On Clustering Tasks in IC-Optimal Dags

Mark Sims
Univ. Massachusetts
Amherst, MA 01003 USA
msims@cs.umass.edu

Gennaro Cordasco
Universita di Salerno
Fisciano 84084, ITALY
cordasco@dia.unisa.it

Arnold L. Rosenberg
Colorado State University
Fort Collins, CO 80523, USA
rsnbrg@engr.colostate.edu

Abstract

Strategies are developed for "fattening" the tasks of computation-dags so as to accommodate the heterogeneity of remote clients in Internet-based computing (IC). Earlier work has developed the underpinnings of IC-Scheduling theory*, an algorithmic framework for scheduling computations having intertask dependencies for IC. The theory's schedules strive to render tasks eligible for execution at the maximum possible rate, so as to: (a) utilize remote clients' computational resources well, by enhancing the likelihood of having work to allocate to an available client; (b) lessen the likelihood of a computation's stalling for lack of tasks that are eligible for allocation. The current study begins to enhance IC-Scheduling theory so that it can accommodate the varying computational resources of remote clients. The techniques developed here render a dag multi-granular by clustering its tasks. Several clustering strategies are developed: one works for any dag but produces only a limited variety of "fattened" tasks; others exploit the detailed structure of the dag being scheduled but allow a broad range of "fattened" tasks.*

1. Introduction

Ongoing work [15, 16, 14, 5, 7] is developing *IC-Scheduling theory*, an algorithmic framework for scheduling computations having intertask dependencies, for the several modalities of Internet-based computing (*IC*, for short)—including Grid computing ([2, 9, 8]), global computing ([3]), and volunteer computing ([12]). Acknowledging the *temporal unpredictability* of IC—communication is over the Internet, and remote clients are usually not dedicated to the computation being scheduled—the theory strives to craft schedules that maximize the rate at which tasks are rendered eligible for allocation to remote clients, with the aim of: (*a*) enhancing the utilization of remote clients by always having work to allocate to an available client; (*b*) lessening the likelihood of a computation's stalling pending computation of already-allocated tasks. Schedules that are *IC optimal*—i.e., optimal according to the metrics of IC-Scheduling theory—are displayed in [6] for a large variety of practical computations and computational paradigms; two simulation studies—[13], which focuses on four real scientific computations, and [10], which derives IC-optimal schedules for hundreds of artificially generated computations—suggest that schedules produced under IC-Scheduling theory often have substantial computational benefits over schedules produced by a variety of common heuristics (including the FIFO strategy used by Condor [4]).

IC-Scheduling theory has evolved from the case studies of [15, 16] to the initial algorithmic theory of [14] to the significant algorithmic extensions in [5, 7]; the theory can now schedule IC-optimally a large repertoire of familiar "real" computations [6]. There remain, though, practically significant topics that the theory has yet to address: one such is the topic of this paper. In all modalities of IC, remote clients may differ drastically in computing power/resources. In volunteer computing, e.g., some clients may be running x86-based pc's while others are running multi-core, multi-GHz ones. These disparities can be even greater in Grid computing, since the hardware allocated to the IC computation may be whatever is not needed for more critical local work. Yet, thus far, IC-Scheduling envisions an idealized scenario in which the IC server allocates one task at a time to a remote client as it becomes available. Following systems-oriented studies such as [11], we have begun to study how to make a computation's tasks *multi-granular,* in order to accommodate the computational heterogeneity of clients in IC. We still specify a computation using fine-grain tasks, but we now *cluster* tasks as necessary to coarsen them. In this paper, we obviate the *ad hoc* clustering strategies for specific compu-

0190-3918/08 $25.00 © 2008 IEEE
DOI 10.1109/ICPP.2008.14

tations of [6], by developing *systematic* task-clustering strategies that apply to broad families of computations that admit IC-optimal schedules. We strive for techniques that allow us to choose the size of a client's assigned task-cluster—i.e., the coarseness of the client's assigned task—*dynamically,* acknowledging our *a priori* ignorance of the resources a client will have at each arrival. Our focus on computations that admit IC-optimal schedules limits significantly the class of computations that we can deal with: many computations do not admit any IC-optimal schedule [14]. However, the computations that we can deal with include a broad range of (especially) scientific computations encountered in real IC, including, e.g., the ones from [6].

Main results. We develop a variety of task-clustering strategies, noting that some may be preferred over others in certain computing environments (e.g., some may produce clusters that demand less inter-client communication). All of our strategies ensure that the residual dag after every clustering admits an IC-optimal schedule—so that we retain the algorithmic benefits of the theory, even as we accommodate the heterogeneity of clients. Our first strategy works for any dag that admits an IC-optimal schedule, but it produces only a limited variety of clustered tasks; our other strategies exploit the detailed structure of the dag being scheduled (as exposed by the scheduling algorithms of [14]), but they allow a broad range of task clusters.

2. Background and Motivation

2.1. A Scheduling Model for IC.

Basic concepts. A *(computation-)dag* $\mathcal{G}$ has a set $\mathcal{N}_{\mathcal{G}}$ of *nodes*, each representing a task in a computation, and a set $\mathcal{A}_{\mathcal{G}}$ of *arcs*, each representing an intertask dependency. For $(u \to v) \in \mathcal{A}_{\mathcal{G}}$: • 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 a node 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, where 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 $\mathcal{G}_1$ and $\mathcal{G}_2$ is the dag with node-set $\mathcal{N}_{\mathcal{G}_1} \cup \mathcal{N}_{\mathcal{G}_2}$ and arc-set $\mathcal{A}_{\mathcal{G}_1} \cup \mathcal{A}_{\mathcal{G}_2}$.

