} = \emptyset$, the *sum* $\mathcal{G}_1 + \mathcal{G}_2$ of dags $\mathcal{G}_1$ and $\mathcal{G}_2$ is the dag with node-set $\mathcal{N}_{\mathcal{G}_1} \cup \mathcal{N}_{\mathcal{G}_2}$ and arc-set $\mathcal{A}_{\mathcal{G}_1} \cup \mathcal{A}_{\mathcal{G}_2}$.

B. Dag schedules and their quality. When executing a dag $\mathcal{G}$, each $v \in \mathcal{N}_\mathcal{G}$ becomes ELIGIBLE (for execution) once all its parents have been executed. (Hence, sources are ELIGIBLE until executed.) We do not recompute nodes, so a node ceases to be ELIGIBLE once executed. In compensation, executing $v \in \mathcal{N}_\mathcal{G}$ may render new nodes ELIGIBLE when v is their last parent to be executed. A *schedule* for $\mathcal{G}$ is a rule for selecting which ELIGIBLE node to execute at each step when executing $\mathcal{G}$. (It is, thus, a *topological sort* of $\mathcal{G}$ [6].)

We measure the IC quality of a schedule Σ via the rate at which Σ's successive node-executions produce new ELIGIBLE nodes—the more, the better. (*We measure time in an event-driven manner*, as the number of nodes that have been executed to that point.) Our earlier work strove to execute $\mathcal{G}$'s nodes in an order that maximizes the number of ELIGIBLE nodes on $\mathcal{G}$ *at every step of the computation*; a schedule that achieves this demanding goal is IC-optimal. Dual intuition motivates IC optimality: (1) Schedules that produce ELIGIBLE nodes more quickly may prevent a computation's stalling pending the return of already allocated tasks. (2) If the server receives many requests for tasks at (roughly) the same time, then having more ELIGIBLE tasks

available allows it to satisfy more requests, thereby increasing "parallelism." Because *many dags admit no IC-optimal schedule* [12], we are moderating the goal of IC optimality here, by striving to maximize the *average* number of ELIGIBLE nodes on $\mathcal{G}$ at each step—a goal that can *always* be achieved; a schedule that achieves this goal is AREA-maximizing.

2.2. The AREA-Maximization Metric

A. The basic metric. For schedule Σ for dag $\mathcal{G}$, and integer $T \in [0, N_{\mathcal{G}}]$,[1] $E_\Sigma(T)$ is the number of nodes of $\mathcal{G}$ that are ELIGIBLE at step T when Σ executes $\mathcal{G}$.[2] Σ's *eligibility profile* is the $(N_{\mathcal{G}}+1)$-entry vector $\Pi(\Sigma) = \langle E_\Sigma(0), E_\Sigma(1), \ldots, E_\Sigma(N_{\mathcal{G}})\rangle$. (We include $E_\Sigma(0)$ (the number of sources of $\mathcal{G}$) and $E_\Sigma(N_{\mathcal{G}})$ ($\equiv 0$) for completeness.)

The *AREA* of schedule Σ, $AREA(\Sigma)$, is the sum

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

The normalized AREA, $\widehat{E}(\Sigma) \stackrel{\text{def}}{=} AREA(\Sigma) \div N_{\mathcal{G}}$, is the average number of nodes that are ELIGIBLE when Σ executes $\mathcal{G}$. We term this average "*area*" by formal analogy with using Riemann sums to approximate integrals. Our goal is to find an AREA-maximizing schedule for $\mathcal{G}$, i.e., a schedule $\Sigma^\star$ such that

$$AREA(\Sigma^\star) = \max_{\Sigma \text{ a schedule for } \mathcal{G}} AREA(\Sigma).$$

B. Streamlining the metric. The following lemma simplifies the quest for AREA-maximizing schedules.

Lemma 1 *Altering a schedule Σ for dag $\mathcal{G}$ so that it executes all of $\mathcal{G}$'s nonsinks before any of its sinks cannot decrease Σ's AREA-quality.*

Proof. For $i \in [0, N_{\mathcal{G}}]$, let $e_i = E_\Sigma(i)$.: if Σ executes a sink at step $h \in [0, N_{\mathcal{G}}]$, then $e_h = e_{h-1} - 1$. We show by induction that deferring the execution of a sink can only improve $AREA(\Sigma)$. Say that Σ executes sink $v \in \mathcal{N}_{\mathcal{G}}$ at step ℓ. Compare $AREA(\Sigma)$ with the AREA of the schedule Σ_v that mimics Σ in every way, *except* that it delays executing v until step $N_{\mathcal{G}}$ (the last step). $AREA(\Sigma_v) = e'_0 + \cdots + e'_{N_{\mathcal{G}}}$, where

$$e'_i = \begin{cases} e_i & \text{for } i \in [0, \ell-1] \\ e_i + 1 & \text{for } i \in [\ell, N_{\mathcal{G}} - 1]. \end{cases} \quad (2)$$

This is valid because at step $\in [\ell, N_{\mathcal{G}} - 1]$, Σ_v has the same ELIGIBLE nodes as Σ, *plus* node v. We thus have:

$$AREA(\Sigma_v) = AREA(\Sigma) - e_\ell + (N_{\mathcal{G}} - \ell).$$

We are now done because Σ has $N_{\mathcal{G}} - \ell \leq e_\ell$ node-executions remaining. ∎

Focus on a dag $\mathcal{G}$ with n nonsinks, N nonsources, s sources, and S sinks (note: $N_{\mathcal{G}} = s + N = S + n$).

The *Area* metric. By Lemma 1, we lose no generality by focusing on schedules that execute $\mathcal{G}$'s sinks only after all of its nonsinks. Certain segments of $\mathcal{G}$'s execution proceed the same under any such schedule. We can streamline analyses by ignoring these segments. The last S entries in the profile $\Pi(\Sigma)$ of such a schedule are: $S-1, \ldots, 1, 0$. As we seek an AREA-maximizing schedule for $\mathcal{G}$, we can, thus, ignore these last S entries and strive to maximize, not $AREA(\Sigma)$, but, rather

$$Area(\Sigma) \stackrel{\text{def}}{=} \sum_{i=0}^{n} E_\Sigma(i) = AREA(\Sigma) - \binom{S}{2}. \quad (3)$$

The *area* metric. Any schedule Σ that honors Lemma 1 can be viewed as a linearization of $\mathcal{G}$'s nonsinks followed by a linearization of its sinks. For each $t \in [1, N_{\mathcal{G}}]$, let $e_\Sigma(t)$ denote the number of nodes (perforce, nonsources) of $\mathcal{G}$ that are rendered ELIGIBLE by the node-execution at step t of Σ. Clearly, $E_\Sigma(t) = s - t + \sum_{j=1}^{t} e_\Sigma(j)$. Hence,

$$\begin{aligned} Area(\Sigma) &= \sum_{t=0}^{n} E_\Sigma(t) = \sum_{t=0}^{n} \left(s - t + \sum_{j=1}^{t} e_\Sigma(j) \right) = \\ &= \sum_{t=1}^{n} \sum_{j=1}^{t} e_\Sigma(j) + (n+1)s - \binom{n+1}{2}. \end{aligned}$$

The only portion of $Area(\Sigma)$ that actually depends on choices made by Σ is, thus,

$$\begin{aligned} area(\Sigma) &\stackrel{\text{def}}{=} \sum_{t=1}^{n} \sum_{j=1}^{t} e_\Sigma(j) \\ &= n e_\Sigma(1) + (n-1) e_\Sigma(2) + \ldots + e_\Sigma(n). \end{aligned}$$

We can now choose among three essentially equivalent metrics, $AREA(\Sigma)$, $Area(\Sigma)$, and $area(\Sigma)$, as dictated by each particular analysis of schedules.

2.3. IC-Optimality vs. AREA-Maximality

Our quest for an AREA-maximizing schedule for dag $\mathcal{G}$ strictly subsumes the quest for an IC-optimal schedule for $\mathcal{G}$, when $\mathcal{G}$ admits an IC-optimal schedule.

Lemma 2 (a) *Whereas some dags do not admit any IC-optimal schedule [12], every dag admits an AREA-maximizing schedule.* **(b)** *If dag $\mathcal{G}$ admits an IC-optimal schedule, then:* (i) *every AREA-maximizing schedule for $\mathcal{G}$ is IC optimal;* (ii) *every IC-optimal schedule for $\mathcal{G}$ is AREA-maximizing.*

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

[2] The condition for IC-optimality can now be written: $(\forall t)\ E_\Sigma(t) = \max_{\Sigma' \text{ a schedule for } \mathcal{G}} \{E_{\Sigma'}(t)\}$.

