Stochastic scheduling with variable profile and precedence constraints

Zhen Liu, Eric Sanlaville

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.: (1) 39 63 55 11

Rapports de Recherche

N° 1525

Programme 1
Architectures parallèles, Bases de données,
Réseaux et Systèmes distribués

STOCHASTIC SCHEDULING WITH VARIABLE PROFILE AND PRECEDENCE CONSTRAINTS

Zhen LIU
Eric SANLAVILLE

Septembre 1991

RR-1525

Stochastic Scheduling with Variable Profile and Precedence Constraints

Zhen LIU

INRIA
Centre Sophia Antipolis
2004 Route des Lucioles
BP 109, 06561 Valbonne
FRANCE

Eric SANLAVILLE

Laboratoire MASI
Université Pierre et Marie Curie
4, place Jussieu
75252 Paris Cedex 05
FRANCE

August 28, 1991

Abstract

In this paper, we consider the stochastic profile scheduling problem of a partially ordered set of tasks on uniform processors. The set of available processors varies in time. The running times of the tasks are independent random variables with exponential distributions. We obtain simple optimal preemptive policies which stochastically minimize the makespan when the partial order is an interval order, or an in-forest, or an out-forest.

Keywords: Stochastic Scheduling, Profile Scheduling, Makespan, Precedence Constraint, Interval Order, In-Forest, Out-Forest, Uniform Processors, Stochastic Ordering.

Ordonnancement stochastique avec profil variable et contraintes de précédence

Zhen LIU

INRIA
Centre Sophia Antipolis
2004 Route des Lucioles
BP 109, 06561 Valbonne
FRANCE

Eric SANLAVILLE

Laboratoire MASI
Université Pierre et Marie Curie
4, place Jussieu
75252 Paris Cedex 05
FRANCE

26 août 1991

Résumé

Dans ce rapport, nous considérons le problème de l'ordonnancement stochastique d'un ensemble partiellement ordonné de tâches sur des processeurs uniformes. L'ensemble des processeurs disponibles varie au cours du temps. Les durées d'exécution des tâches sont des variables aléatoires indépendantes de même distribution exponentielle. Nous obtenons des politiques préemptives simples qui minimisent stochastiquement la durée totale d'exécution des tâches, quand l'ordre partiel est un ordre d'intervalle, une forescence ou une anti-forescence.

Mots-Clefs: Ordonnancement stochastique, ordonnancement avec profil variable, durée totale d'exécution, ordre d'intervalles, forescence, anti-forescence, processeurs uniformes, ordre stochastique.

1 Introduction

Consider the following scheduling problem. We are given a set of tasks to be run in a system consisting of uniform processors (i.e., processors having different speeds). The executions of these tasks have to satisfy some precedence constraints which are described by a directed acyclic graph, referred to as the task graph. Three types of task graphs will be considered: interval order, in-forest and out-forest. The processing requirements of the tasks are independent random variables with a common exponential distribution. The set of processors available to these tasks varies in time, due to, e.g., failures of the processors or executions of higher-priority tasks. The availability of the processors is referred to as the profile, and it can be arbitrary. The goal is to find optimal preemptive schedules that stochastically minimize the makespan.

When the task graph is an in-forest, and the profile is a constant set of two processors, Chandy and Reynolds [2] proved that the Highest Level first (HL) policy minimizes the expected makespan. Here, the level of a task is simply the distance from it to the root of the tree it appears. Bruno [1] subsequently showed that HL stochastically minimizes the makespan when the system has two identical parallel processors. Pinedo and Weiss [11] extended this last result to the case where tasks at different levels may have different expected task running times. Frostig [7] further generalized the result of Pinedo and Weiss to include increasing likelihood ratio distributions for the task running times. When the number of identical parallel processors in the system is arbitrarily fixed, and the task running times have a common exponential distribution, Papadimitriou and Tsitsiklis [9] proved that HL is asymptotically optimal as the number of tasks tends to infinity.

Coffman and Liu [3] investigated the stochastic scheduling of out-forest on identical parallel processors with constant profile. For the uniform out-forests where all the subtrees are ordered by an embedding relation (see definition in Section 5), they showed that an intuitive priority scheduling policy induced by the embedding relation, referred to as the Largest Tree (LT) policy in this paper, stochastically minimizes the makespan when there are two processors. If, in addition, the out-forests satisfy a uniform root-embedding constraint, then the greedy policy stochastically minimizes the makespan for arbitrary number of processors.

Papadimitriou and Yannakakis [10] studied the deterministic scheduling of interval-ordered tasks. Under the assumptions of unit-execution-time and constant profile, they showed that for any arbitrary number of identical processors, the simple list scheduling induced by the interval order, referred to as the Most Successors (MS) policy in this paper,

minimizes the makespan.

The notion of profile scheduling was first introduced by Ullman [13] and later by Garey et al. [8] in the complexity analysis of deterministic scheduling algorithms. Dolev and Warmuth [4,5,6] carried out various studies on the deterministic nonpreemptive profile scheduling with parallel identical processors. In such a case, the profile is simply the number of available processors at any time. When the tasks have unit-execution-time, Dolev and Warmuth obtained polynomial algorithms for specific profiles (e.g., zigzag profile, bounded profile, etc.) and specific task graphs (e.g., in-forest, out-forest, opposing forest, flat graph, etc.).

In this paper, we investigate the profile scheduling in the stochastic setting with uniform processors. The scheduling is allowed to be preemptive, and the profile is arbitrary and can be unknown a priori. We show that the MS, HL, and LT policies stochastically minimize the makespan of the task graph having an interval order structure, an in-forest structure and a uniform out-forest structure, respectively. Moreover, unless in the degenerated case where there is a single available processor all the time, they are the only optimal policies for the stochastic minimization of the makespan. The results concerning interval-ordered tasks are new, even for constant profile and parallel identical processors. The optimality of the HL policy extends the result of [2] to stochastic minimization of the makespan and to the variable profile. The optimalities of the HL and LT policies extend the results of [1,3] to variable profile and uniform processors.

Our paper is organized as follows. In Section 2, we describe the scheduling problem in detail, and we present some preliminaries. In Sections 3, 4 and 5, we prove that the policies MS, HL and LT stochastically minimize the makespan of the interval-order task graphs, in-forests and uniform out-forests, respectively.

2 Problem Description and Preliminaries

A *task graph* $G = (V, E)$ is a directed acyclic graph, where $V = \{1, 2, \cdots, |V|\}$ is the set of vertices representing the tasks, $E \subset V \times V$ is the set of edges representing the *precedence constraints*: $(i, j) \in E$ if and only if task i must be completed before task j can start. Denote by $p(i)$ and $s(i)$ the sets of immediate *predecessors* and *successors* of $i \in V$, respectively, i.e.,

$$p(i) = \{j : (j, i) \in E\}, \qquad s(i) = \{j : (i, j) \in E\}.$$

Let $S(i)$ be the set of (not necessarily immediate) successors of $i \in V$, i.e.,

$$\forall i: \quad \text{if} \quad s(i) = \emptyset \quad \text{then} \quad S(i) = \emptyset \quad \text{else} \quad S(i) = s(i) \bigcup \left(\bigcup_{j \in s(i)} S(j) \right).$$

Three classes $\mathcal{C}_{i.o}, \mathcal{C}_{i.f}, \mathcal{C}_{o.f}$ of task graphs will be considered in the paper.

interval order $G \in \mathcal{C}_{i.o}$: Each vertex i corresponds to an interval b_i in the real line such that $(i,j) \in E$ if and only if $x \in b_i$ and $y \in b_j$ imply $x < y$.

in-forest $G \in \mathcal{C}_{i.f}$: Each vertex has at most one immediate successor: $|s(i)| \leq 1$, $i \in V$. A vertex $i \in V$ is called a leaf of in-forest G if $p(i) = \emptyset$. A vertex $i \in V$ is called a root of in-forest G if $s(i) = \emptyset$.

out-forest $G \in \mathcal{C}_{o.f}$: Each vertex has at most one immediate predecessor: $|p(i)| \leq 1$, $i \in V$. A vertex $i \in V$ is called a leaf of out-forest G if $s(i) = \emptyset$. A vertex $i \in V$ is called a root of out-forest G if $p(i) = \emptyset$.