When one executes a computation-dag[1] $\mathcal{G}$, a node v becomes *eligible* (for execution) only after all of its parents have been executed. (Hence, sources are always eligible.) We do not allow recomputation, so a node loses eligibility once executed. In compensation, after a node v has been executed, new nodes may be rendered eligible; this occurs when v is their last parent to be executed.

[1] For brevity, we henceforth omit the qualifier "computation."

A *schedule* for a dag $\mathcal{G}$ is a rule for selecting which eligible node to execute at each step when executing $\mathcal{G}$. We measure the quality of an execution of $\mathcal{G}$ by the number of eligible nodes after each node-execution—the more, the better. (Note that *we measure time in an event-driven manner*, as the number of nodes that have been executed to that point; each node execution is a *step*.) Our goal is to execute $\mathcal{G}$'s nodes in an order that maximizes the production rate of eligible nodes *at every step of the computation*. A schedule for $\mathcal{G}$ that achieves this demanding goal is said to be *IC-optimal*.

The significance of IC optimality stems from the following scenarios. Schedules that produce eligible nodes more quickly may: (1) reduce the chance of a computation's stalling when remote clients are slow—so that new tasks cannot be allocated pending the return of already allocated ones; (2) satisfy more requests for tasks that arrive at (roughly) the same time, thereby increasing "parallelism." The simulation studies in [13, 10] both bolster our hope that the preceding intuition does indeed enhance the computational speed achievable under IC.

IC-optimal schedules via dag structure. We review the major ideas in the algorithmic scheduling framework of [14], noting first that *the framework mandates executing all of a dag's nonsinks before any of its sinks*.

The priority relation $\triangleright$. For $i = 1, 2$, let the dag $\mathcal{G}_i$ have s_i nonsinks, and let it admit the IC-optimal schedule Σ_i. Say that the following inequalities hold:

$$(\forall x \in [0, s_1])\ (\forall y \in [0, s_2]):\ \ E_{\Sigma_1}(x) + E_{\Sigma_2}(y) \\ \leq E_{\Sigma_1}(\min\{s_1, x+y\}) + E_{\Sigma_2}(\max\{0, x+y-s_1\}).$$

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 1. **(a)** *The relation $\triangleright$ is transitive.* **(b)** *One can decide in time $O(s_1 s_2)$ whether or not $\mathcal{G}_1 \triangleright \mathcal{G}_2$.*

Building complex dags via *composition*.

- Start with a set $\mathcal{B}$ of base dags.
  We focus on *connected bipartite* dags, each termed a *CBBB*, for "Connected Bipartite Building Block."
- One composes $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{B}$—possibly 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, $\mathcal{G}_1 + \mathcal{G}_2$, of $\mathcal{G}_1, \mathcal{G}_2$. Rename nodes to ensure that $\mathcal{N}_{\mathcal{G}}$ is disjoint from $\mathcal{N}_{\mathcal{G}_1}$ and $\mathcal{N}_{\mathcal{G}_2}$.

  – 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.

– Pairwise identify the nodes in S_1 and S_2 in some way. The resulting set is $\mathcal{G}$'s node-set; the induced set of arcs is $\mathcal{G}$'s arc-set.[2]

- 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* $[\mathcal{G}_1 \Uparrow \mathcal{G}_2]$. Moreover, we say that $\mathcal{G}_i$ *composes directly with* $\mathcal{G}_j$ when forming $\mathcal{G}$ just when some sink of $\mathcal{G}_j$ is merged with a source of $\mathcal{G}_i$. Importantly, composition is associative.

A. Scheduling using composition and $\triangleright$-priority. $\mathcal{G}$ is a $\triangleright$*-linear composition* of CBBBs $\mathcal{G}_1, \ldots, \mathcal{G}_n$ 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 $\triangleright$-priority orders $\mathcal{G}$'s CBBBs consistently with the partial order imposed by topological dependencies.

Theorem 1 ([14]). *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 following schedule Σ for $\mathcal{G}$ 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.*

A suite of algorithms in [14] determine whether or not a dag $\mathcal{G}$ can be decomposed into a set of CBBBs $\mathcal{G}_i$ that satisfy Theorem 1.

If $\mathcal{G}$ is composite of type $\mathcal{G}_1 \Uparrow \cdots \Uparrow \mathcal{G}_n$, then its *super-dag* $\mathcal{G}'$ has node-set $\{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$; its arcs indicate the compositions that created $\mathcal{G}$: if $\mathcal{G}$ was formed by merging sinks of $\mathcal{G}_i$ with sources of $\mathcal{G}_j$, then $(\mathcal{G}_i \to \mathcal{G}_j) \in \mathcal{A}_{\mathcal{G}'}$. Some samples will hopefully hone the reader's intuition. Focus first on two genres of mesh-dags: *out-meshes,* whose arcs point away from the "origin" (the unique source) and *in-meshes,* whose arcs point toward the "origin" (the unique sink). Fig. 1 shows that, in both cases, the relevant super-dag is a "path-dag." Consider next the *FFT* dag, which encapsulates

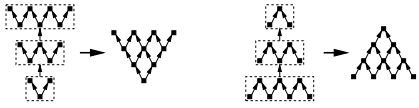

Figure 1. Out- and in-meshes as compositions.

the dependencies in the Fast-Fourier Transform (FFT) algorithm. Fig. 2 shows that this super-dag is a smaller version of the FFT dag.