Proof Sketch. **(a)** $\mathcal{G}$ admits an AREA-maximizing schedule because it admits $\leq N_{\mathcal{G}}!$ schedules in all. The first known dags that do not admit IC-optimal schedules appear in [12].[3] **(b)** By definition. ■

3. AREA-Maximization via Duality

The *dual* of dag $\mathcal{G}$ is the dag $\widetilde{\mathcal{G}}$ obtained by reversing all of $\mathcal{G}$'s arcs. One can "read off" an AREA-maximizing schedule for either of $\mathcal{G}$ or $\widetilde{\mathcal{G}}$ from such a schedule for the other.

Let $\mathcal{G}$ have n nonsinks, the set $U=\{u_1,\ldots,u_n\}$, and N nonsources, the set $V=\{v_1,\ldots,v_N\}$. Let schedule Σ for $\mathcal{G}$ execute $\mathcal{G}$'s nonsinks in the order $u_{k_1},\ldots,u_{k_n}$, after which, Σ executes $\mathcal{G}$'s sinks. Each execution of a nonsink u_{k_j} renders ELIGIBLE a (possibly empty) *packet* of nonsources, $P_j=\{v_{j,1},\ldots,v_{j,i_j}\}$. (Clearly, $\mathcal{G}$ has an arc from u_{k_j} to each $v\in P_j$.) Thus, Σ renders $\mathcal{G}$'s nonsources ELIGIBLE in a sequence of packets:

$$P_1=\{v_{1,1},\ldots,v_{1,i_1}\},\ \ldots,\ P_n=\{v_{n,1},\ldots,v_{n,i_n}\}. \quad (4)$$

A schedule $\widetilde{\Sigma}$ for $\widetilde{\mathcal{G}}$ is *dual to* Σ if it executes $\widetilde{\mathcal{G}}$'s nonsinks—which, recall, are $\mathcal{G}$'s nonsources—in an order of the form[4] $[[v_{n,1},\ldots,v_{n,i_n}]],\ \ldots,\ [[v_{1,1},\ldots,v_{1,i_1}]]$, after which, $\widetilde{\Sigma}$ executes $\widetilde{\mathcal{G}}$'s sinks. $\widetilde{\mathcal{G}}$ will generally admit many schedules that are dual to Σ. Note that both Σ and $\widetilde{\Sigma}$ honor Lemma 1.

Theorem 1 *If a schedule Σ is AREA-maximizing for a dag $\mathcal{G}$, then any schedule $\widetilde{\Sigma}$ that is dual to Σ is AREA-maximizing for $\widetilde{\mathcal{G}}$.*

Proof. We compare $area(\Sigma)$ with $area(\widetilde{\Sigma})$ for some arbitrary $\widetilde{\Sigma}$ that is dual to Σ.

Lemma 3 *If $\widetilde{\Sigma}$ is dual to Σ, then $area(\widetilde{\Sigma})\geq area(\Sigma)+n-N$.*

Proof Sketch. Let schedule $\widetilde{\Sigma}$ for $\widetilde{\mathcal{G}}$ be dual to Σ. Whenever all nodes in a packet P_t are executed under $\widetilde{\Sigma}$, at least one nonsource of $\widetilde{\mathcal{G}}$ becomes ELIGIBLE. Hence,

$$area(\widetilde{\Sigma}) = Ne_{\widetilde{\Sigma}}(1) + (N-1)e_{\widetilde{\Sigma}}(2) + \cdots + e_{\widetilde{\Sigma}}(N). \quad (5)$$

Note next that, for each $i\in[0,n-1]$, if $e_\Sigma(n-i)>0$, then $P_{n-i}\neq\emptyset$, so that

$$e_{\widetilde{\Sigma}}\left(\sum_{j=0}^{i} e_\Sigma(n-j)\right) \geq 1.$$

Similarly, for each $i\in[0,n-i]$, if $e_\Sigma(n-i)>0$ *and* $\sum_{k=1}^{x} e_\Sigma(n-i-k)=0$, then $P_{n-i}\neq\emptyset$ but the x packets preceding it are empty, so that

$$e_{\widetilde{\Sigma}}\left(\sum_{j=0}^{i} e_\Sigma(n-j)\right) \geq x+1.$$

In this case, executing all in P_{n-i} and its x preceding empty packets renders $\geq x+1$ nodes ELIGIBLE, namely, $u_{k_{n-i}}$, $\ldots$, $u_{k_{n-i-x}}$. Invoking (5), we therefore have the following lower bound on $area(\widetilde{\Sigma})$. In expressing the bound, we use double parentheses to convert a logical value to a numerical one; e.g., $((a>0))=$ **if** $a>0$ **then** 1 **else** 0. Careful analysis of the contributions of executing the various packets yields:

$$\begin{aligned}
area(\widetilde{\Sigma}) \;\geq\;& \left(\sum_{j=1}^{n-1} e_\Sigma(j)+1\right)\cdot e_{\widetilde{\Sigma}}(e_\Sigma(n)) \\
&+((e_\Sigma(n-1)>0))\cdot \\
&\quad\cdot\left(\sum_{j=1}^{n-2} e_\Sigma(j)+1\right)\cdot e_{\widetilde{\Sigma}}\left(\sum_{j=0}^{1} e_\Sigma(n-j)\right) \\
&+\cdots \\
&+((e_\Sigma(n-i)>0))\cdot \\
&\quad\cdot\left(\sum_{j=1}^{n-i-1} e_\Sigma(j)+1\right)\cdot e_{\widetilde{\Sigma}}\left(\sum_{j=0}^{i} e_\Sigma(n-j)\right) \\
&+\cdots \\
&+((e_\Sigma(1)>0))\cdot e_{\widetilde{\Sigma}}(N) \\
\geq\;& \left(\sum_{j=1}^{n-1} e_\Sigma(j)+1\right)+\left(\sum_{j=1}^{n-2} e_\Sigma(j)+1\right)+\cdots \\
&+\left(\sum_{j=1}^{n-i-1} e_\Sigma(j)+1\right)+\cdots+1 \\
=\;& n+\sum_{i=1}^{n-1}\sum_{j=1}^{i} e_\Sigma(j) \\
=\;& n+\sum_{i=1}^{n}\sum_{j=1}^{i} e_\Sigma(j)-N \\
=\;& n+area(\Sigma)-N,
\end{aligned}$$

whence the lemma. ■

When Σ is AREA-maximizing, the inequality of Lemma 3 becomes an equality:

Lemma 4 *If Σ is AREA-maximizing and $\widetilde{\Sigma}$ is dual to Σ, then $area(\widetilde{\Sigma}) = area(\Sigma)+n-N$.*

[3] We describe other examples in Lemma 7 and Theorems 7, 8.

[4] $[[a,\ldots,b]]$ denotes an unspecified permutation of $\{a,\ldots,b\}$.

Proof Sketch. Assume, for contradiction, that Σ and $\widetilde{\Sigma}$ are as in the lemma but that

$$area(\widetilde{\Sigma}) > area(\Sigma)+n-N. \tag{6}$$

Consider a schedule for $\mathcal{G}$ that is dual to $\widetilde{\Sigma}$; call it Σ'. Since $\widetilde{\mathcal{G}}$'s sources are $\mathcal{G}$'s sinks, and vice versa, we can combine Lemma 3's inequality with (6) to infer that $area(\Sigma') \geq area(\widetilde{\Sigma})+N-n > area(\Sigma)$, which contradicts Σ's alleged AREA-maximality. ■

Reasoning almost identical to that in the proof of Lemma 4 now yields Theorem 1. Assume, for contradiction, that $\mathcal{G}$ has an AREA-maximizing schedule Σ one of whose duals, say $\widetilde{\Sigma}$, is not AREA-maximizing for $\widetilde{\mathcal{G}}$. There must then be another schedule, call it $\widetilde{\Sigma}'$, for $\widetilde{\mathcal{G}}$ such that $area(\widetilde{\Sigma}') > area(\widetilde{\Sigma})$. Focus on any schedule Σ' for $\mathcal{G}$ that is dual to $\widetilde{\Sigma}'$, and note that $area(\Sigma') \geq area(\widetilde{\Sigma}')-n+N > area(\Sigma)$. This contradicts the alleged AREA-maximality of Σ. ■

4. AREA-Maximization and MAX-Linear Arrangement

A basic question regarding any desirable property X of schedules is, "How hard is it to craft schedules that have property X?" Although history tells us that crafting schedules with most interesting properties is (likely to be) computationally hard—usually NP-Hard or worse—it is often difficult to pin down this hardness formally. This is, thus far, the case with AREA-maximization. However, we can show that this problem does relate, via a polynomial-time reduction, to a problem that is also likely to be computationally hard, yet which has also defied definitive placement in the complexity hierarchy, viz., the *MAX Linear-Arrangement Problem for Dags* (*MAX-LA*) [13]. While the reduction that yields Theorems 2 and 3 does not show definitively that AREA-maximization is computationally hard, it does suggest such hardness. Additionally, the relationship that we establish between AREA-maximization and MAX-LA gives us an avenue for efficiently finding AREA-maximizing schedules for significant classes of dags.