Note that these classes of graphs have the following closure property: for all $\mathcal{C} \in \{\mathcal{C}_{i.o}, \mathcal{C}_{i.f}, \mathcal{C}_{o.f}\}$, if $G = (V, E) \in \mathcal{C}$ then $G - \{v\} \in \mathcal{C}$ for all $v \in V$ such that $p(v) = \emptyset$, where $G - \{v\}$ is the graph obtained by deleting vertex v and its adjacent edges. Such a closure property (by deletion) will be used in establishing our results.

The processing requirements of the tasks are identical and independently distributed (i.i.d.) random variables with a common exponential distribution of a constant parameter, say 1. The running time of a task is the processing requirement of the task divided by the speed of the processor on which the task is running.

There are $K \geq 1$ *uniform processors*, indexed by $1, 2, \cdots, K$. The speed of processor k is denoted by α_k. The running times of tasks on processor k, $1 \leq k \leq K$, are thus i.i.d. random variables with exponential distribution of parameter α_k. We assume, by convention, that $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_K$.

The set of processors available to these tasks varies in time, due to, e.g., failures of the processors or executions of higher-priority tasks. The availability of the processors is referred to as the *profile*, and is specified by the sequence $\{a_n, M_n\}_{n=1}^{\infty}$, where the random variables $0 = a_1 < a_2, < \cdots < a_n < \cdots$ are the epochs where the profile is changed, and M_n, $n \geq 1$, is a random set whose elements are the indices of the processors available

during the time interval $[a_n, a_{n+1})$. The profile $\{a_n, M_n\}_{n=1}^{\infty}$ is assumed to be independent of the running times of the tasks. Without loss of generality, we assume that for all $n \geq 1$, $M_n \neq \emptyset$. We will also assume that the profile is not changed infinitely often during any finite time interval: for all $x \in I\!R^+$, there is some finite $n \geq 1$ such that $a_n > x$.

The *scheduling policies* decide when an *enabled task*, i.e., an unassigned unfinished task all of whose predecessors have finished, should be assigned to an available processor. At any time, a task can be assigned to at most one processor, and a processor can execute at most one task. Throughout this paper, we assume that the scheduling policies are preemptive.

We assume that the scheduler has no information on the samples of the (remaining) processing requirements of the tasks. At any time t, $a_n \leq t < a_{n+1}$, the scheduler may not have any information on the truncated sequence $\{a_l, M_l\}_{l=n+1}^{\infty}$. In other words, the scheduler may not know either the future epochs when the profile changes or the sets of available processors in the future. Within such a framework, dynamique preemptive scheduling is necessary.

Let Ψ denote the class of such policies. For any $\pi \in \Psi$, denote by $\pi(G)$ the makespan of the partially ordered set of tasks G, i.e., the maximum of the completion times of the tasks in G.

The goal of the paper is to find policies in Ψ that *stochastically minimize* the makespan of G in the classes $\mathcal{C}_{i.o}, \mathcal{C}_{i.f}$ and some subclasses of $\mathcal{C}_{o.f}$. A policy π_o is said to be optimal within a class $\mathcal{C}$, if for any policy $\pi \in \Psi$, $\pi_o(G) \leq_{st} \pi(G)$ for all $G \in \mathcal{C}$, where the symbol $\leq_{st}$ refers to the standard stochastic inequality. The random variable $X \in I\!R^+$ is *stochastically smaller* than the random variable $Y \in I\!R^+$, denoted by $X \leq_{st} Y$, if and only if $P[X \leq x] \geq P[Y \leq x]$ for all $x \in I\!R^+$.

In order to simplify the proofs of the main results in this paper, we make some reductions on the class of policies Ψ.

Observe first that due to the memoryless property of the exponential distributions, the distribution of the remaining running time of a task running on processor k is still exponential with parameter α_k, provided processor k is available, $1 \leq k \leq K$. If we represent the state of the system by the set of available processors, the remaining task graph and the distributions of the remaining running times of the tasks, then the state does not change between the instants of task completions and of profile modifications. Therefore, we can, without loss of generality, confine ourselves to the class of policies where

preemptions and new task assignments occur only at the instants of task completions and profile modifications. These instants are referred to as the *decision epochs*. Hence, we assume that all the policies in Ψ make their scheduling decisions at these time instants only.

A policy is *idling* if it allows a processor to remain idle when there is an enabled task waiting for assignment. It is easy to see that an optimal policy should never be idling as the distributions of the task running times have infinite support. Therefore, we will also assume that all the policies in Ψ are *nonidling*.

The proofs of our main results will use a coupling argument based on the following well-known result due to Strassen:

Lemma 2.1 *(Strassen) Two random variables $X, Y \in \mathbb{R}$ satisfy $X \leq_{st} Y$ if and only if there exist two random variables $\hat{X}$ and $\hat{Y}$ defined on a common probability space such that $X =_{st} \hat{X}$, $Y =_{st} \hat{Y}$, and $\hat{X} \leq \hat{Y}$ almost surely (a.s.), where $=_{st}$ denotes the equality in distribution.*

3 Stochastic Profile Scheduling of Interval-Order Graph

Let $G = (V, E) \in \mathcal{C}_{i.o.}$ be an interval-order graph. Note that the interval order graphs are connected. An equivalent definition of the interval order (cf. [10]) is that for all $i, j \in V$, either $s(i) \subseteq s(j)$ or $s(j) \subseteq s(i)$. Thus, for all $i, j \in V$, either $S(i) \subseteq S(j)$ or $S(j) \subseteq S(i)$.

Let $T(G) = \{T_1, T_2, \cdots, T_g\}$ be a partition of V obtained by the equality relation on the sets of successors: for all $1 \leq i \leq g$, $u, v \in T_i$ if and only if $S(u) = S(v)$. The sets $T_1, \cdots, T_g$ are labeled in such a way that for all $1 \leq i < j \leq g$, $u \in T_i$ and $v \in T_j$ imply $S(u) \supset S(v)$.

The Most Successors (MS) policy is defined as follows: at any decision epoch, MS assigns the task having the largest set of successors among the enabled tasks to the fastest available processor. In other words, MS is a priority scheduling where the tasks in T_i have higher priority than those in T_{i+1}, $1 \leq i < g$. We will show that for all the profiles specified in Section 2, a policy is optimal for the class of interval-order graphs if and only if it is MS.

Theorem 3.1 *For any profile* $\{a_n, M_n\}_{n=1}^{\infty}$ *and any set of interval-ordered tasks* $G \in \mathcal{C}_{i.o}$,

$$\forall \pi \in \Psi : \qquad MS(G) \leq_{st} \pi(G). \tag{3.1}$$

Moreover, MS is the only policy that satisfies relation (3.1).

Before proceeding with the proof of the theorem, we define a majorization relation and present some lemmas.

Let $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ be two subgraphs of G obtained by successively deleting the vertices of G having no predecessor in G or in the previously obtained subgraphs. From the closure property, G^1 and G^2 are in $\mathcal{C}_{i.o}$. Let $T_i^j = T_i \cap V^j$, $j = 1, 2$, $i = 1, \cdots, g$. Graph G^1 is said to be majorized by G^2, referred to as $G^1 \prec_s G^2$, if and only if

$$\forall i,\ 1 \leq i \leq g : \qquad \sum_{k=1}^{i} |T_k^1| \leq \sum_{k=1}^{i} |T_k^2|,$$

where $|T_i^j|$ is the cardinality of set T_i^j, $j = 1, 2$, $i = 1, \cdots, g$.