Some useful CBBBs and their inter-priorities. In order to instantiate our results with specific computations,

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

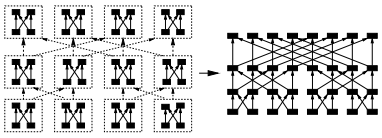

Figure 2. FFT dag: a composition of "butterflies."

we need a repertoire of useful CBBBs. We choose one that is seen in [14, 6] to produce a wide variety of computations, ranging from the FFT and Discrete-Laplace Transform to matrix-multiplication-like computations, and computational paradigms ranging from wavefront-like computations to computations employing the divide-and-conquer and parallel-prefix operators.

The following descriptions proceed left to right along Fig. 3; the drawings distinguish "left" from "right."

W-dags. For $d > 1$, the $(1, d)$*-W-dag* $\mathcal{W}_{1,d}$ has one source and d sinks, with an arc from the source to each sink. For $a, b > 0$, the $(a + b, d)$-W-dag $\mathcal{W}_{a+b,d}$ is obtained from the (a, d)-W-dag $\mathcal{W}_{a,d}$ and the (b, d)-W-dag $\mathcal{W}_{b,d}$ by identifying (or, merging) the rightmost sink of the former with the leftmost sink of the latter.

M-dags. For $s > 0, d > 1$, the (s, d)*-M-dag* $\mathcal{M}_{s,d}$ is obtained from $\mathcal{W}_{s,d}$ by *duality:* reversing all arcs.

N-dags. For $s > 0$, the s*-N-dag* $\mathcal{N}_s$ is obtained from $\mathcal{W}_{s-1,2}$ by adding a new source on the right whose sole arc goes to the rightmost sink. $\mathcal{N}_s$'s leftmost source—its *anchor*—has a child that has no other parents.

(Bipartite) Cycle-dags. For $s > 1$, the s*-(Bipartite) Cycle-dag* $\mathcal{C}_s$ is obtained from $\mathcal{N}_s$ by adding a new arc from the rightmost source to the leftmost sink—so that each source v has arcs to sinks v and $v + 1 \bmod s$.

Figs. 1, 2 suggest that many "real" computations are compositions of the preceding CBBBs. E.g.: the k-sink out-mesh grows from its "origin," via $\mathcal{W}_{1,2}$, then $\mathcal{W}_{2,2}$, $\ldots$, then $\mathcal{W}_{k,2}$; dually, the k-source in-mesh grows from its sources M-dags (Fig. 1); the FFT dag is built from copies of $\mathcal{C}_2$ (Fig. 2); the expansion (resp., reduction) portion of an expansion-reduction-dag is built from instances of $\mathcal{W}_{1,2}$ (resp., $\mathcal{M}_{1,2}$).

We exploit $\triangleright$-priorities among our CBBBs.

Theorem 2 ([14]). *Necessary and sufficient conditions for $\triangleright$-priority among our CBBBs:*

1. *For all $s, d > 0$, $\mathcal{W}_{s,d} \triangleright \mathcal{G}$ for:*
   - *$\mathcal{W}_{s',d'}$ when $[d' < d]$ or $[[d' = d]$ and $[s' \geq s]]$;*
   - *all M-dags, N-dags, and Cycle-dags.*
2. *For all $s > 0$, $\mathcal{N}_s \triangleright \mathcal{G}$ for:*
   - *every $\mathcal{N}_{s'}$, for all s'; all M-dags.*

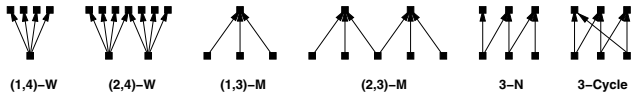

Figure 3. *A useful repertoire of bipartite building-block dags (CBBBs).*

3. *For all $s > 0$, $\mathcal{C}_s \rhd \mathcal{G}$ for:*
   - *$\mathcal{C}_s$; all M-dags.*
4. *For all $s, d > 0$, $\mathcal{M}_{s,d} \rhd \mathcal{M}_{s',d'}$ when $[d' > d]$ or $[[d' = d]$ and $[s' \leq s]]$.*

B. Duality-based scheduling tools. The *dual,* $\widetilde{\mathcal{G}}$, of dag $\mathcal{G}$ is obtained by reversing all of $\mathcal{G}$'s arcs. One can infer both IC-optimal schedules and $\rhd$-priorities for a dag $\mathcal{G}$ from corresponding entities for $\widetilde{\mathcal{G}}$

Let $\mathcal{G}$ have n nonsinks, $U = \{u_1, \ldots, u_n\}$, and N nonsources, $V = \{v_1, \ldots, v_N\}$. Let schedule Σ execute U's nodes in the order $u_{k_1}, \ldots, u_{k_n}$ (followed by all of $\mathcal{G}$'s sinks). Each execution of a nonsink, say u_{k_j}, renders eligible a (possibly empty) "packet" of nonsources, $P_j = \{v_{j,1}, \ldots, v_{j,i_j}\}$. Thus, Σ renders $\mathcal{G}$'s nonsources eligible in a sequence of "packets:"

$$P_1 = \{v_{1,1}, \ldots, v_{1,i_1}\}, \ \ldots, \ P_n = \{v_{n,1}, \ldots, v_{n,i_n}\}.$$

A schedule $\widetilde{\Sigma}$ for $\widetilde{\mathcal{G}}$ is *dual to* Σ if it executes $\widetilde{\mathcal{G}}$'s nonsinks—i.e., V's nodes—in an order of the form[3]

$$[[v_{n,1}, \ldots, v_{n,i_n}]], \ \ldots, \ [[v_{1,1}, \ldots, v_{1,i_1}]],$$

after which, $\widetilde{\Sigma}$ executes $\widetilde{\mathcal{G}}$'s sinks. ($\widetilde{\mathcal{G}}$ generally admits many schedules that are dual to Σ.)