For dag $\mathcal{G}$ with schedule Σ and any $v \in \mathcal{N}_{\mathcal{G}}$, denote by $\lambda_\Sigma(v)$ the index that Σ assigns v in its linearization of $\mathcal{N}_{\mathcal{G}}$; e.g., if Σ linearizes $\mathcal{N}_{\mathcal{G}}$ in the order $v_1,\ldots,v_{N_{\mathcal{G}}}$, then for each k, $\lambda_\Sigma(v_k)=k$. The *linear-arrangement value* (*LA-value*) of Σ, denoted LA-VAL(Σ), is the sum LA-VAL(Σ) $\stackrel{\text{def}}{=} \sum_{(u\to v)\in\mathcal{A}_{\mathcal{G}}} [\lambda_\Sigma(v)-\lambda_\Sigma(u)]$ of the "stretches" of $\mathcal{G}$'s arcs under λ_Σ. (Since Σ is valid, $\lambda_\Sigma(v)>\lambda_\Sigma(u)$ for each $(u\to v)\in\mathcal{A}_{\mathcal{G}}$.) One can think about LA-Value as the *cumulative cutwidth* under λ_Σ: Consider two nodes, $u,v\in\mathcal{N}_{\mathcal{G}}$, that are *consecutive* under λ_Σ. The *cut formed by u and v under λ_Σ* is the set of all $(x\to y)\in\mathcal{A}_{\mathcal{G}}$ for which $\lambda_\Sigma(x)\leq\lambda_\Sigma(u)$ and $\lambda_\Sigma(v)\leq\lambda_\Sigma(y)$. The *width* of the cut is the size of this set. Easily, the sum of all cutwidths under λ_Σ is precisely LA-VAL(Σ). The *maximizing* version of LA (MAX-LA) for $\mathcal{G}$ is to find a schedule for $\mathcal{G}$ of maximum LA-Value. The MAX-LA problem for general dags resides in the class MAX-SNP [13].

Let us be given a dag $\mathcal{G}$ and a valid schedule Σ for $\mathcal{G}$; Fig. 2(left) supplies a running example. A *tree-matching* in $\mathcal{G}$ is a set $A\subseteq\mathcal{A}_{\mathcal{G}}$ such that no $v\in\mathcal{N}_{\mathcal{G}}$ is entered by more than one arc of A. $\mathcal{G}^{(A)}$ is the dag obtained from $\mathcal{G}$ by removing from $\mathcal{A}_{\mathcal{G}}$ all arcs that belong to A (but retaining all of $\mathcal{G}$'s nodes). A *tree-matched schedule* for $\mathcal{G}$ is a sequence of $m\geq 1$ tree-matchings, $A_1,\ldots,A_m$, with an associated sequence of $m+1$ successively smaller subdags of $\mathcal{G}$: $(\mathcal{G}_0=\mathcal{G})$, $(\mathcal{G}_1=\mathcal{G}_0^{(A_1)}),\ldots,$ $(\mathcal{G}_m=\mathcal{G}_{m-1}^{(A_m)})$. (The "successively smaller" requirement means that each A_i is nonempty.) These sequences must satisfy:

- For each $i\in[1,m]$, A_i is a tree-matching in $\mathcal{G}_{i-1}$ such that, for all arcs $(u\to v)\in A_i$, u is ELIGIBLE in $\mathcal{G}_{i-1}$.
- $\mathcal{G}_m$ has no arcs: $A_{\mathcal{G}_m}=\emptyset$.

Every scheduling of a dag $\mathcal{G}$ via tree-matchings gives rise to a *timed version*, $T(\mathcal{G})$, of $\mathcal{G}$; cf. Fig. 2(right). If the scheduling of $\mathcal{G}$ via tree-matchings involves the sequence $A_1,\ldots,A_m$ of tree-matchings then $T(\mathcal{G})$ has $m+1$ *tiers*, $T_0(\mathcal{G}),\ldots,T_m(\mathcal{G})$; in Fig. 2(right), $m=3$. $T(\mathcal{G})$ has the following structure.

- Each tier has a copy of $\mathcal{G}$'s nodes.
- $T(\mathcal{G})$ has *intra-tier* arcs that play the roles of $\mathcal{G}$'s arcs. The arcs within tier $i\in[1,m]$ are copies of the arcs from tree-matching A_i instantiated within $T_i(\mathcal{G})$'s copy of $\mathcal{G}$'s nodes. Intra-tier arcs appear as solid arcs in Fig. 2(right). Tier 0, which serves only to simplify our analysis later, has no intra-tier arcs.
- $T(\mathcal{G})$ has *inter-tier* arcs that connect the copy of each node $v\in\mathcal{N}_{\mathcal{G}}$ in tier i to the copy of v in tier $i+1$, for $i\in[0,m-1]$; these arcs appear as dashed arcs in Fig. 2(right).

Note that each tier $T_i(\mathcal{G})$ with $i>0$ has at least one intra-tier arc because $A_i\neq\emptyset$; hence, $m\leq 1+\max_{v\in\mathcal{N}_{\mathcal{G}}}(\textit{indegree}(v))$.

Schedule Σ for $\mathcal{G}$ leads naturally to a schedule $\widehat{\Sigma}$ for $T(\mathcal{G})$: $\widehat{\Sigma}$ schedules $T(\mathcal{G})$ tier by tier, honoring Σ within each tier; hence, Σ's associated linearization λ_Σ leads naturally to the linearization $\lambda_{\widehat{\Sigma}}$ of $T(\mathcal{G})$. See Fig. 3.

Lemma 5 *Let Σ be a schedule for the dag $\mathcal{G}$, and let $T(\mathcal{G})$ be an m-tier timed version of $\mathcal{G}$. Then LA-VAL($\lambda_{\widehat{\Sigma}}$) = LA-VAL($\lambda_\Sigma$) + $mN_{\mathcal{G}}^2$.*

Proof. Focus on $\mathcal{G}$ and $T(\mathcal{G})$ (cf. Fig. 2(right)) and on schedules Σ and $\widehat{\Sigma}$, as represented by λ_Σ and $\lambda_{\widehat{\Sigma}}$ (cf. Fig. 3). $T(\mathcal{G})$ has $mN_{\mathcal{G}}$ dashed arcs. Under $\lambda_{\widehat{\Sigma}}$, each dashed arc (u,v) has "stretch" $N_{\mathcal{G}}$; i.e., $\lambda_{\widehat{\Sigma}}(v)-\lambda_{\widehat{\Sigma}}(u)=N_{\mathcal{G}}$.