Lemma 3.2 *Let* $G^1 = (V^1, E^1)$ *and* $G^2 = (V^2, E^2)$ *be two subgraphs of* $G \in \mathcal{C}_{i.o}$ *obtained by successively deleting the vertices of* G *having no predecessor in* G *or in the previously obtained subgraphs. If* $G^1 \prec_s G^2$ *then*

$$MS(G^1) \leq_{st} MS(G^2).$$

Proof. Owing to the memoryless property of the exponential distributions, we can consider a coupled processing model where all processors $1, \cdots, K$, whenever they are available, are continually executing tasks. When a completion occurs, and there is no task assigned to that processor, it corresponds to the completion of a fictitious task. When a task is assigned to a processor, it is assigned a running time equal to the remainder of the running time already underway at that processor. Thus, if tasks $u \in G^1$ and $v \in G^2$ are assigned to the same processor at some time, they have the same (remaining) running time.

We will show that in such a probability space

$$MS(G^1) \leq MS(G^2) \quad a.s. \tag{3.2}$$

Applying further Strassen's Theorem (cf. Lemma 2.1) will allow us to conclude the assertion of the lemma.

Denote by $\{c_n\}_{n=1}^{\infty}$ the (increasing) sequence of completion times of the tasks in G^1 and G^2 at the available processors under the MS policy. Let $\{d_n\}_{n=1}^{\infty}$ be the superposition of the sequences of the decision epochs of MS for G^1 and G^2 in such a probability space. More specifically, $\{d_n\}_{n=1}^{\infty}$ is the superposition of the sequences of profile modification times $\{a_n\}_{n=1}^{\infty}$ and of the task completion times $\{c_n\}_{n=1}^{\infty}$. Clearly, $d_1 = a_1 = 0$.

For $j = 1, 2$, and $n \geq 1$, let $G^j_{MS}(n) = (V^j(n), E^j(n))$ be the remaining graph of G^j at time d_n under MS in the coupled model. Denote by $T^j_i(n) = T_i \cap V^j(n)$, $j = 1, 2$, $1 \leq i \leq g$, and $n \geq 1$. We show that for all $n \geq 1$,

$$G^1_{MS}(n) \prec_s G^2_{MS}(n), \tag{3.3}$$

which immediately implies (3.2).

The relation (3.3) is proved by induction on n. For $n = 1$, it is trivial that

$$G^1_{MS}(1) = G^1 \prec_s G^2 = G^2_{MS}(1).$$

Assume that (3.3) holds for some $n \geq 1$.

Let there be m available processors at time d_n^+. Without loss of generality, we assume that the processors $1, 2, \cdots, m$ are available. Recall that by convention, these processors are ordered by there speed: $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$, and processor 1 is the fastest.

Under the MS policy, the tasks with the largest sets of successors are assigned to the fastest processors. Let $u_1, u_2, \cdots, u_{m_1}$ be the tasks of $G^1_{MS}(n)$ that are assigned to the processors $1, 2, \cdots, m_1$, respectively, under the MS policy. Similarly, let $v_1, v_2, \cdots, v_{m_2}$ be the tasks of $G^2_{MS}(n)$ that are assigned to the processors $1, 2, \cdots, m_2$, respectively, under the MS policy. By definition, $m \geq \max(m_1, m_2)$, and

$$S(u_1) \supseteq S(u_2) \supseteq \cdots \supseteq S(u_{m_1}), \qquad S(v_1) \supseteq S(v_2) \supseteq \cdots \supseteq S(v_{m_2}).$$

If $G^1_{MS}(n)$ is empty, then (3.3) trivially holds for $n + 1$. If the epoch d_{n+1} corresponds to a profile modification, then

$$G^1_{MS}(n+1) = G^1_{MS}(n) \prec_s G^2_{MS}(n) = G^2_{MS}(n+1),$$

so that relation (3.3) holds for $n+1$. Assume now that $G^1_{MS}(n)$ is not empty and that d_{n+1} corresponds to a completion at some processor, say processor h, $1 \leq h \leq \max(m_1, m_2)$.

Let u_h, if any, belong to $T_a^1(n)$, and v_h, if any, belong to $T_b^2(n)$. Tasks u_h and/or v_h are finished at time d_{n+1}. There are three cases to be investigated.

Case 1: $m_1 > m_2$ and $m_2 + 1 \leq h \leq m_1$.

In this case, only task u_h is finished. It is easy to see that

$$\forall i,\ 1 \leq i \leq g: \qquad \sum_{k=1}^{i} |T_k^1(n+1)| \leq \sum_{k=1}^{i} |T_k^1(n)| \leq \sum_{k=1}^{i} |T_k^2(n)| = \sum_{k=1}^{i} |T_k^2(n+1)|,$$

so that relation (3.3) holds for $n+1$.

Case 2: $h \leq \min(m_1, m_2)$.

In this case, both tasks u_h and v_h are finished. There are two subcases:

Case 2.1: $a \leq b$.

It is simple that

$$\forall i,\ 1 \leq i \leq a-1: \qquad \sum_{k=1}^{i} |T_k^1(n+1)| = \sum_{k=1}^{i} |T_k^1(n)| \leq \sum_{k=1}^{i} |T_k^2(n)| = \sum_{k=1}^{i} |T_k^2(n+1)|.$$

$$\forall i,\ a \leq i \leq b-1: \qquad \sum_{k=1}^{i} |T_k^1(n+1)| = -1 + \sum_{k=1}^{i} |T_k^1(n)| \leq -1 + \sum_{k=1}^{i} |T_k^2(n)|$$

$$= -1 + \sum_{k=1}^{i} |T_k^2(n+1)| \leq \sum_{k=1}^{i} |T_k^2(n+1)|.$$

$$\forall i,\ b \leq i \leq g: \qquad \sum_{k=1}^{i} |T_k^1(n+1)| = -1 + \sum_{k=1}^{i} |T_k^1(n)|$$

$$\leq -1 + \sum_{k=1}^{i} |T_k^2(n)| = \sum_{k=1}^{i} |T_k^2(n+1)|.$$

Therefore, relation (3.3) holds for $n+1$.

Case 2.2: $a > b$.

Observe first that

$$\forall i,\ 1 \leq i \leq b-1: \qquad \sum_{k=1}^{i} |T_k^1(n+1)| = \sum_{k=1}^{i} |T_k^1(n)| \leq \sum_{k=1}^{i} |T_k^2(n)| = \sum_{k=1}^{i} |T_k^2(n+1)|.$$

$$\forall i,\ a \leq i \leq g: \qquad \sum_{k=1}^{i} |T_k^1(n+1)| = -1 + \sum_{k=1}^{i} |T_k^1(n)|$$

$$\leq -1 + \sum_{k=1}^{i} |T_k^2(n)| = \sum_{k=1}^{i} |T_k^2(n+1)|.$$

We are going to examine the case $b \leq i \leq a-1$.

Let $1 \leq i_1, i_2 \leq g$ be the integers defined as follows:

$$T_1^1(n) = \cdots = T_{i_1-1}^1(n) = \emptyset, \qquad T_{i_1}^1(n) \neq \emptyset,$$

$$T_1^2(n) = \cdots = T_{i_2-1}^2(n) = \emptyset, \qquad T_{i_2}^2(n) \neq \emptyset.$$

Note that i_1 and i_2 are uniquely defined. Under the inductive assumption, one has that

$$\sum_{k=1}^{i_1} |T_k^2(n)| \geq \sum_{k=1}^{i_1} |T_k^1(n)| > 0.$$

Thus, $i_1 \geq i_2$.