Theorem 3 ([5])**.** *Let $\mathcal{G}$ admit the IC-optimal schedule Σ. Any schedule for $\widetilde{\mathcal{G}}$ that is dual to Σ is IC optimal.*

Our clusterings that exploit detailed structure in a dag $\mathcal{G}$ require that $\mathcal{G}$ be a $\rhd$-linear composition of CBBBs and that it be presented in a way that exposes the structure that Theorem 1 uses to craft an IC-optimal schedule. Such a presentation *honors* the following:

$$\begin{array}{ll} \mathcal{G} \text{ is composite of type} & \mathcal{G}_1 \Uparrow \cdots \Uparrow \mathcal{G}_n \\ \text{where} & \mathcal{G}_1 \rhd \cdots \rhd \mathcal{G}_n \\ \text{and each } \mathcal{G}_i \text{ admits an IC-optimal schedule } \Sigma_i. & \end{array} \quad (1)$$

2.2. A Formal Approach to Clustering Tasks.

We acknowledge the *ad hoc* clustering techniques in [6] that inspired the development here.

Focus henceforth on a dag $\mathcal{G}$ that is accompanied by an IC-optimal schedule Σ. We want to cluster $\mathcal{G}$'s tasks into coarser tasks whose granularities accommodate the varying computational resources of remote clients. This goal is inspired by systems-oriented sources such as [11] that strive to accomplish the same goal in *ad hoc* ways (but, of course, for dags of arbitrary structures). Specifically, when we (playing the IC server) are approached by a remote client C requesting a task, we wish to allocate to C a *fat-task* of $\mathcal{G}$, i.e., a set $F \subseteq \mathcal{N}_{\mathcal{G}}$, whose size (relative to other allocated [fat-]tasks) is appropriate for C's computational power. Having done this, we are left with the *residual dag* $\mathcal{G}^{(F)}$, i.e., the induced subdag of $\mathcal{G}$ on the node-set $\mathcal{N}_{\mathcal{G}} \setminus F$.

A fat-task F should be self-contained: each task in F should be an eligible node or a non-eligible node together with all of its ancestors, back to a set of eligible nodes; i.e., the client that receives F should execute it with no further communication. See Fig. 4.

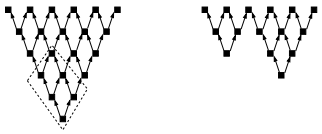

Figure 4. Out-mesh $\mathcal{G}$ and outlined fat-task F; $\mathcal{G}^{(F)}$.

In detail, we strive for fat-tasks F that have:

1. characteristics that we always require—which all of our strategies achieve:
   (a) F should (almost) match the current client's resources; i.e., we should be able to find fat-tasks of (almost) every size.
   (b) $\mathcal{G}^{(F)}$ should admit an IC-optimal schedule.
2. characteristics that are desirable, but often not achievable simultaneously:
   (a) F should render eligible no fewer nodes of $\mathcal{G}$ than would any other $|F|$-node fat-task.
   (b) The communication required to supply inputs for F and receive its results should be small. (Few arcs should connect F with $\mathcal{G}^{(F)}$.)

3. Two General Structure-Based Strategies

Our first strategy for fattening tasks applies to arbitrary dags that admit IC-optimal schedules. The strate-

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

gies in subsequent subsections demand that the IC-optimal schedules arise via Theorem 1.

3.1. The *Direct* Strategy. **Direct clustering: general**. A schedule Σ for a dag $\mathcal{G}$ is a one-to-one map of $\mathcal{N}_{\mathcal{G}}$ onto the set[4] $[1, |\mathcal{N}_{\mathcal{G}}|]$. ($\Sigma$ thereby associates each node of $\mathcal{G}$ with its execution step.) Our first strategy for building fat-tasks—*the direct task-clustering strategy*—assigns tasks to a fat-task in order of their execution by Σ. Thus, the k-node fat-task F_k is the set

$$F_k \;=\; \{\Sigma^{-1}(1), \ldots, \Sigma^{-1}(k)\}. \tag{2}$$

Theorem 4. *Let $\mathcal{G}$ admit the IC-optimal schedule Σ, and let F_k be any fat-task created for $\mathcal{G}$ via direct clustering.*[5] *F_k is* **(a)** self-contained *and* **(b)** productive. **(c)** *$\mathcal{G}^{(F_k)}$ admits an IC-optimal schedule.*

Sketch. **(a)** Since each $v \in F_k$ is eligible when it is executed, all of v's ancestors must have been executed before v, hence must belong to F_k.
(b) This follows because Σ is an *IC-optimal* schedule.
(c) If $\mathcal{G}^{(F_k)}$ did not admit an IC-optimal schedule, then, in particular, the schedule Σ' for $\mathcal{G}^{(F_k)}$ that is the length-$(|\mathcal{N}_{\mathcal{G}}| - k)$ "tail" of Σ would not be IC optimal. (Σ' mimics Σ after it executes the nodes in F_k.) There would then be a schedule Σ'' for $\mathcal{G}^{(F_k)}$ and a step t'' such that[6] $\widehat{E}_{\Sigma''}(t'') > \widehat{E}_{\Sigma'}(t'')$. However, consider the schedule $\Sigma^{\#}$ for $\mathcal{G}$ that mimics Σ for k steps on $\mathcal{G}$, then mimics Σ'' for t'' steps on $\mathcal{G}^{(F_k)}$. By assumption, we would have $\widehat{E}_{\Sigma^{\#}}(k + t'') > \widehat{E}_{\Sigma}(k + t'')$, which would contradict Σ's alleged IC optimality! □

Direct clustering: $\triangleright$-linear composite dags. *Focus henceforth on dags $\mathcal{G}$ that honor (1), and on the IC-optimal schedule Σ for each such $\mathcal{G}$ that Theorem 1 yields.* For these dags, the direct clustering strategy is discernible in $\mathcal{G}$'s structure.