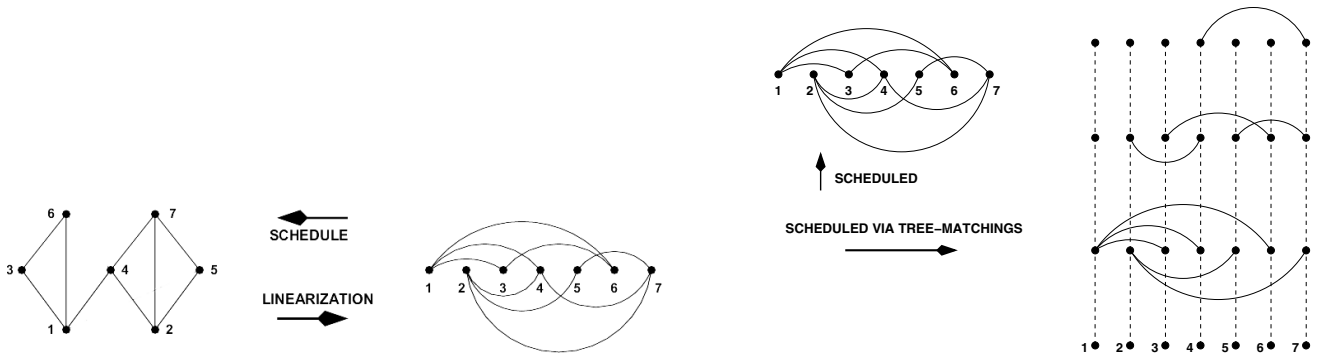

Figure 2. (Left) *A dag $\mathcal{G}$ (arcs go upward) scheduled via Σ: node-execution order and λ_Σ (arcs go rightward.)* (Right) *$\mathcal{G}$ and an associated tiered version $T(\mathcal{G})$ produced via tree-matchings.*

Hence, the dashed arcs cumulatively contribute $mN_{\mathcal{G}}^2$ to LA-VAL$(\lambda_{\widehat{\Sigma}})$. Each solid arc of $T(\mathcal{G})$ has the same "stretch" under λ_Σ as under $\lambda_{\widehat{\Sigma}}$, so they cumulatively contribute LA-VAL(λ_Σ) to LA-VAL$(\lambda_{\widehat{\Sigma}})$. ■

The message is: *Solving MAX-LA for $\mathcal{G}$ is equivalent to solving MAX-LA for $T(\mathcal{G})$.*

Theorem 2 *Let Σ be a schedule for a dag $\mathcal{G}$, and let $T(\mathcal{G})$ be an $(m+1)$-tier timed version of $\mathcal{G}$. Then*

$$AREA(\widehat{\Sigma}) = \textit{LA-VAL}(\lambda_\Sigma) + (mN_{\mathcal{G}} - A_{\mathcal{G}})N_{\mathcal{G}} + \binom{N_{\mathcal{G}}+1}{2}.$$

Proof. Each node of $T(\mathcal{G})$ has indegree 0, 1, or 2:

- Nodes on tier 0 of $T(\mathcal{G})$ are the only nodes having indegree 0.
- Each node v on a tier $i>0$ of $T(\mathcal{G})$ has a single dashed, inter-tier, arc entering it. Additionally, v has at most one solid, intra-tier, arc entering it. This solid arc is "inherited" from an arc of $\mathcal{G}$ that enters the node of $\mathcal{G}$ of which v is a copy. If this solid arc exists, then v has indegree 2; otherwise, v has indegree 1.

We can precisely assess the contribution of each category of nodes to $AREA(\widehat{\Sigma})$.

Nodes of indegree 0 are ELIGIBLE for the same number of steps under schedules Σ for $\mathcal{G}$ and $\widehat{\Sigma}$ for $T(\mathcal{G})$.

Nodes of indegree 1*:* each such node v becomes ELIGIBLE as soon as its single parent is executed. By construction, v's single incoming arc is dashed, so v remains ELIGIBLE for exactly $N_{\mathcal{G}}$ steps, since all dashed arcs share this length.

Nodes of indegree 2*:* Each such node v becomes ELIGIBLE as soon as its parent along the solid incoming arc is executed—because v's incoming dashed arc comes from the preceding tier of $T(\mathcal{G})$, while its incoming solid arc comes from the present tier. Thus, the duration of v's ELIGIBILITY is exactly the amount that v's incoming solid arc contributes to LA-VAL(λ_Σ), exactly as in the solid-arc analysis of indegree-1 nodes.

Summing up, the cumulative number of steps when nodes of $T(\mathcal{G})$ are ELIGIBLE $(= AREA(\widehat{\Sigma}))$ exceeds LA-VAL(λ_Σ) by an amount C that depends on the *size* of $\mathcal{G}$ but not on its linearization. Specifically, C equals the cumulative length of some of $T(\mathcal{G})$'s dashed arcs: the $N_{\mathcal{G}}$ arcs from tier 0 to tier 1, plus the contributions of the $mN_{\mathcal{G}} - A_{\mathcal{G}}$ arcs that do not share a target node with any arc of $\mathcal{G}$. Because (cf. the proof of Lemma 5) each of $T(\mathcal{G})$'s dashed arcs has "stretch" $N_{\mathcal{G}}$, we have $C = \binom{N_{\mathcal{G}}+1}{2} + (mN_{\mathcal{G}} - A_{\mathcal{G}})N_{\mathcal{G}}$. ■

Theorem 2's polynomial-time mapping of the Area-maximization problem to MAX-LA yields the main result of this section.

Theorem 3 *In the worst case, Area maximization is no easier than MAX-LA for dags.*

5. Area Maximization for Monotonic Tree-Dags

A tree-dag is *monotonic* if it is either an *out-tree* (a single-source dag in which each nonsource has a single parent) or an *in-tree* (the dual of an out-tree). See[5] Fig. 4(left); by replacing each edge with an arc that points upward (resp., downward), the trees in the figure become out-trees (resp., in-trees).

5.1. MAX-LA Is (Almost) MAX-AREA for Out-Trees

For out-trees, every Area-maximizing schedule is an LA-Value-maximizing schedule, and conversely. (This is clearly not true for dags that are not trees.)

[5] To simplify subsequent drawings, edges represent upward arcs.

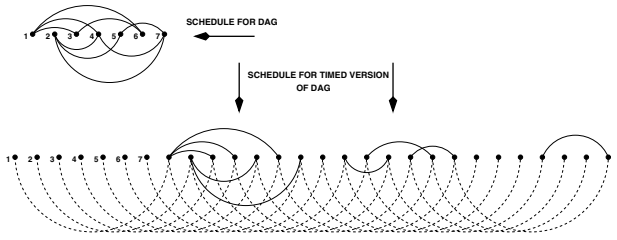

Figure 3. *$\mathcal{G}$ under schedule Σ and $T(\mathcal{G})$ under schedule $\widehat{\Sigma}$; arcs go rightward.*

Lemma 6 *For any out-tree $\mathcal{T}$ and associated schedule Σ, $AREA(\Sigma) = LA\text{-}VAL(\Sigma) + 1$. Thus, every AREA-maximizing schedule maximizes LA-Value, and conversely.*

Proof. Consider an arbitrary out-tree $\mathcal{T}$ and an arbitrary schedule Σ for $\mathcal{T}$. Because each nonsource of $\mathcal{T}$ has indegree 1, the outdegree of each $v \in \mathcal{N}_{\mathcal{T}}$ is the number of nodes that become ELIGIBLE when v is executed. This means that the cutwidth under λ_Σ between v and its successor is the total number of nodes that are ELIGIBLE after v and all of its predecessors under Σ are executed. Thus, $AREA(\Sigma) - 1$ (the "-1" compensates for the unique source that is ELIGIBLE at time 0) equals the cumulative cutwidth of $\mathcal{T}$ under Σ, which equals the cumulative "stretch" of $\mathcal{T}$'s arcs under Σ, which equals LA-VAL(Σ). Thus, for out-trees, MAX-LA is the same as AREA-maximization. ■

5.2. Maximizing Area for Monotonic Tree-Dags

Concentrate first on out-trees. Many out-trees admit IC-optimal schedules, including those in which each node's outdegree is no larger than its parent's [12]; see the lefthand tree of Fig. 4(left). Some out-trees that violate this condition do not admit any IC-optimal schedule, as exemplified by the righthand tree of Fig. 4(left).

Lemma 7 *The righthand out-tree $\mathcal{T}$ of Fig. 4(left) does not admit an IC-optimal schedule.*

Proof Sketch. The unique optimal first two steps of executing $\mathcal{T}$ execute node A, then C; the unique optimal first three steps execute, in order, nodes A, B, D. ■

Lemma 8 *One can find an Area-maximizing schedule for any n-node out-tree in time $O(n \log n)$.*

Proof Sketch. An algorithm in [1] produces a MIN Linear Arrangement of any n-node out-tree in time $O(n \log n)$. The algorithm produces a MAX-LA linearization if it uniformly substitutes maximizations for minimizations. One then invokes Lemma 6. ■

In-trees and out-trees are mutually dual; hence, Theorem 1 extends Lemma 8 to in-trees.

Theorem 4 *One can find an Area-maximizing schedule for any n-node monotonic tree-dag in time $O(n \log n)$.*

6. Compositions of Cliques and Cycles

We now develop hints for crafting Area-maximizing schedules for computationally significant families of dags that, in general, do not admit IC-optimal schedules. The families of interest are built by *composing* bipartite cliques and cycles in complex ways. *The technical tools introduced here can be adapted to many other families of dags.*

Building complex dags via *composition*. The operation of *composition* is a fundamental tool for building complex computation-dags from simpler ones.