For $1 \leq i, k \leq g$, let $n_k(i) = |T_k \cap S(T_i)|$, where $S(T_i) = S(v)$ for some $v \in T_i$. Note that $n_k(i) = 0$ if $k \leq i$. Note also that $n_k(i) \geq n_k(i+1)$. According to the definitions, $n_k(i_j)$ is the number of *unabled* tasks of $T_k^j(n)$ in $G_{MS}^j(n)$, $j = 1, 2$, $1 \leq k \leq g$, where a task is unabled if all of its predecessors are not finished.

For all i, $b \leq i \leq a-1$, we have $u_h \notin T_1 \cup T_2 \cup \cdots \cup T_i$, so that there are at most $h-1$ enabled tasks $u_1, u_2, \cdots, u_{h-1}$ in $T_1^1(n) \cup T_2^1(n) \cup \cdots \cup T_i^1(n)$. Thus

$$\sum_{k=1}^{i} |T_k^1(n)| \leq h - 1 + \sum_{k=1}^{i} n_k(i_1).$$

On the other hand, $v_h \in T_h$ implies that the h enabled tasks $v_1, v_2, \cdots, v_h$ are in $T_1^2(n) \cup T_2^2(n) \cup \cdots \cup T_i^2(n)$. Hence,

$$\sum_{k=1}^{i} |T_k^2(n)| \geq h + \sum_{k=1}^{i} n_k(i_2).$$

The above two inequalities together with the fact that $n_k(i_1) \leq n_k(i_2)$ entail that

$$\sum_{k=1}^{i} |T_k^1(n)| \leq -1 + \sum_{k=1}^{i} |T_k^2(n)|.$$

Therefore, for all i, $b \leq i \leq a-1$,

$$\sum_{k=1}^{i} |T_k^1(n+1)| = \sum_{k=1}^{i} |T_k^1(n)| \leq -1 + \sum_{k=1}^{i} |T_k^2(n)| = \sum_{k=1}^{i} |T_k^2(n+1)|,$$

which completes the proof of (3.3) for $n+1$ in Case 2.2.

Case 3: $m_1 < m_2$ and $m_1 + 1 \leq h \leq m_2$.

In this case, only task v_h is finished.

It is clear that for all $1 \leq i \leq b-1$,

$$\sum_{k=1}^{i} |T_k^1(n+1)| = \sum_{k=1}^{i} |T_k^1(n)| \leq \sum_{k=1}^{i} |T_k^2(n)| = \sum_{k=1}^{i} |T_k^2(n+1)|.$$

Since $m \geq m_2 \geq m_1 + 1$, and that only m_1 tasks of $G_{MS}^1(n)$ are assigned to the available processors, we obtain that for all $i \geq b$,

$$\sum_{k=1}^{i} |T_k^1(n)| \leq m_1 + \sum_{k=1}^{i} n_k(i_1). \tag{3.4}$$

On the other hand, $v_h \in T_b$ implies that the h enabled tasks $v_1, v_2, \cdots, v_h$ are in $T_1^2(n) \cup T_2^2(n) \cup \cdots \cup T_i^2(n)$. Hence, for all $i \geq b$,

$$\sum_{k=1}^{i} |T_k^2(n)| \geq h + \sum_{k=1}^{i} n_k(i_2) \geq m_1 + 1 + \sum_{k=1}^{i} n_k(i_2) \geq m_1 + 1 + \sum_{k=1}^{i} n_k(i_1). \tag{3.5}$$

Inequalities (3.4) and (3.5) entail that

$$\sum_{k=1}^{i} |T_k^1(n)| \leq -1 + \sum_{k=1}^{i} |T_k^2(n)|.$$

Therefore, for all $i \geq b$,

$$\sum_{k=1}^{i} |T_k^1(n+1)| = \sum_{k=1}^{i} |T_k^1(n)| \leq -1 + \sum_{k=1}^{i} |T_k^2(n)| = \sum_{k=1}^{i} |T_k^2(n+1)|,$$

so that (3.3) holds for $n+1$ in Case 3.

Therefore, by induction, relation (3.3) holds for all $n \geq 1$. Consequently, $MS(G^1) \leq MS(G^2)$ *a.s.* in that probability space. Using further Strassen's Theorem concludes the proof. ∎

Lemma 3.3 *Let $G \in \mathcal{C}_{i.o}$ be a task graph. Let $\pi, \rho \in \Psi$ be two policies which follow the MS rule all the time except at the first decision epoch. At the first decision epoch, policy π*

assigns tasks $v_1, v_2, \cdots, v_k$ *to the available processors* $q_1, q_2, \cdots, q_k$, *respectively, with, by assumption,* $S(v_1) \supseteq S(v_2) \supseteq \cdots \supseteq S(v_k)$. *Whereas policy* ρ *assigns tasks* $v_1, v_2, \cdots, v_k$ *to the processors* $q_{\chi(1)}, q_{\chi(2)}, \cdots, q_{\chi(k)}$, *respectively, where* χ *is the permutation on* $\{1, 2, \cdots, k\}$ *such that* $q_{\chi(1)} < q_{\chi(2)} < \cdots < q_{\chi(k)}$. *Then,*

$$\rho(G) \leq_{st} \pi(G).$$

Proof. If $q_1 < q_2 < \cdots < q_k$, then ρ is identical to π so that the assertion trivially holds. Assume that there are integers $1 \leq i < j \leq k$ such that $q_i > q_j$. Let χ_1 be the permutation on $\{1, 2, \cdots, k\}$ defined as follows:

$$\chi_1(i) = j, \quad \chi_1(j) = i, \quad \chi_1(n) = n, \quad \forall n \in \{1, 2, \cdots, k\} - \{i, j\}.$$

Let $\rho_1 \in \Psi$ be a policy which follows the MS rule all the time except at the first decision epoch. At the first decision epoch, policy ρ_1 assigns tasks $v_1, v_2, \cdots, v_k$ to the processors $q_{\chi_1(1)}, q_{\chi_1(2)}, \cdots, q_{\chi_1(k)}$, respectively. We will show that

$$\rho_1(G) \leq_{st} \pi(G). \tag{3.6}$$

We couple the running times on processors $q_1, \cdots, q_k$ in such a way that under both policies π and ρ_1, the running time on processor q_n starting from time 0 is τ_n, $1 \leq n \leq k$. In such a coupled model, the second decision epoch occur at the same time under both policies. Let this time epoch be fixed, referred to as d_2,

$$d_2 = \min(a_2, \tau_1, \cdots, \tau_k).$$

If this decision epoch corresponds to a profile modification, i.e., $d_2 = a_2$, then $\rho_1(G) =_{st} \pi(G)$.

Assume now that the second decision epoch corresponds to a task completion. It is verified that

$$\pi(G) = d_2 + \frac{1}{\lambda} \sum_{n=1}^{k} \alpha_{q_n} MS(G - \{v_n\}), \tag{3.7}$$

$$\rho_1(G) = d_2 + \frac{1}{\lambda} \sum_{n=1}^{k} \alpha_{q_{\chi_1(n)}} MS(G - \{v_n\}), \tag{3.8}$$

where $\lambda = \sum_{n=1}^{k} \alpha_{q_n}$. Moreover, the random variables $MS(G - \{v_n\})$, $1 \leq n \leq k$, are mutually independent.

According to Lemma 3.2,

$$MS(G - \{v_i\}) \leq_{st} MS(G - \{v_j\}).$$

Thus, the fact that $q_i > q_j$ (so that $\alpha_{q_i} \leq \alpha_{q_j}$) implies that

$$\alpha_{q_j} MS(G - \{v_i\}) + \alpha_{q_i} MS(G - \{v_j\}) \leq_{st} \alpha_{q_i} MS(G - \{v_i\}) + \alpha_{q_j} MS(G - \{v_j\}),$$

so that

$$\rho_1(G) \leq_{st} \pi(G).$$

Unconditioning d_2 in the above relation yields (3.6).