Corollary 1. *The following procedure builds a productive fat-task F and yields a residual dag $\mathcal{G}^{(F)}$ that admits an IC-optimal schedule.*

Add all sources of $\mathcal{G}_1$ in the order of Σ_1, then all sources of $\mathcal{G}_2$ in the order of Σ_2, and so on, until the desired fat-task size is reached or until all sources have been added. In the former case, we are done; in the latter, add sinks of $\mathcal{G}$ in any order until either none remain or the desired fat-task size has been reached.

Note that $\mathcal{G}^{(F)}$ is generally not a $\triangleright$-linear composition of CBBBs, because removing F can destroy the required relationship between topological order and $\triangleright$-priority order. Our challenge is to develop techniques which ensure that $\mathcal{G}^{(F)}$ *is* a $\triangleright$-linear composition.

3.2. A Staggered Clustering Strategy. We now begin our search for task-clustering strategies that may be useful only in various contexts, because they often sacrifice productivity for other desiderata.

A word about productivity will motivate the upcoming development. While direct clustering satisfies all of our goals, including productivity, other quality criteria warrant exploring other avenues for building fat-tasks. One prime such criterion, for instance, is to control the amount of communication needed to supply a client with a fat-task and to receive the results of executing that task. As a concrete example, consider the dag $\mathcal{G}$ in Fig. 5(a). Easily, $\mathcal{G}$ is composite of type

$$\begin{aligned}\mathcal{W}_{1,2} \Uparrow \mathcal{W}_{1,2} \Uparrow \mathcal{W}_{2,2} \Uparrow \mathcal{W}_{2,2} \Uparrow \mathcal{W}_{3,2} \Uparrow \mathcal{W}_{3,2} \\ \Uparrow \mathcal{W}_{4,2} \Uparrow \mathcal{W}_{4,2} \Uparrow \mathcal{W}_{5,2} \Uparrow \mathcal{W}_{6,2} \Uparrow \mathcal{W}_{10,2},\end{aligned}$$

hence, by Theorem 2, is a $\triangleright$-linear composition:

$$\begin{aligned}\mathcal{W}_{1,2} \triangleright \mathcal{W}_{1,2} \triangleright \mathcal{W}_{2,2} \triangleright \mathcal{W}_{2,2} \triangleright \mathcal{W}_{3,2} \triangleright \mathcal{W}_{3,2} \\ \triangleright \mathcal{W}_{4,2} \triangleright \mathcal{W}_{4,2} \triangleright \mathcal{W}_{5,2} \triangleright \mathcal{W}_{6,2} \triangleright \mathcal{W}_{10,2}\end{aligned} \tag{3}$$

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

- Execute the two copies of $\mathcal{W}_{1,2}$,
- then the two copies of $\mathcal{W}_{2,2}$,
- then the two copies of $\mathcal{W}_{4,2}$,
- then the copy of $\mathcal{W}_{5,2}$, the copy of $\mathcal{W}_{6,2}$, and the copy of $\mathcal{W}_{10,2}$, in that order.

Say that one seeks a 6-node fat-task F from $\mathcal{G}$. Direct clustering produces a task $F^{(\mathrm{dir})}$ comprising the two copies of $\mathcal{W}_{1,2}$ and of $\mathcal{W}_{2,2}$ from the head of (3). $\mathcal{G}^{(F^{(\mathrm{dir})})}$ (see Fig. 5(b)) is a $\triangleright$-linear composition:

$$\mathcal{W}_{3,2} \triangleright \mathcal{W}_{3,2} \triangleright \mathcal{W}_{4,2} \triangleright \mathcal{W}_{4,2} \triangleright \mathcal{W}_{5,2} \triangleright \mathcal{W}_{6,2} \triangleright \mathcal{W}_{10,2};$$

hence, admits an IC-optimal schedule. However, one must "cut" eight arcs in order to excise $F^{(\mathrm{dir})}$ from $\mathcal{G}$, representing 8 data that the client receiving $F^{(\mathrm{dir})}$ must communicate to the server after completing its computation. As an alternative strategy, let F consist of the 6-node prefix from the lefthand out-mesh of $\mathcal{G}$—which comprises one copy each of $\mathcal{W}_{1,2}$, $\mathcal{W}_{2,2}$, and $\mathcal{W}_{3,2}$. $\mathcal{G}^{(F)}$ (see Fig. 5(c)) also is a $\triangleright$-linear composition:

$$\mathcal{W}_{1,2} \triangleright \mathcal{W}_{2,2} \triangleright \mathcal{W}_{3,2} \triangleright \mathcal{W}_{4,2} \triangleright \mathcal{W}_{4,2} \triangleright \mathcal{W}_{5,2} \triangleright \mathcal{W}_{6,2} \triangleright \mathcal{W}_{10,2};$$

hence, also admits IC-optimnal schedule. Moreover, only *six* arcs need be "cut" to excise F from $\mathcal{G}$. Thus, this strategy produces a fat-task that requires less communication than the direct strategy!

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

[5] $\mathcal{G}$ may admit many IC-optimal schedules, each leading to a different "directly selected" F_k.

[6] For dag $\mathcal{H}$, schedule Σ for $\mathcal{H}$, and integer t, $E_{\Sigma}(t)$ (resp., $\widehat{E}_{\Sigma}(t)$) is the number of eligible nonsources (resp., nodes) on $\mathcal{H}$ at step t.