- Start with a set $\mathcal{B}$ of base dags. (The base dags in [3, 12] are *connected* and *bipartite*.)
- One composes $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{B}$—which could be the same dag with nodes renamed to achieve disjointness—to obtain a composite dag $\mathcal{G}$, as follows. (a) Let $\mathcal{G}$ begin as the sum, $\mathcal{G}_1 + \mathcal{G}_2$, of $\mathcal{G}_1, \mathcal{G}_2$. Rename nodes to ensure that $\mathcal{N}_{\mathcal{G}} \cap (\mathcal{N}_{\mathcal{G}_1} \cup \mathcal{N}_{\mathcal{G}_2}) = \emptyset$. ($b$) 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 $\mathcal{G}_2$. (c) Pairwise identify (i.e., merge) the nodes of S_1 and S_2 in some way. The resulting node-set is $\mathcal{N}_{\mathcal{G}}$; the induced set of arcs is $\mathcal{A}_{\mathcal{G}}$.[6]
- 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* $[\mathcal{G}_1 \Uparrow \mathcal{G}_2]$. Easily, the composition operation is associative.

6.1. Clique- and Cycle-Based Building Blocks

We use the following families of bipartite dags—which appear often as subdags of real computation-dags—to illustrate the ideas we develop in this section; see Fig. 5.

Bipartite cliques. For integers $s > 1$, the *s-source (bipartite) clique* $\mathcal{Q}_s$ has s sources and s sinks, and an arc from each source to each sink. Of special interest is $\mathcal{Q}_2$, the *butterfly dag,* which is the basic building block of the well-known FFT [6] and butterfly dags.

Bipartite cycles. For integers $s > 1$, the *s-source (bipartite) cycle* $\mathcal{C}_s$ has s sources, $u_0, \ldots, u_{s-1}$ and s sinks,

[6] Arc $(u \to v)$ is *induced* if $\{u, v\} \subseteq \mathcal{N}_{\mathcal{G}}$.

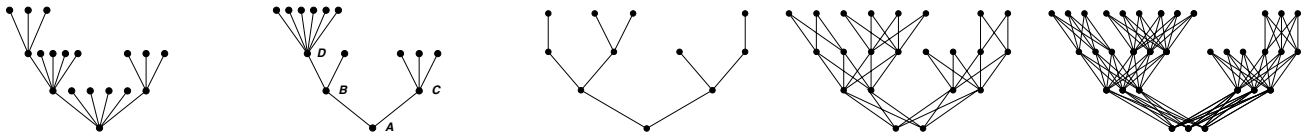

Figure 4. (Left) *Two trees.* (Right) *An out-tree, its* 2*-thickened version, its* 3*-thickened version.*

$v_0,\ldots,v_{s-1}$. $\mathcal{C}_s$'s arcs connect each source u_i to sinks v_i and $v_{(i+1 \bmod s)}$. $\mathcal{C}_2 = \mathcal{Q}_2$ is a building block of the FFT dag; larger cycles are building blocks of dags such as the data-dependency dag of the recursive matrix-multiplication algorithm [4].

6.2. A Simple Scheduling Principle

Let $\mathcal{G}$ be a dag and Σ a schedule for $\mathcal{G}$. Focus on a step t in Σ's execution of $\mathcal{G}$ and on a $v \in \mathcal{N}_{\mathcal{G}}$ that is ELIGIBLE at step t. The *yield* of v at step t, denoted $Yld_{\Sigma}(v;t)$, is the number of nodes that would be rendered ELIGIBLE if Σ were to execute v at that step.

The strict demands of IC-optimality—its "at every step" demand—means that, at every step in executing $\mathcal{G}$, a schedule must execute a node of maximum yield. (So the entire challenge for an IC-optimal schedule is to break ties wisely.) Of course, AREA-maximizing schedules may violate this principle; in fact, an AREA-maximizing schedule for the righthand out-tree of Fig. 4(left) *must* violate this principle. However, there is a weak version of the principle that does hold for AREA-maximizing schedules.

For each $x \in \mathcal{N}_{\mathcal{G}}$, denote by $T_{\Sigma}(x)$ the step at which Σ executes x

Lemma 9 *Let $\mathcal{G}$ be a dag and Σ a schedule for $\mathcal{G}$. Say that at step t in Σ's execution of $\mathcal{G}$, there exist two nodes, u and v, that do not share any children, such that: both u and v are* ELIGIBLE *at step t,* and $[Yld_{\Sigma}(u;t) > Yld_{\Sigma}(v;t)]$. *If Σ now executes v and u consecutively, in that order—i.e., $T_{\Sigma}(v) = t$ and $T_{\Sigma}(u) = t+1$—then Σ is* not *an AREA-maximizing schedule.*

Proof. Let Σ be as described, and assume, for contradiction, that Σ *is* an AREA-maximizing schedule. We derive a contradiction by comparing Σ's eligibility profile against that of the schedule Σ' for $\mathcal{G}$ that is identical to Σ, except that it reverses the order of executing nodes u and v.

1. For all steps $t' < t$, Σ and Σ' behave identically, so that $E_{\Sigma}(t') = E_{\Sigma'}(t')$. (Note that neither u nor v has been executed during this period.)

2. Since u and v are independent in the sense of the Lemma, we have

$$\begin{array}{rcccl} Yld_{\Sigma}(u;t) & = & Yld_{\Sigma}(u;t+1) & = & Yld_{\Sigma'}(u;t) \\ Yld_{\Sigma'}(v;t) & = & Yld_{\Sigma'}(v;t+1) & = & Yld_{\Sigma}(v;t). \end{array}$$

   Therefore:

$$\begin{array}{rcl} E_{\Sigma}(t) & = & E_{\Sigma}(t-1) + Yld_{\Sigma}(v;t) \\ E_{\Sigma}(t+1) & = & E_{\Sigma}(t) + Yld_{\Sigma}(u;t+1) \\ & = & E_{\Sigma}(t-1) + Yld_{\Sigma}(v;t) + Yld_{\Sigma}(u;t) \\ E_{\Sigma'}(t) & = & E_{\Sigma}(t-1) + Yld_{\Sigma}(u;t) \\ E_{\Sigma'}(t+1) & = & E_{\Sigma'}(t) + Yld_{\Sigma'}(v;t+1) \\ & = & E_{\Sigma}(t-1) + Yld_{\Sigma}(v;t) + Yld_{\Sigma}(u;t) \end{array}$$

   Hence, $E_{\Sigma'}(t) > E_{\Sigma}(t)$, while $E_{\Sigma'}(t+1) = E_{\Sigma}(t+1)$.

3. For all steps $t' > t+1$, Σ and Σ' behave identically, so that $E_{\Sigma}(t') = E_{\Sigma'}(t')$. (Note that both u and v have been executed before this period begins.)

This reckoning shows that $AREA(\Sigma') > AREA(\Sigma)$, which contradicts Σ's alleged AREA-maximality. ■

Lemma 9 has not-quite-obvious implications for scheduling cycles and cliques.

Lemma 10 *Let $\mathcal{G}$ be a composition of cycles and Σ an AREA-maximizing schedule for $\mathcal{G}$. Say that at step t in Σ's execution of $\mathcal{G}$, some constituent cycle $\mathcal{C}$ of $\mathcal{G}$ has:* (1) *at least one* EXECUTED *source,* (2) *at least one* ELIGIBLE *source, and* (3) *all other sources either* EXECUTED *or* ELIGIBLE. *Then at step $t+1$, Σ will execute a source of $\mathcal{C}$.*

Proof. The lemma depends on the following obvious fact about the structure of cycles. No matter how one executes the sources of a cycle $\mathcal{C}$, the Yield of the first-executed source is zero. If one then executes $\mathcal{C}$'s sources in cyclic order from the first-executed source, then the sequence of Yields of $\mathcal{C}$'s remaining sources is $1,1,\ldots,2$. Assume, therefore, that Σ finds itself with a cycle $\mathcal{C}$ as described in the lemma. Once having executed a source of $\mathcal{C}$, Lemma 9 assures us that Σ will not execute any source of any other cycles until it has completed executing all sources of $\mathcal{C}$. ■

We can strengthen Lemma 10 for cliques—even ones having different numbers of sources and sinks.

Lemma 11 *Let the dag $\mathcal{G}$ have a clique $\mathcal{Q}$ as a subdag. Any Area-maximizing schedule Σ for $\mathcal{G}$ executes all sources of $\mathcal{Q}$ in consecutive steps.*

Proof. Let $\mathcal{G}$ and $\mathcal{Q}$ be as in the lemma. Focus on a schedule Σ for $\mathcal{G}$ that does not behave in the prescribed manner. We show that Σ is not Area-maximizing.