Consider now policy ρ_1. If $\chi_1 = \chi$, then we are done. Otherwise, there are integers $1 \leq i' < j' \leq k$ such that $q_{\chi_1(i')} > q_{\chi_1(j')}$. Let χ_2 be the permutation on $\{1, 2, \cdots, k\}$ defined as follows:

$$\chi_2(i') = \chi_1(j'), \quad \chi_2(j') = \chi_1(i'), \quad \chi_2(n) = \chi_1(n), \quad \forall n \in \{1, 2, \cdots, k\} - \{i', j'\}.$$

Let $\rho_2 \in \Psi$ be a policy which follows the MS rule all the time except at the first decision epoch. At the first decision epoch, policy ρ_1 assigns tasks $v_1, v_2, \cdots, v_k$ to the processors $q_{\chi_2(1)}, q_{\chi_2(2)}, \cdots, q_{\chi_2(k)}$, respectively. As above, we can show that

$$\rho_2(G) \leq_{st} \rho_1(G).$$

Repeating this procedure for at most $k(k-1)/2$ times yields finally policy ρ such that

$$\rho(G) \leq_{st} \cdots \leq_{st} \rho_2(G) \leq_{st} \rho_1(G) \leq_{st} \pi(G).$$

■

Lemma 3.4 *Let $G \in C_{i.o}$ be a task graph. Let $\pi, \rho \in \Psi$ be two policies which follow the MS rule all the time except at the first decision epoch. At the first decision epoch, policy π assigns tasks $v_1, \cdots, v_{k-1}, v_k$ to the available processors $q_1, \cdots, q_{k-1}, q_k$, respectively, whereas ρ assigns tasks $v_1, \cdots, v_{k-1}, v$ to the processors $q_1, \cdots, q_{k-1}, q_k$, respectively. If $v \notin \{v_1, \cdots, v_k\}$ and if $S(v) \supseteq S(v_k)$, then*

$$\rho(G) \leq_{st} \pi(G).$$

Proof. The proof is similar to that of Lemma 3.3, and uses Lemma 3.2 to obtain $MS(G - \{v\}) \leq_{st} MS(G - \{v_k\})$. The detailed proof is left to the interested reader. ■

Lemma 3.5 *Let $G \in C_{i.o}$ be a task graph. Let $\pi, \rho \in \Psi$ be two policies which follow the MS rule all the time except at the first decision epoch. At the first decision epoch, policy π assigns tasks $v_1, \cdots, v_{k-1}, v_k$ to the available processors $q_1, \cdots, q_{k-1}, q_k$, respectively, whereas ρ assigns tasks $v_1, \cdots, v_{k-1}, v_k$ to the processors $q_1, \cdots, q_{k-1}, q$, respectively. If $q \notin \{q_1, \cdots, q_k\}$ and if $q < q_k$, then*

$$\rho(G) \leq_{st} \pi(G).$$

Proof. Let $\tau_1(\pi), \cdots, \tau_k(\pi)$ be the mutually independent random variables of the running times of the tasks $v_1, v_2, \cdots, v_k$ starting from time 0 under policy π, respectively, where $\tau_i(\pi)$ has an exponential distribution with parameter α_{q_i}, $1 \leq i \leq k$.

Let $\tau_1(\rho), \cdots, \tau_{k-1}(\rho), \tau(\rho)$ be the mutually independent random variables of the running times of the tasks $v_1, v_2, \cdots, v_{k-1}, v_k$ starting from time 0 under policy ρ, respectively, where $\tau_i(\rho)$ is identical to $\tau_i(\pi)$ in law, $1 \leq i \leq k-1$, and $\tau(\rho)$ has an exponential distribution with parameter α_q.

Since $\alpha_q \geq \alpha_{q_k}$, we have $\tau(\rho) \leq_{st} \tau_k(\pi)$. This last inequality together with the mutual independence of the above defined random variables allow us to use Strassen's Theorem and to obtain random variables

$$\hat{\tau}_1(\pi), \cdots, \hat{\tau}_k(\pi), \quad \text{and} \quad \hat{\tau}_1(\rho), \cdots, \hat{\tau}_{k-1}(\rho), \hat{\tau}(\rho)$$

on a common probability space $\hat{P}$ such that

$$(\hat{\tau}_1(\pi), \cdots, \hat{\tau}_k(\pi)) =_{st} (\tau_1(\pi), \cdots, \tau_k(\pi)),$$

$$(\hat{\tau}_1(\rho), \cdots, \hat{\tau}_{k-1}(\rho), \hat{\tau}(\rho)) =_{st} (\tau_1(\rho), \cdots, \tau_{k-1}(\rho), \tau(\rho)).$$

and moreover,

$$\hat{\tau}_1(\pi) = \hat{\tau}_1(\rho), \quad \cdots, \quad \hat{\tau}_{k-1}(\pi) = \hat{\tau}_{k-1}(\rho), \quad \text{and} \quad \hat{\tau}_k(\pi) \geq \hat{\tau}(\rho), \quad a.s.$$

Let

$$X = \min(a_2, \hat{\tau}_1(\rho), \cdots, \hat{\tau}_{k-1}(\rho), \hat{\tau}(\rho)).$$

If $X < \hat{\tau}(\rho)$, then $\hat{\rho}(G) = \hat{\pi}(G)$, $a.s.$ If $X = \hat{\tau}(\rho)$, then task v_k is finished under ρ but not under π. Owing to Lemma 3.2, we get $MS(G - \{v_k\}) \leq_{st} MS(G)$. As $MS(G)$ and $MS(G - \{v_k\})$ are independent of X, we can construct on the same probability space $\hat{P}$ random variables $\widehat{MS}(G)$ and $\widehat{MS}(G - \{v_k\})$ such that

$$\widehat{MS}(G) =_{st} MS(G), \quad \widehat{MS}(G - \{v_k\}) =_{st} MS(G - \{v_k\}),$$

and

$$\widehat{MS}(G - \{v_k\}) \leq \widehat{MS}(G), \quad a.s.$$

Thus, in such a probability space,

$$\hat{\rho}(G) = X + \widehat{MS}(G - \{v_k\}) \leq X + \widehat{MS}(G) = \hat{\pi}(G), \quad a.s.$$

Therefore, in both cases, we have $\hat{\rho}(G) \leq \hat{\pi}(G)$, $a.s.$ Applying again Strassen's Theorem readily yields $\rho(G) \leq_{st} \pi(G)$. ∎

Lemma 3.6 *Let $G \in \mathcal{C}_{i.o}$ be a task graph. Let $\pi \in \Psi$ be a policy which follows the MS rule all the time except at the first decision epoch. Then,*

$$MS(G) \leq_{st} \pi(G).$$

Proof. The assertion of the lemma is an easy consequence of Lemmas 3.3, 3.4 and 3.5. Indeed, in order to stochastically minimize the makespan,

- due to Lemma 3.3, when a given set of tasks is to be assigned to a given set of processors, the tasks with larger sets of successors should be assigned to faster processors;
- due to Lemma 3.4, to any given set of processors, the enabled tasks with the largest sets of successors should be assigned;
- due to Lemma 3.5, for any given set of tasks, the fastest processors should be used.

Repeating these arguments, one can pathwisely improve policy π until all the task assignments at the first decision epoch agree with the MS policy. ∎

We are now in a position to prove Theorem 3.1.

Proof of Theorem 3.1. Fix profile $\{a_n, M_n\}_{n=1}^{\infty}$, the task graph $G \in \mathcal{C}_{i.o}$, and the policy $\pi \in \Psi$.

Due to the assumptions that only a finite number of profile changes are made during any finite time interval, and that at least one processor is available at any time, the tasks are run at rate at least α_K at any time. Thus, the time to finish the tasks in the finite task graph is finite, so that the number of decisions made by π for processing these tasks is finite. Denote by $\{d_n\}_{n=1}^{D}$ the finite sequence of decision epochs of π for finishing tasks in G.