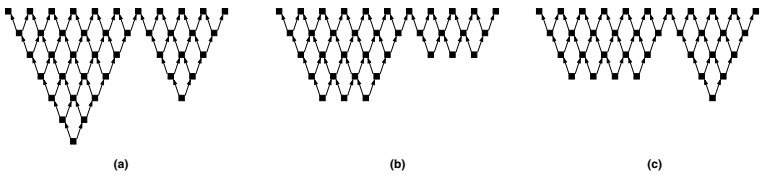

Figure 5. A situation that motivates nonproductive clusterings.

An aside. Viewed as dags, both $F^{(\mathrm{dir})}$ and F are $\triangleright$-linear compositions, hence admit IC-optimnal schedules. Of course, this fact is significant only if the remote client parcels the fat-task out to other clients.

The preceding alternative strategy is not an *ad hoc* one designed uniquely for $\mathcal{G}$: it is a *staggered* extension of Corollary 1. The *Staggered* clustering strategy builds a fat-task F by always "donating" *entire* CBBBs from $\mathcal{G}$ to F. This is similar to the Direct Strategy, except that it can "skip over" a constituent CBBB at will.

Lemma 2. *The following procedure creates a fat-task F such that both $\mathcal{G}^{(F)}$ and the induced subdag of $\mathcal{G}$ on node-set F admit IC-optimal schedules.*

Add all sources of some $\mathcal{G}_{i_1}$ in the order of Σ_{i_1}, then all sources of some $\mathcal{G}_{i_2}$, where $i_2 > i_1$, in the order of Σ_{i_2}, and so on.

Sketch. The induced dag on F is composite of type $\mathcal{G}_{i_1} \Uparrow \cdots \Uparrow \mathcal{G}_{i_\ell}$ for some $\ell \geq 1$; $\mathcal{G}^{(F)}$ is composite of type $\mathcal{G}_{j_1} \Uparrow \cdots \Uparrow \mathcal{G}_{j_m}$, where $j_1 < \cdots < j_m$ and $\{j_1, \ldots, j_m\} = [1, n] \setminus \{i_1, \ldots, i_\ell\}$. From (1) and the transitivity of $\triangleright$-priority:

$$[\mathcal{G}_{j_1} \triangleright \cdots \triangleright \mathcal{G}_{j_m}] \text{ and } [\mathcal{G}_{i_1} \triangleright \cdots \triangleright \mathcal{G}_{i_\ell}].$$

Two invocations of Theorem 1 complete the proof. □

4. Clusterings for Specific Dag-Families

We turn finally to strategies that exploit detailed structure of $\mathcal{G}$, including its specific constituent CBBBs. While restricting the class of applicable dags, these results identify dags that accommodate *all* fat-tasks—i.e., are "universal" donors. *With such dags, one can choose clusterings that optimize whatever criteria one wishes.*

4.1. The enabling framework.

We seek to remove a fat-task F from $\mathcal{G}$ by removing a fat-task $F_i \subseteq F$ from each $\mathcal{G}_i$ in $\mathcal{G}$'s $\triangleright$-chain of CBBBs (1), in such a way that $\mathcal{G}^{(F)}$ admits an IC-optimal schedule. We find the following sufficient condition for achieving this goal.

Theorem 5. *Let $F = F_1 \cup \cdots \cup F_k$ be a fat-task for $\mathcal{G}$, where, for each $i \in [1, k]$, $F_i \subseteq \mathcal{N}_{\mathcal{G}_i}$. $\mathcal{G}^{(F)}$ admits an IC-optimal schedule whenever the following conditions hold, for some $k \leq n$.*

Each $\mathcal{G}_i^{(F_i)}$ is a sum

$$\mathcal{G}_i^{(F_i)} \;=\; (\mathcal{G}_{i,1} + \cdots + \mathcal{G}_{i,n_i}) + \{v_1, \ldots, v_{m_i}\},$$

of CBBBs (the $\mathcal{G}_{i,j}$) and isolated nodes (the v_ℓ), where, for all $i \in [1, k]$ and $a \in [1, n_i]$:

1. *$\mathcal{G}_{i,a} \triangleright \mathcal{G}_i$;*
2. *$\mathcal{G}_{q,r} \triangleright \mathcal{G}_{i,a}$ for each parent $\mathcal{G}_{q,r}$ of $\mathcal{G}_{i,a}$ in the super-dag of $\mathcal{G}_i^{(F_i)}$.*[7]
3. *for all $b \in [1, n_i]$: every pair of CBBBs, $\mathcal{G}_{i,a}$ and $\mathcal{G}_{i,b}$, are* $\triangleright$-comparable[8]

Sketch. Because F contains nodes only from $\mathcal{G}_1, \ldots, \mathcal{G}_k$, $\mathcal{G}^{(F)}$ contains a subdag that is composite of type $\mathcal{G}_{k+1} \Uparrow \cdots \Uparrow \mathcal{G}_n$. (Recall: $\mathcal{G}_{k+1} \triangleright \cdots \triangleright \mathcal{G}_n$.)

1. By the transitivity of $\triangleright$, for all $j \in [1, k]$ and every summand $\mathcal{G}_{j,h}$ of $\mathcal{G}_j^{(F_j)}$, we have $\mathcal{G}_{j,h} \triangleright \mathcal{G}_{j+1}^{(F_{j+1})}$.
2. By hypothesis, $[\mathcal{G}_{q,r} \triangleright \mathcal{G}_{i,j}]$ for any $\mathcal{G}_{q,r}$ with $q \leq k$ and $r \leq n_q$ that $\mathcal{G}_{i,j}$ composes directly with.
3. By hypothesis, all sibling summands in each sum $(\mathcal{G}_{i,1} + \cdots + \mathcal{G}_{i,n_i})$ are $\triangleright$-comparable.