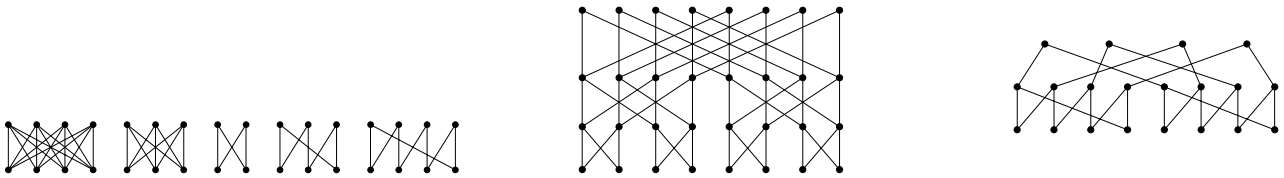

Figure 5. (Left) *Left to right:* $\mathcal{Q}_4$, $\mathcal{Q}_3$, $(\mathcal{Q}_2 = \mathcal{C}_2)$, $\mathcal{C}_3$, $\mathcal{C}_4$. (Center) $\mathcal{Q}_2 = \mathcal{C}_2$ *as a building block of the FFT dag.* (Right) $\mathcal{C}_4$ *as a building block of the recursive matrix-multiplication dag.*

No matter how Σ executes the sources of $\mathcal{Q}$, each executed source save the last—call it ω—has *zero* Yield; the Yield of ω is the number of sinks of $\mathcal{Q}$. Because Σ does not execute $\mathcal{Q}$'s sources consecutively, there exist $x, y \in \mathcal{N}_{\mathcal{G}}$ such that:

- x is a source of $\mathcal{Q}$ that is distinct from ω; y is not a source of $\mathcal{Q}$.
- $T_\Sigma(x) = T_\Sigma(y) - 1$; i.e., Σ executes x one step before y. In other words, Σ interrupts its execution of $\mathcal{Q}$ after executing x, in order to execute y.

Let $z \in \mathcal{N}_{\mathcal{G}}$ be the first positive-Yield node that Σ executes after x; i.e., executing z renders ≥ 1 new nodes of $\mathcal{G}$ ELIGIBLE. (z could be y.) Consider two cases.

Case 1: $z \neq \omega$. Focus on the schedule Σ' that is identical to Σ, except that:

- Σ' accelerates by one step all node-executions that Σ performs from step $T_\Sigma(x)+1$ to step $T_\Sigma(z)$ (inclusive); i.e.,

$$(\forall v)\Big[[T_\Sigma(x) < T_\Sigma(v) \leq T_\Sigma(z)] \Rightarrow [T_{\Sigma'}(v) = T_\Sigma(v) - 1]\Big].$$

- Σ' executes x at step $T_\Sigma(z)$: $T_{\Sigma'}(x) = T_\Sigma(z)$.

We compare the areas of Σ and Σ', by comparing their profiles. We claim that $E_\Sigma(T_\Sigma(z)-1) < E_{\Sigma'}(T_\Sigma(z)-1)$, while $E_\Sigma(t) = E_{\Sigma'}(t)$ for all other $t \in [0, N_{\mathcal{G}}]$. These assertions are justified as follows.

- By definition, Σ and Σ' behave identically both before Σ executes x and after Σ executes z.
- No new node is rendered ELIGIBLE under either schedule from step $T_\Sigma(x) - 1 = T_{\Sigma'}(x) - 1$ until node z is executed. z is executed at step $T_\Sigma(z)$ under Σ and at step $T_{\Sigma'}(z) = T_\Sigma(z) - 1$ under Σ'.

By (1), we thus have $Area(\Sigma) < Area(\Sigma')$, so that Σ is not Area-maximizing.

Case 2: $z = \omega$. Repeat the argument of Case 1 with y as the node whose execution is deferred until after z's. Since $y \notin \mathcal{N}_{\mathcal{Q}}$, the adapted details are easy. ■

For some compositions of cliques, notably including FFT dags, Lemma 11's necessary condition is also sufficient; see Lemma 2(b) and [15].

6.3. Applications of the Simple Principle

A. *Thickened* dags. Let $\mathcal{G}$ be a dag and k an integer. The *k-thickening* of $\mathcal{G}$ is obtained by replacing each $v \in \mathcal{N}_{\mathcal{G}}$ with k nodes, $v^{(1)}, \ldots, v^{(k)}$, and replacing each $(x,y) \in \mathcal{A}_{\mathcal{G}}$ with a copy of $\mathcal{Q}_k$ that has $x^{(1)}, \ldots, x^{(k)}$ as sources and $y^{(1)}, \ldots, y^{(k)}$ as sinks. Fig. 4(right) instantiates this definition for an out-tree $\mathcal{G}$.

Lemma 12 *Let $\mathcal{G}$ be a dag, Σ an AREA-maximizing schedule for $\mathcal{G}$, and $\mathcal{G}^{(k)}$ the k-thickening of $\mathcal{G}$. The schedule $\Sigma^{(k)}$ that operates as follows is an AREA-maximizing schedule for $\mathcal{G}^{(k)}$. $\Sigma^{(k)}$ processes each of $\mathcal{G}^{(k)}$'s embedded copies of $\mathcal{Q}_k$ in the same order as Σ executes the corresponding nodes of $\mathcal{G}$; $\Sigma^{(k)}$ processes each copy of $\mathcal{Q}_k$ by executing its sources consecutively.*

Proof. Lemma 11 tells us that $\Sigma^{(k)}$ must execute the sources of each embedded copy of $\mathcal{Q}_k$ consecutively. If $\Sigma^{(k)}$ processes each such copy in the indicated order, then the number of ELIGIBLE nodes on $\mathcal{G}^{(k)}$ under $\Sigma^{(k)}$ will follow a pattern: After executing the last source of some copy of $\mathcal{Q}_k$, the number of ELIGIBLE nodes on $\mathcal{G}^{(k)}$ will be exactly k times the corresponding number for $\mathcal{G}$ under Σ. This number will decrease by 1 at each step, as $\Sigma^{(k)}$ executes the sources of its next chosen copy of $\mathcal{Q}_k$, resuming its "exactly k times" status as soon as this copy's sources are all executed.

Say, for contradiction, that some schedule, Σ', for $\mathcal{G}^{(k)}$ has greater AREA than $\Sigma^{(k)}$. Since all embedded cliques used to "thicken" $\mathcal{G}$ have the same number of sources, there would then be steps t at which $E_{\Sigma'}(t) > E_{\Sigma^{(k)}}(t)$. At each such step, Σ' would have to have ELIGIBLE nodes from more embedded copies of $\mathcal{Q}_k$ than $\Sigma^{(k)}$ does—which means that Σ' would be identifying (via the correspondence between arcs of $\mathcal{G}$ and cliques in $\mathcal{G}^{(k)}$) more ELIGIBLE nodes on $\mathcal{G}$ than Σ produces. Σ' would thus lead us to a

schedule for $\mathcal{G}$ whose AREA exceeds Σ's, contradicting Σ's AREA-maximality. ■

B. Compositions of cycles and of cliques.

AREA-based scheduling priorities. The algorithmic setting of [12] schedules a dag $\mathcal{G}$ by decomposing it into *building blocks* and prioritizing the execution of these constituent subdags. The priority relation $\triangleright$ that enables this scheduling strategy formalizes the following intuitive notion. For two building blocks $\mathcal{G}_1$ and $\mathcal{G}_2$, the assertion "$\mathcal{G}_1 \triangleright \mathcal{G}_2$" means that *one can only increase the IC-quality of a schedule Σ by having Σ execute all of $\mathcal{G}_1$'s nonsinks before any of $\mathcal{G}_2$'s.* $\triangleright$-priority is formalized and used as follows in [12]. The dag $\mathcal{G}$ is a $\triangleright$*-linear composition* of subdags $\mathcal{G}_1,\ldots,\mathcal{G}_n$ if: (a) $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \cdots \Uparrow \mathcal{G}_n$; (b) $\mathcal{G}_i \triangleright \mathcal{G}_{i+1}$, for all $i \in [1, n-1]$. This means that there is a topological sort of $\mathcal{G}$ in which: (1) each $\mathcal{G}_i$'s nodes appear consecutively; (2) the $\mathcal{G}_i$ appear in an order consistent with their $\triangleright$-priorities.