In order to show the theorem, we construct a finite sequence of policies $\{\pi_n\}_{n=1}^{D}$. For $1 \leq n \leq D$, policy π_n takes exactly the same scheduling decisions as π at times $d_1, d_2, \cdots, d_{n-1}$, i.e., the same tasks are assigned to the same processors at these time instants under policies π and π_n. From time d_n, policy π_n follows the MS rule. By definition, π_1 is an MS policy.

We use forward induction on n to show that for all n, $1 \leq n \leq D$,

$$\pi_n(G) \leq_{st} \pi(G). \tag{3.9}$$

For $1 \leq n \leq D$, let G_n be the remaining task graph of G at time d_n under π. It is clear that $G_1 = G$ and G_D has a single vertex. Applying Lemma 3.6 (or, more precisely, Lemma 3.5) to the task graph G_D implies that

$$\pi_D(G) = d_D + MS(G_D) \leq_{st} d_D + \pi(G_D) = \pi(G).$$

Therefore, (3.9) holds for $n = D$. Assume that relation (3.9) holds for some $2 \leq n \leq D$. Then

$$\pi_{n-1}(G) = d_{n-1} + MS(G_{n-1}) \leq_{st} d_{n-1} + \pi_n(G_{n-1}) = \pi_n(G),$$

where we used Lemma 3.6 to obtain the inequality. Thus, by induction, relation (3.9) holds for all $1 \leq n \leq D$. Taking $n = 1$ immediately implies relation (3.1).

Moreover, it is not difficult to see that MS is the only policy that satisfies relation (3.1). Indeed, from the proofs of Lemmas 3.3, 3.4 and 3.5, one can show that if a policy π violates the MS rule, then, with positive probability $MS(G) < \pi(G)$. ∎

4 Stochastic Profile Scheduling of In-Forest

Let $G = (V, E) \in \mathcal{C}_{i.f}$ be an in-forest. A task of in-forest G is enabled if and only if it is a leaf of G. The level of a vertex $i \in V$, denoted by $l(i)$, is the distance from it to the root of the tree it appears. The level of the roots is zero by convention. Let $L(G)$, or simply L when there is no ambiguity, denote the level of the in-forest G, i.e., $L(G) = \max_{i \in V} l(i)$. Let $N_i(G)$ be the number of tasks of G at level i, $i \geq 0$. Clearly $N_i(G) = 0$ for all $i > L(G)$.

The Highest Level first (HL) policy is defined as follows: at any decision epoch, HL assigns the task at the highest level among the enabled tasks to the fastest available processor. Hence, HL is a priority scheduling where the tasks at level l have higher

priority than those at level $l-1$, $1 \leq l \leq L$. We will show that for all the profiles which have at most two available processors (among the $K \geq 2$ processors) at any time, a policy is optimal for the class of in-forests if and only if it is HL.

Theorem 4.1 *For any profile* $\{a_n, M_n\}_{n=1}^{\infty}$ *such that* $|M_n| \leq 2$, $n \geq 1$, *and for any in-forest* $G \in \mathcal{C}_{i.f}$,

$$\forall \pi \in \Psi : \qquad HL(G) \leq_{st} \pi(G). \tag{4.1}$$

Moreover, HL is the only policy that satisfies relation (4.1).

Note that the above theorem holds only when $|M_n| \leq 2$. Simple counterexamples can be found when there are three processors (see [2]).

In order to prove the theorem, we need to define another majorization relation (referred to as "flatter than" in Chandy and Reynolds [2]) and to present some lemmas.

Let $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ be two in-forests. Forest G^1 is said to be flatter than G^2, denoted by $G^1 \prec_f G^2$, if and only if

$$\forall i,\ i \geq 0 : \qquad \sum_{k \geq i} N_k(G^1) \leq \sum_{k \geq i} N_k(G^2).$$

Note that $G^1 \prec_f G^2$ implies $L(G^1) \leq L(G^2)$.

Lemma 4.2 *For any profile* $\{a_n, M_n\}_{n=1}^{\infty}$ *such that* $|M_n| \leq 2$, $n \geq 1$, *and for any in-forests* $G^1, G^2 \in \mathcal{C}_{i.f}$, *if* $G^1 \prec_f G^2$ *then*

$$HL(G^1) \leq_{st} HL(G^2).$$

Proof. The lemma is shown by combining the ideas of the proofs of Theorem 6 in [2] and Lemma 3.2.

As in Lemma 3.2, we consider a coupled processing model where all processors $1, \cdots, K$, whenever they are available, are continually executing tasks. When a completion occurs, and there is no task assigned to that processor, it corresponds to the completion of a fictitious task. When a task is assigned to a processor, it is assigned a running time equal to the remainder of the running time already underway at that processor. Thus, if tasks $u \in G^1$ and $v \in G^2$ are assigned to the same processor at some time, they have the same (remaining) running time.

We will show that in such a probability space

$$HL(G^1) \leq HL(G^2) \quad a.s. \tag{4.2}$$

Denote by $\{c_n\}_{n=1}^{\infty}$ the (increasing) sequence of completion times of the tasks in G^1 and G^2 at the available processors under the HL policy. Let $\{d_n\}_{n=1}^{\infty}$ be the superposition of the sequences of profile modification times $\{a_n\}_{n=1}^{\infty}$ and of the task completion times $\{c_n\}_{n=1}^{\infty}$. Clearly, $d_1 = a_1 = 0$.

For $j = 1, 2$, and $n \geq 1$, let $G^j_{HL}(n) = (V^j(n), E^j(n))$ be the remaining graph of G^j at time d_n under HL in the coupled model. Denote by $N^j_i(n) = N_i(G^j_{HL}(n))$, $j = 1, 2$, $i \geq 0$, and $n \geq 1$. We show by induction that for all $n \geq 1$,

$$G^1_{HL}(n) \prec_f G^2_{HL}(n). \tag{4.3}$$

which immediately implies (4.2).

For $n = 1$, it is trivial that

$$G^1_{HL}(1) = G^1 \prec_f G^2 = G^2_{HL}(1).$$

Assume that (4.3) holds for some $n \geq 1$.

Let there be $m \leq 2$ available processors at time d_n^+. Without loss of generality, we assume that the processors $1, \cdots, m$ are available. Recall that processor 1 is the fastest.

Under the HL policy, the enabled tasks at the highest levels are assigned to the fastest processors. Let $u_1, \cdots, u_{m_1}$ (resp. $v_1, \cdots, v_{m_2}$) be the tasks of $G^1_{HL}(n)$ (resp. $G^2_{HL}(n)$) that are assigned to the processors $1, \cdots, m_1$ (resp. $1, \cdots, m_2$) under the HL policy. By definition, $m \geq \max(m_1, m_2)$, and

$$l(u_1) \geq \cdots \geq l(u_{m_1}), \qquad l(v_1) \geq \cdots \geq l(v_{m_2}).$$

If $G^1_{HL}(n)$ is empty, then (4.3) trivially holds for $n+1$. If the epoch d_{n+1} corresponds to a profile modification, then

$$G^1_{HL}(n+1) = G^1_{HL}(n) \prec_f G^2_{HL}(n) = G^2_{HL}(n+1),$$

so that relation (4.3) holds for $n+1$. Assume now that $G^1_{HL}(n)$ is not empty and that d_{n+1} corresponds to a completion at some processor, say processor h, $1 \leq h \leq \max(m_1, m_2)$.

Let u_h, if any, be at level l_1, and v_h, if any, be at level l_2. Tasks u_h and/or v_h are finished at time d_{n+1}. There are four possible cases.

Case 1: $m_1 \geq 1$, $m_2 \geq 1$, $h = 1$.