Putting these facts together, we have: • Both $\mathcal{G}'$, the prefix of $\mathcal{G}^{(F)}$ that resulted from $\mathcal{G}_1, \ldots, \mathcal{G}_k$ and $\mathcal{G}''$, the suffix of $\mathcal{G}^{(F)}$ that involves only $\mathcal{G}_{k+1}, \ldots, \mathcal{G}_n$ are $\triangleright$-linear compositions; • $\mathcal{G}' \triangleright \mathcal{G}''$. Thus, $\mathcal{G}^{(F)}$ is a $\triangleright$-linear composition, hence admits an IC-optimal schedule. □

4.2. Composing *Reticulated* W-, Cycle-, and N-dags.

Certain large classes $\mathbf{C}$ of dags are *universal donors* of fat-tasks. The analyses that establish universality show that, when one removes *any* fat-task F from any $\mathcal{G} \in \mathbf{C}$, the resulting $\mathcal{G}^{(F)}$ satisfies Theorem 5. Hence, these analyses can be adapted to many classes of dags other than those we treat here.

A dag $\mathcal{G}$ that honors (1) is *reticulated* if the following holds. If CBBB $\mathcal{G}_i$ composes directly with CBBB $\mathcal{G}_j$—so that $\mathcal{G}_i$ is a child of $\mathcal{G}_j$ in $\mathcal{G}$'s super-dag—then there is a one-to-one mapping of $\mathcal{G}_j$'s sources into $\mathcal{G}_i$'s sources. (Several sources of $\mathcal{G}_j$ may share a child among $\mathcal{G}_i$'s sources.) When a fat-task "appropriates" a source from a child-CBBB $\mathcal{G}'$ in a reticulated dag, then it also "appropriates" at least one source from one of $\mathcal{G}'$'s parent-CBBBs; cf. Fig. 6.

[7] I.e., $\mathcal{G}_{i,a}$ composes directly with $\mathcal{G}_{q,r}$ when forming $\mathcal{G}_i^{(F_i)}$.

[8] That is, either $[\mathcal{G}_{(i,a)} \triangleright \mathcal{G}_{(i,b)}]$ or $[\mathcal{G}_{(i,b)} \triangleright \mathcal{G}_{(i,a)}]$.

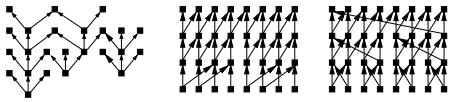

Figure 6. Reticulated compositions: W-, N-, Cycle-dags.

In short, reticulated compositions of W-dags, Cycle-dags, and N-dags are universal donors.

Residual dags for CBBBs. For the three CBBBs we treat now, residual dags are sums of these same three CBBBs, possibly plus isolated nodes that arise when F removes some of $\mathcal{B}$'s sources without their children (which are sinks of $\mathcal{B}$). The isolated nodes will "automatically" get executed correctly so, for brevity, we ignore them, other than to acknowledge their existence.

A fat-task F for a CBBB $\mathcal{B}$ is *compact* if F selects sources that are consecutive in the "standard" drawing of $\mathcal{B}$ in Fig. 3; for Cycle-dags, we interpret "consecutive" cyclically.

Lemma 3. *Removing a compact fat-task F from a CBBB $\mathcal{B}$ produces a residual dag $\mathcal{B}^{(F)}$ that is a sum of isolated nodes plus the following CBBBs.*

$\mathcal{W}$	*outdegree-d W-dag*
$\mathcal{W}^{(F)}$	1 *or* 2 *outdegree-d W-dag(s)*
$\mathcal{C}$	*Cycle-dag $\mathcal{C}$*
$\mathcal{C}^{(F)}$	1 *W-dag*
$\mathcal{N}$	*N-dag $\mathcal{N}$*
$\mathcal{N}^{(F)}$	≥ 0 *outdegree-*2 *W-dags,* 0 *or* 1 *N-dag(s)*

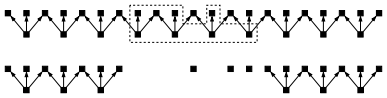

Figure 7. $\mathcal{W}_{10,3}$ and outlined fat-task F; $\mathcal{W}_{10,3}^{(F)}$.

Removing F from $\mathcal{C}$ leaves 1 W-dag; cyclic symmetry makes F's location irrelevant; see Fig. 8.

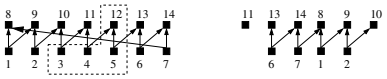

Figure 8. $\mathcal{C}_7$ and outlined fat-task F; $\mathcal{C}_7^{(F)}$.

Removing F from $\mathcal{N}$ leaves 1 outdegree-2 W-dag (resp., 1 N-dag to the left (resp., right) of F; see Fig. 9. □

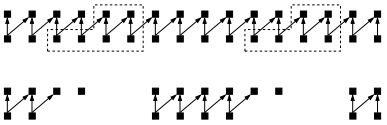

Figure 9. $\mathcal{N}_{16}$ and outlined fat-task F; $\mathcal{N}_{16}^{(F)}$.

Theorem 6. (The universal-donor theorem) *Every reticulated composition of W-dags or of Cycle-dags or of N-dags is a universal donor of fat-tasks.*

Sketch. By [14], every W-dag or Cycle-dag or N-dag admits an IC-optimal schedule.