Theorem 5 ([12]) *If $\mathcal{G}$ is a $\triangleright$-linear composition of $\mathcal{G}_1,\ldots,\mathcal{G}_n$, where each $\mathcal{G}_i$ admits an IC-optimal schedule Σ_i, then 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 nonsinks of $\mathcal{G}_i$, in the order mandated by Σ_i.*
2. *Σ finally executes all sinks of $\mathcal{G}$ in any order.*

We introduce for AREA maximization an analogue of both $\triangleright$-priority and Theorem 5.

Dag $\mathcal{G}_1$ *has AREA-priority over* dag $\mathcal{G}_2$, denoted $\mathcal{G}_1 \overset{\text{A}}{\triangleright} \mathcal{G}_2$, if any schedule Σ for the sum $\mathcal{G}_1 + \mathcal{G}_2$ that operates as follows is AREA-maximizing.

1. Σ executes all nonsinks of $\mathcal{G}_1$ in the order mandated by an AREA-maximizing schedule for $\mathcal{G}_1$.
2. Σ executes all nonsinks of $\mathcal{G}_2$ in the order mandated by an AREA-maximizing schedule for $\mathcal{G}_2$.
3. Σ executes all sinks of $\mathcal{G}_1 + \mathcal{G}_2$ in any order.

Definition 1 *A dag $\mathcal{G}$ is a* $\overset{\text{A}}{\triangleright}$-linear composition *of subdags $\mathcal{G}_1,\ldots,\mathcal{G}_n$ if: (a) $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \cdots \Uparrow \mathcal{G}_n$; (b) $\mathcal{G}_i \overset{\text{A}}{\triangleright} \mathcal{G}_j$, for all $1 \leq i < j \leq n$.*

Theorem 6 *Let $\mathcal{G}$ be a $\overset{\text{A}}{\triangleright}$-linear composition of $\mathcal{G}_1,\ldots,\mathcal{G}_n$. The schedule Σ for $\mathcal{G}$ that proceeds as follows is AREA-maximizing.*

1. *For $i = 1,\ldots,n$, in turn, Σ executes the nodes of $\mathcal{G}$ that correspond to nonsinks of $\mathcal{G}_i$, in the order mandated by some AREA-maximizing schedule for $\mathcal{G}_i$.*
2. *Σ finally executes all sinks of $\mathcal{G}$ in any order.*

Theorem 6 is immediate from the definition of $\overset{\text{A}}{\triangleright}$, which implies that one cannot improve Σ's AREA by interleaving the execution of the nonsinks of any collection of $\mathcal{G}_i$. We see now that Theorem 6 leads us to AREA-maximizing schedules for compositions of bipartite cliques and compositions of bipartite cycles.

$\overset{\text{A}}{\triangleright}$-priorities among bipartite cliques. Sums of cliques generally do not admit any IC-optimal schedule; in particular, $\mathcal{Q}_s + \mathcal{Q}_{s'}$ admits an IC-optimal schedule *only when* $s = s'$ [12]. In contrast, it is easy to find Area-maximizing schedules for an arbitrary sum of cliques $\mathcal{G}$—*even those having differing numbers of sources and sinks*. Lemmas 1 and 11 allow us to consider only schedules that (a) execute $\mathcal{G}$'s sinks only after all of its sources and (b) execute $\mathcal{G}$'s sources clique by clique.

Theorem 7 *A schedule Σ for a sum of cliques $\mathcal{G} = \mathcal{Q}_{s_1} + \cdots + \mathcal{Q}_{s_m}$ is Area-maximizing if, and only if it executes $\mathcal{G}$ clique by clique, in an arbitrary order. Hence, every pair of bipartite cliques are $\overset{\text{A}}{\triangleright}$-"equivalent": we always have* both $[\mathcal{Q}_{s_1} \overset{\text{A}}{\triangleright} \mathcal{Q}_{s_2}]$ *and* $[\mathcal{Q}_{s_2} \overset{\text{A}}{\triangleright} \mathcal{Q}_{s_1}]$.

Proof. Focus on a schedule Σ that honors Lemmas 1 and 11. To simplify notation, say that Σ executes the cliques in the order $\mathcal{Q}_{s_1}, \ldots, \mathcal{Q}_{s_m}$. Note that the quantities $E_\Sigma(t)$ change with increasing t according to a simple pattern. As the sources of each clique $\mathcal{Q}_{s_i}$ are executed, this clique contributes the following numbers of ELIGIBLE nodes to the Area tally:

- s_i nodes during the $s_1 + \cdots + s_{i-1}$ steps before any sources of $\mathcal{Q}_{s_i}$ have been executed;
- for each $j \in [1, s_i - 1]$: $s_i - j$ nodes after precisely j sources of $\mathcal{Q}_{s_i}$ have been executed;
- s_i nodes during the $1 + s_{i+1} + \cdots + s_m$ steps after all sources of $\mathcal{Q}_{s_i}$ have been executed.

In aggregate, $\mathcal{Q}_{s_i}$ contributes the following quantity to $Area(\Sigma)$:

$$Area_i(\Sigma) \stackrel{\text{def}}{=} s_i + s_i(s_1 + \cdots + s_{i-1}) + \binom{s_i}{2} + s_i(1 + s_{i+1} + \cdots + s_m). \quad (7)$$

Because every permutation is a product of transpositions, Σ is Area-maximizing if, and only if, one cannot increase its Area by transposing the execution order of any two summand cliques of $\mathcal{G}$. Let us compare the Area of Σ with that of some schedule obtained from Σ by switching the

execution orders of some $\mathcal{Q}_{s_k}$ and $\mathcal{Q}_{s_\ell}$ for $k<\ell$. To enhance legibility, let us henceforth rename Σ as $\Sigma_{k,\ell}$, and let us call its competitor $\Sigma_{\ell,k}$. The system of equations obtained from instantiating (7) for $i=1,2,\ldots,m$ tells us that both $Area(\Sigma_{k,\ell})$ and $Area(\Sigma_{\ell,k})$ will contain the following quantity (which is written strangely for emphasis).

$$(s_1+\cdots+s_m) + \left(\sum_{i=1}^{m}\sum_{j<i} s_i s_j\right) + \left(\sum_{i=1}^{m}\binom{s_i}{2}\right) + (s_1+\cdots+s_m).$$

The four parenthesized sums come, respectively, from the moment before any sources have been executed, the periods when all of each clique's sources are ELIGIBLE since none has been executed, the periods when each clique's sources are being executed, and the moments when each clique's last source has just been executed. The differences between $Area(\Sigma_{k,\ell})$ and $Area(\Sigma_{\ell,k})$ arise, thus, only from the contributions of each clique $\mathcal{Q}_{s_i}$ *after all of its sources have been executed*. These contributions come in groups depending on the relative sizes of i, k, and ℓ.

1. If either $i<k$ or $i>\ell$, then $\mathcal{Q}_{s_i}$ contributes $s_i(s_{i+1}+\cdots+s_m)$ to both $Area(\Sigma_{k,\ell})$ and $Area(\Sigma_{\ell,k})$.
2. If $k<i<\ell$, then $\mathcal{Q}_{s_i}$ contributes $s_i(s_{i+1}+\cdots+s_{\ell-1}+s_\ell+s_{\ell+1}+\cdots+s_m)$ to $Area(\Sigma_{k,\ell})$, but it contributes $s_i(s_{i+1}+\cdots+s_{\ell-1}+s_k+s_{\ell+1}+\cdots+s_m)$ to $Area(\Sigma_{\ell,k})$.
3. $\mathcal{Q}_{s_k}$ contributes $s_k(s_{k+1}+\cdots+s_m)$ to $Area(\Sigma_{k,\ell})$, but it contributes $s_k(s_{\ell+1}+\cdots+s_m)$ to $Area(\Sigma_{\ell,k})$.
4. $\mathcal{Q}_{s_\ell}$ contributes $s_\ell(s_{\ell+1}+\cdots+s_m)$ to $Area(\Sigma_{k,\ell})$, but it contributes $s_\ell(s_{k+1}+\cdots+s_{\ell-1}+s_k+s_{\ell+1}+\cdots+s_m)$ to $Area(\Sigma_{\ell,k})$.