In this case, both tasks u_1 and v_1 are finished. Under the inductive assumption,

$$l_1 = L(G^1_{HL}(n)) \leq L(G^2_{HL}(n)) = l_2.$$

Since u_1 and v_1 are at the highest level of $G^1_{HL}(n)$ and $G^2_{HL}(n)$, respectively, we obtain that

$$\forall i,\ l_1 + 1 \leq i \leq l_2: \quad \sum_{k \geq i} N^1_k(n+1) = \sum_{k \geq i} N^1_k(n) = 0 \leq -1 + \sum_{k \geq i} N^2_k(n) = \sum_{k \geq i} N^2_k(n+1).$$

$$\forall i,\ 0 \leq i \leq l_1: \quad \sum_{k \geq i} N^1_k(n+1) = -1 + \sum_{k \geq i} N^1_k(n) \leq -1 + \sum_{k \geq i} N^2_k(n) = \sum_{k \geq i} N^2_k(n+1).$$

Therefore, relation (4.3) holds for $n+1$ in Case 1.

Case 2: $m_1 = 2$, $m_2 = 1$ and $h = 2$.

In this case, only task u_2 is finished. It is easy to see that

$$\forall i,\ i \geq 0: \quad \sum_{k \geq i} N^1_k(n+1) \leq \sum_{k \geq i} N^1_k(n) \leq \sum_{k \geq i} N^2_k(n) = \sum_{k \geq i} N^2_k(n+1),$$

so that relation (4.3) holds for $n+1$ in Case 2.

Case 3: $m_1 = 1$, $m_2 = 2$ and $h = 2$.

In this case, there are two available processors at time d_n and only task v_2 is finished. Thus, $G^1_{HL}(n)$ is a chain, i.e., $N^1_i(n) = 1$ for all $0 \leq i \leq L(G^1_{HL}(n))$. At the same time, $l(v_1) \geq l(v_2)$ so that $N^2_{l_2}(n) \geq 2$, which implies that $L(G^1_{HL}(n)) \leq L(G^2_{HL}(n)) = L(G^2_{HL}(n+1))$. Therefore, relation (4.3) holds for $n+1$ in Case 3.

Case 4: $m_1 = m_2 = 2$, $h = 2$.

In this case, both tasks u_2 and v_2 are finished. However, the tasks u_1 and v_1 remain unfinished. Thus,

$$L(G^1_{HL}(n+1)) = L(G^1_{HL}(n)) \leq L(G^2_{HL}(n)) = L(G^2_{HL}(n+1)).$$

There are two subcases to be considered:

Case 4.1: $l_1 \geq l_2$.

It is simple that

$$\forall i,\ i \geq l_2 + 1: \qquad \sum_{k \geq i} N_k^1(n+1) \leq \sum_{k \geq i} N_k^1(n) \leq \sum_{k \geq i} N_k^2(n) = \sum_{k \geq i} N_k^2(n+1).$$

$$\forall i,\ 0 \leq i \leq l_2: \qquad \sum_{k \geq i} N_k^1(n+1) = -1 + \sum_{k \geq i} N_k^1(n) \leq -1 + \sum_{k \geq i} N_k^2(n) = \sum_{k \geq i} N_k^2(n+1).$$

Therefore, relation (4.3) holds for $n+1$.

Case 4.2: $l_1 < l_2$.

Since u_2 is the second highest leaf in $G_{HL}^1(n)$, we get that

$$\forall i,\ l_1 + 1 \leq i \leq L(G_{HL}^1(n+1)): \qquad N_i^1(n+1) = N_i^1(n) = 1.$$

Thus,

$$\forall i,\ i > L(G_{HL}^1(n+1)): \qquad \sum_{k \geq i} N_k^1(n+1) = 0 \leq \sum_{k \geq i} N_k^2(n+1).$$

$$\forall i,\ l_1 + 1 \leq i \leq L(G_{HL}^1(n+1)): \qquad \sum_{k \geq i} N_k^1(n+1) = L(G_{HL}^1(n+1)) - i + 1$$

$$\leq L(G_{HL}^2(n+1)) - i + 1 \leq \sum_{k \geq i} N_k^2(n+1).$$

$$\forall i,\ 0 \leq i \leq l_1: \qquad \sum_{k \geq i} N_k^1(n+1) = -1 + \sum_{k \geq i} N_k^1(n)$$

$$\leq -1 + \sum_{k \geq i} N_k^2(n) = \sum_{k \geq i} N_k^2(n+1).$$

Therefore, relation (4.3) holds for $n+1$, which completes the proof of Case 4.2.

Hence, by induction, relation (4.3) holds for all $n \geq 1$. Consequently, $HL(G^1) \leq HL(G^2)$ *a.s.* in that probability space. Applying further Strassen's Theorem allows us to conclude. ∎

Lemma 4.3 *Let $\{a_n, M_n\}_{n=1}^{\infty}$ be a profile such that $|M_n| \leq 2$, $n \geq 1$, and let $G \in \mathcal{C}_{i.f}$ be an in-forest. Assume that $M_1 = \{q_1, q_2\}$ and $q_1 < q_2$. Let $\pi, \rho \in \Psi$ be two policies which follow the HL rule all the time except at the first decision epoch. At time $a_1 = 0$, policy*

π assigns tasks v_1, v_2 to the available processors q_2 and q_1, respectively. Whereas policy ρ assigns tasks v_1, v_2 to processors q_1 and q_2, respectively. If $l(v_1) \geq l(v_2)$, then

$$\rho(G) \leq_{st} \pi(G).$$

Proof. The proof can be done in an analogous way to the proof of Lemma 3.3 and uses Lemma 4.2. The detailed proof is omitted. ∎

Lemma 4.4 *Let $\{a_n, M_n\}_{n=1}^{\infty}$ be a profile such that $|M_n| \leq 2$, $n \geq 1$, and let $G \in C_{i.f}$ be an in-forest. Let $\pi, \rho \in \Psi$ be two policies which follow the HL rule all the time except at the first decision epoch. At time $a_1 = 0$, policy π assigns tasks v_1, v_k to the available processors q_1, q_k, respectively, $1 \leq k \leq 2$. Policy ρ differs from π in the assignment of task v_k: another enabled task $v \notin \{v_1, \cdots, v_k\}$ instead of v_k is assigned to processor q_k, $1 \leq k \leq 2$. If $l(v) \geq l(v_k)$, then*

$$\rho(G) \leq_{st} \pi(G).$$

Proof. The proof is similar to that of Lemma 3.3, and uses Lemma 4.2 to obtain $HL(G - \{v\}) \leq_{st} HL(G - \{v_k\})$. The detailed proof is omitted. ∎

Lemma 4.5 *Let $\{a_n, M_n\}_{n=1}^{\infty}$ be a profile such that $|M_n| \leq 2$, $n \geq 1$, and let $G \in C_{i.f}$ be an in-forest. Let $\pi, \rho \in \Psi$ be two policies which follow the HL rule all the time except at the first decision epoch. At time $a_1 = 0$, $M_1 = \{q_1, q_2\}$ and policy π assigns a single task v to processor q_2 whereas ρ assigns v to processor q_1. If $q_1 < q_2$, then*

$$\rho(G) \leq_{st} \pi(G).$$

Proof. The proof is easily obtained by mimicking the proof of Lemma 3.5 and by using Lemma 4.2 instead of Lemma 3.2. The detailed proof is omitted. ∎

Lemma 4.6 *Let $\{a_n, M_n\}_{n=1}^{\infty}$ be a profile such that $|M_n| \leq 2$, $n \geq 1$, and let $G \in C_{i.f}$ be an in-forest. Let $\pi \in \Psi$ be a policy which follows the HL rule all the time except at the first decision epoch. Then,*

$$HL(G) \leq_{st} \pi(G).$$

Proof. The assertion of the lemma can be simply proved by pathwisely improving policy π using Lemmas 4.3, 4.4 and 4.5, until all the task assignments at the first decision epoch agree with the HL policy. The detailed proof is omitted. ∎

Proof of Theorem 4.1. The proof of Theorem 4.1 is now straightforward using Lemma 4.6. See the proof of Theorem 3.1 for details. ∎

5 Stochastic Profile Scheduling of Out-Forest

Let $G = (V, E) \in \mathcal{C}_{o.f}$ be an out-forest. A task of out-forest G is enabled if and only if it is a root of G. Vertex $v \in V$ and all its successors is a subtree of G, denoted by $T_G(v)$ or simply $T(v)$ when there is no ambiguity. We denote by $|T(v)|$ the size of $T(v)$, i.e., the number of vertices in $T(v)$.