For W-dags. Specialized to W-dags, (1) becomes:

$$\begin{array}{ll} \mathcal{G} \text{ is composite of type} & \mathcal{W}^{(1)} \Uparrow \cdots \Uparrow \mathcal{W}^{(n)} \\ \text{where} & \mathcal{W}^{(1)} \rhd \cdots \rhd \mathcal{W}^{(n)}. \end{array} \tag{4}$$

Let F be a fat-task for $\mathcal{G}$, that excises nodes from CBBBs $\mathcal{W}^{(1)}, \ldots, \mathcal{W}^{(k)}$. By Lemma 3, $\mathcal{G}^{(F)}$ is composite of type $\widehat{\mathcal{W}}^{(1)} \Uparrow \cdots \Uparrow \widehat{\mathcal{W}}^{(n)}$, where:

$$\widehat{\mathcal{W}}^{(j)} = \begin{cases} \mathcal{W}^{(j)} & \text{if } j > k \\ \mathcal{W}^{(j,1)} + \cdots + \mathcal{W}^{(j,m_j)} & \text{for some } m_j, \\ & \text{if } j \in [1,k]. \end{cases}$$

Several invocations of Theorem 2 now verify the three clauses of Theorem 5. Specifically, for each j:

(1) For all a, $\mathcal{W}^{(j,a)} \rhd \mathcal{W}_j$ because any W-dag has lower $\rhd$-priority than any of its sub-W-dags.

(2) Each $\mathcal{W}^{(j,a)}$ has lower $\rhd$-priority than any of its parents in $\mathcal{G}$'s super-dag. Specifically, if F_j removes sources from $\mathcal{W}^{(j)}$ that are sinks of one of $\mathcal{W}^{(j)}$'s parents, $\mathcal{W}^{(q)}$, then it removes at least that many sources also from $\mathcal{W}^{(q)}$. Hence, if $\mathcal{W}^{(q)} \rhd \mathcal{W}^{(j)}$ *before* removing F, then $\mathcal{W}^{(q,r)} \rhd \mathcal{W}^{(j,a)}$ after the removal.

(3) Every two W-dags are $\rhd$-comparable.

The arguments for Cycle-dags and N-dags have the same general structure. □

4.3. Compositions of M-dags.

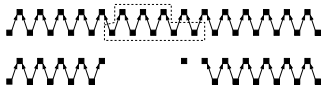

Figure 10. $\mathcal{M}_{15,2}$ and outlined fat-task F; $\mathcal{M}_{15,2}^{(F)}$.

Fig. 10 suggests why M-dags are harder to analyze than our other CBBBs. fat-tasks split reticulated

compositions of other CBBBs split into easily "controlled" sums (Lemma 3), but we have yet to discover an analogue of "reticulated" for compositions of M-dags. Lacking this, we invoke a much different analysis technique here, which depends on Theorem 3. The reader should be able to adapt our result about *indegree*-2 M-dags to M-dags of arbitrary in-degrees.

A composition $\mathcal{G}$ of W-dags that honors (1) is *anti-reticulated* if there is a drawing of $\mathcal{G}$ such that: If CBBB $\mathcal{G}_i$ is a child of $\mathcal{G}_j$ in $\mathcal{G}$'s super-dag—then there is a set of noncrossing independent arcs that maps every source of $\mathcal{G}_i$ to a distinct source of $\mathcal{G}_j$. Next, the *closure* $\widehat{F}$ of a fat-task for $\mathcal{G}$, F, is the *dag* obtained from $\mathcal{G}$ by removing all nodes of $\mathcal{G}^{(F)}$ *except for isolated nodes*. $\mathcal{N}_{\widehat{F}}$ may differ from F if $\widehat{F}$ contains nodes that are isolated in $\mathcal{G}^{(F)}$ but are connected to some node in F. The following lemma yields to arguments similar to earlier ones.

Lemma 4. *Let $\mathcal{G}$ be an anti-reticulated composition of W-dags that honors (1). For every fattened dag F, the dag $\widehat{F}$ admits an IC-optimal schedule.*

Theorem 7. *Let $\mathcal{G}$ be composition of M-dags that honors (1) such that $\mathcal{G}$'s dual dag $\widetilde{\mathcal{G}}$ is anti-reticulated, then $\mathcal{G}$ is a universal donor of fat-tasks.*

Sketch. Let F be a fat-task for $\mathcal{G}$. Let $\widetilde{F}$ be the $(N_{\mathcal{G}} - |\widehat{F}|)$-node fat-task for $\mathcal{G}$'s dual dag $\widetilde{\mathcal{G}}$, that is obtained by removing all the nodes of $\widehat{F}$. Easily, then, $\widetilde{\mathcal{G}}^{(\widehat{F})} = \widehat{\widetilde{F}}$. Lemma 4 says that $\widetilde{\mathcal{G}}^{(\widehat{F})}$ admits an IC-optimal schedule, and Theorem 3 extends this to $\mathcal{G}^{(\widehat{F})}$. The result now follows because $\mathcal{G}^{(\widehat{F})}$ and $\mathcal{G}^{(F)}$ can differ only in the latter dag's having additional isolated nodes. □

5. Conclusions

We have begun to study how to cluster the tasks of computation-dags—to obtain coarse-grain tasks—so as to accommodate the heterogeneity of remote clients in Internet-based computing. Acknowledging the importance of having many clustering strategies, that address the exigencies of the many computational environments that abound throughout the Internet, we present a range of task-clustering strategies, ranging from those that ignore the specific structure of the dag being scheduled, to those that exploit the gross structure of the dag, to those that depend in a very detailed way on that structure.

References

[1] L.I. Bluestein (1970): A linear filtering approach to the computation of the Discrete Fourier Transform. *IEEE Trans. Audio Electroacoustics, AU-18*, 451–455.

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

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

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

[5] 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.

[6] G. Cordasco, G. Malewicz, A.L. Rosenberg (2007): Applying IC-scheduling theory to some familiar computations. *Wkshp. on Large-Scale, Volatile Desktop Grids (PCGrid'07)*.

[7] 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.

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

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

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

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

[12] 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.

[13] 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.

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

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

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