The preceding analysis tells us that the cliques of various categories make the following contributions to the difference $Area(\Sigma_{k,\ell})-Area(\Sigma_{\ell,k})$. The cliques of category #1 do not contribute; the cliques of category #2 cumulatively contribute $(s_\ell-s_k)(s_{k+1}+\cdots+s_{\ell-1})$; clique $\mathcal{Q}_{s_k}$ contributes $s_k(s_{k+1}+\cdots+s_\ell)$; clique $\mathcal{Q}_{s_\ell}$ contributes $(-s_\ell)(s_k+\cdots+s_{\ell-1})$. In aggregate, then, $Area(\Sigma_{k,\ell})-Area(\Sigma_{\ell,k}) = 0$, whence the theorem. ■

$\overset{\text{A}}{\triangleright}$-priorities among bipartite cycles.

Theorem 8 *A schedule Σ for a sum of cycles $\mathcal{G}=\mathcal{C}_{s_0}+\cdots+\mathcal{C}_{s_m}$ is Area-maximizing if, and only if, Σ executes all of $\mathcal{G}$'s sources cycle by cycle and then executes $\mathcal{G}$'s sinks in arbitrary order. Hence, every pair of bipartite cycles are $\overset{\text{A}}{\triangleright}$-"equivalent": we always have* both $[\mathcal{C}_{s_1}\overset{\text{A}}{\triangleright}\mathcal{C}_{s_2}]$ *and* $[\mathcal{C}_{s_2}\overset{\text{A}}{\triangleright}\mathcal{C}_{s_1}]$.

Proof. Lemma 1 tells us to execute all sinks after all sources; Lemma 10 tell us to execute $\mathcal{G}$'s sources cycle by cycle, because every sum of cycles is a (degenerate) composition of cycles. We concentrate, therefore, on proving the sufficiency of the theorem's conditions. To this end, focus on a schedule Σ that executes $\mathcal{G}$'s sources cycle by cycle. Easily, Σ's eligibility profile has the following entries.

- Just before executing the first source of each cycle—which includes the times after executing the last source of each cycle—there are $s_0+\cdots+s_m$ ELIGIBLE nodes on $\mathcal{G}$.

  An unexecuted cycle $\mathcal{C}_{s_i}$ contributes s_i ELIGIBLE sources to the sum; an executed cycle $\mathcal{C}_{s_j}$ contributes s_j ELIGIBLE sinks to the sum. The latter assertion is true because executing the last source of a cycle renders two new nodes ELIGIBLE.

- Just after executing the first source of each cycle, there are $s_0+\cdots+s_m-1$ ELIGIBLE nodes on $\mathcal{G}$.

  This holds because executing the first source of a cycle does not render any new node ELIGIBLE.

- Just after executing each source of each cycle *except for the first and the last*, there are $s_0+\cdots+s_m-1$ ELIGIBLE nodes on $\mathcal{G}$.

  This holds because executing each of these sources renders one new node ELIGIBLE.

The entire execution profile of Σ thus consists of $m+1$ occurrences of $(s_0+\cdots+s_m)$, and $s_0+\cdots+s_m-m$ occurrences of $(s_0+\cdots+s_m-1)$, so that

$$Area(\Sigma) = (s_0+\cdots+s_m)^2+m,$$

Being symmetric in the s_i, $Area(\Sigma)$ is, thus, independent of the order of executing the (sources of) $\mathcal{G}$'s constituent cycles. ■

AREA-maximizing schedules for compositions of cliques and cycles.

Theorem 9 *Let $\mathcal{G}$ be composite of type $\mathcal{G}_1\Uparrow\cdots\Uparrow\mathcal{G}_n$, where, ambiguously, the $\mathcal{G}_i$ are either all bipartite cliques or all bipartite cycles. In either case, $\mathcal{G}$ is a $\overset{\text{A}}{\triangleright}$-linear composition: $\mathcal{G}_1\overset{\text{A}}{\triangleright}\cdots\overset{\text{A}}{\triangleright}\mathcal{G}_n$. Hence, any schedule for $\mathcal{G}$ that executes the $\mathcal{G}_i$ one at a time, in any order, is AREA-maximizing.*

Proof Sketch. By Theorems 7, 8, we have $\mathcal{G}_i\overset{\text{A}}{\triangleright}\mathcal{G}_j$ for all $i,j\in[1,n]$. ■

7. Conclusions

This paper introduces a new metric, AREA-maximization, for evaluating the quality of dag-schedules for IC-platforms. As a new branch of IC-Scheduling theory, this metric represents a novel paradigm for scheduling computational jobs having inter-task dependencies—namely, to measure a schedule's quality by the rate at which it renders tasks eligible for execution. IC-Scheduling theory has thus far striven for schedules that maximize the number of eligible tasks at every step of a computation—a goal that is inconsistent with the structure of many computations. AREA-maximization weakens this "at every step" goal to an "on average" goal which is *always achievable*.

Progress thus far. The AREA-maximization metric, enjoys several significant properties. (a) Every computation-dag $\mathcal{G}$ admits an AREA-maximizing schedule. (b) If a computation-dag $\mathcal{G}$ admits an IC-optimal schedule then every AREA-maximizing schedule is also IC-optimal. Thus, we never lose scheduling quality by focusing on maximizing AREA, rather than on achieving IC-optimality. (c) AREA-maximization has close relations with other graph-theoretic notions which both help to analyze its computational complexity and lead to an efficient algorithm for *monotonic tree-dags,* a computationally important class of dags, which often do not admit any IC-optimal schedule. Moreover, the AREA-maximization metric preserves several worthwhile properties of IC-Scheduling, including two which help one find optimal schedules: (1) One loses no quality by requiring a schedule to execute all of a computation-dag $\mathcal{G}$'s sinks only after executing all of its nonsinks; (2) one can easily "read off" an AREA-maximizing schedule for a computation-dag $\mathcal{G}$ from an AREA-maximizing schedule for the dual of $\mathcal{G}$ (obtained by reversing all of $\mathcal{G}$'s arcs). Finally, via some novel technical tools, one can efficiently craft AREA-maximizing schedules for large classes of dags that are built from bipartite cliques and cycles, two classes of dags that are the most intransigent for IC-Scheduling.

Projected work. Further theoretical work on the AREA-maximization metric will focus on pinning down the complexity of the scheduling problem, and expanding the class of dag families that we can schedule efficiently … perhaps using the new concept of $\overset{\text{A}}{\triangleright}$-priority. Also planned is an experimental study of the new metric, along the lines of [4, 9, 11].

Acknowledgments. The research of A. Rosenberg was supported in part by NSF Grants CCF-0342417 and CNS-0842578.

References

[1] D. Adolphson and T.C. Hu (1973): Optimal linear ordering. *SIAM J. Appl. Math. 25*, 403–423.

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

[3] G. Cordasco, G. Malewicz, A.L. Rosenberg (2007): Advances in IC-scheduling theory: scheduling expansive and reductive dags and scheduling dags via duality. *IEEE Trans. Parallel and Distributed Systems 18*, 1607–1617.

[4] G. Cordasco, G. Malewicz, A.L. Rosenberg (2007): Applying IC-scheduling theory to some familiar computations. *Wkshp. on Large-Scale, Volatile Desktop Grids.*

[5] G. Cordasco, G. Malewicz, A.L. Rosenberg (2008): Extending IC-scheduling theory via the Sweep Algorithm. *16th Euromicro Intl. Conf. on Parallel, Distributed, and Network-Based Processing (PDP'08)*, 366–373.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein (1999): *Introduction to Algorithms* (2nd Edition). MIT Press, Cambridge, Mass.

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

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

[9] R. Hall, A.L. Rosenberg, A. Venkataramani (2007): A comparison of dag-scheduling strategies for Internet-based computing. *21st IEEE Intl. Parallel and Distr. Processing Symp.*

[10] 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.

[11] G. Malewicz, I. Foster, A.L. Rosenberg, M. Wilde (2007): A tool for prioritizing DAGMan jobs and its evaluation." *J. Grid Computing 5*, 197–212.

[12] G. Malewicz, A.L. Rosenberg, M. Yurkewych (2006): Toward a theory for scheduling dags in Internet-based computing. *IEEE Trans. Comput. 55*, 757–768.

[13] C.H. Papadimitriou and M. Yannakakis (1991): Optimization, approximation, and complexity classes. *J. Computer and System Scis. 43*, 425–440.

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

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

[16] M. Sims, G. Cordasco, A.L. Rosenberg (2008): On clustering tasks in IC-optimal dags. *37th Intl. Conf. on Parallel Processing (ICPP'08)*.

[17] S.W. White and D.C. Torney (1993): Use of a workstation cluster for the physical mapping of chromosomes. *SIAM NEWS*, March, 1993, 14–17.