The Largest Tree (LT) policy is defined as follows: at any decision epoch, LT assigns the task v whose subtree $T(v)$ is the largest among all the subtrees to the fastest available processor. Hence, LT is a priority scheduling where task u has higher priority than task v if $|T(u)| > |T(v)|$.

In general, policy LT is not optimal within the class of out-forests $\mathcal{C}_{o.f}$. Counterexamples are provided in [3]. However, we will show that within the classes of *uniform* and *r-uniform* out-forests (introduced in Coffman and Liu [3]), a policy is optimal if and only if it is LT.

Let $T_1, T_2 \in \mathcal{C}_{o.f}$ be two out-trees. The out-tree T_1 is said to *embed* the out-tree T_2, or T_2 is *embedded* in T_1, denoted by $T_1 \succ_e T_2$ or $T_2 \prec_e T_1$, if T_2 is isomorphic to a subgraph of T_1. Formally, T_1 embeds T_2 if there exists an injective function f from T_2 into T_1 such that $\forall u, v \in T_2$, $v \in s(u)$ implies $f(v) \in s(f(u))$. The function f is called an *embedding function*.

Let r_1 and r_2 be the roots of the out-trees T_1 and T_2, respectively. If $T_1 \succ_e T_2$ and if there is an embedding function f such that $f(r_2) = r_1$, then f is a *root-embedding function*, and we write $T_1 \succ_r T_2$ or $T_2 \prec_r T_1$.

An out-forest $G \in \mathcal{C}_{o.f}$ is said to be *uniform* (respectively *r-uniform*) if all its subtrees $\{T(v), v \in G\}$ can be ordered by the embedding (respectively root-embedding) relation. The class of uniform (respectively r-uniform) forests is denoted by $\mathcal{C}_{u.o.f}$ (respectively

$\mathcal{C}_{r.o.f}$). It is clear that $\mathcal{C}_{r.o.f} \subset \mathcal{C}_{u.o.f} \subset \mathcal{C}_{o.f}$.

The embedding relation is extended to uniform out-forests as follows. Let $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ be two uniform out-forests. Assume that the vertices of G^1 and G^2 are indexed in such a way that

$$T_{G^1}(1) \succ_e T_{G^1}(2) \succ_e \cdots \succ_e T_{G^1}(|V^1|).$$

$$T_{G^2}(1) \succ_e T_{G^2}(2) \succ_e \cdots \succ_e T_{G^2}(|V^2|).$$

Out-forest G^1 is embedded in G^2, referred to as $G^1 \prec_e G^2$, if and only if

$$|V^1| \leq |V^2|, \quad \text{and} \quad \forall i,\ 1 \leq i \leq |V^1| : \quad T_{G^1}(i) \prec_e T_{G^2}(i).$$

Similarly, $G^1 \prec_r G^2$ if and only if

$$T_{G^1}(1) \succ_r T_{G^1}(2) \succ_r \cdots \succ_r T_{G^1}(|V^1|),$$

$$T_{G^2}(1) \succ_r T_{G^2}(2) \succ_r \cdots \succ_r T_{G^2}(|V^2|),$$

$$|V^1| \leq |V^2|, \quad \text{and} \quad \forall i,\ 1 \leq i \leq |V^1| : \quad T_{G^1}(i) \prec_r T_{G^2}(i).$$

Consider the optimal scheduling within the class of uniform out-forests. We will show that LT policy is optimal if the profile is such that at any time there are at most two available processors.

Theorem 5.1 *For any profile $\{a_n, M_n\}_{n=1}^{\infty}$ such that $|M_n| \leq 2$, $n \geq 1$, and for any uniform out-forest $G \in \mathcal{C}_{u.o.f}$,*

$$\forall \pi \in \Psi : \qquad LT(G) \leq_{st} \pi(G). \tag{5.1}$$

Moreover, LT is the only policy that satisfies relation (5.1).

We use the same scheme as in the previous two sections to prove Theorem 5.1. For sake of simplicity, we will not provide the detailed proof. Rather, we state some lemmas and point out how they can be proved and how they are used to prove the theorem. We have first the following lemma.

Lemma 5.2 *Let $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ be two subforests of a uniform out-forest $G \in \mathcal{C}_{u.o.f}$ obtained by successively deleting the roots of G or of the previously obtained subgraphs. For any profile $\{a_n, M_n\}_{n=1}^{\infty}$ such that $|M_n| \leq 2$, $n \geq 1$, if $G^1 \prec_e G^2$ then*

$$LT(G^1) \leq_{st} LT(G^2).$$

The proof of the above lemma is analogous to the proof of Theorem 1 of Coffman and Liu [3]. It can be shown by induction on the decision epochs of the coupled model of G^1 and G^2 under the LT policy. Although the proof of Theorem 1 in [3] was given for constant profile and parallel identical processors, the fact that both graphs G^1 and G^2 are scheduled by the LT policy allows us to use the same argument. The detailed proof is left to the interested reader.

Based on this lemma, we can show (see the proofs of Lemmas 3.3, 3.4 and 3.5) that

Lemma 5.3 *Let $\{a_n, M_n\}_{n=1}^{\infty}$ be a profile such that $|M_n| \leq 2$, $n \geq 1$, and let $G \in \mathcal{C}_{u.o.f}$ be a uniform out-forest. Let $\pi, \rho \in \Psi$ be two policies which follow the LT rule all the time except at the first decision epoch. At time $a_1 = 0$, policy π assigns tasks v_1, v_k to the available processors q_1, q_k, respectively, whereas ρ assigns tasks u_1, u_k to the available processors p_1, p_k, respectively, $1 \leq k \leq 2$.*

(a) *If*

- $M_1 = \{q_1, q_2\}$, $q_1 > q_2$, $p_1 = q_2$, $p_2 = q_1$, *and*
- $k = 2$, $u_1 = v_1$, $u_2 = v_2$, $T(u_1) \succ_e T(u_2)$;

(b) *or*

- $M_1 = \{q_1, q_2\}$, $p_1 = q_1$, $p_2 = q_2$, *and*
- $k = 2$, $u_1 = v_1$, $T(u_2) \succ_e T(v_2)$;

(c) *or* $M_1 = \{q_1\}$, $p_1 = q_1$, *and* $k = 1$, $T(u_1) \succ_e T(v_1)$;

(d) *or* $M_1 = \{q_1, p_1\}$, $p_1 < q_1$, *and* $k = 1$, $u_1 = v_1$;

then

$$\rho(G) \leq_{st} \pi(G).$$

In the above lemma, item (a) indicates that when the given two tasks are to be assigned to the given two processors, the task with larger subtree should be assigned to faster processor; items (b) and (c) indicate that to any given set of available processors, the tasks with the largest subtrees should be assigned; item (d) indicates that the fastest available processor should be used for any given task. Consequently, we can show (see the proof of Lemma 3.6) that

Lemma 5.4 *Let $\{a_n, M_n\}_{n=1}^{\infty}$ be a profile such that $|M_n| \leq 2$, $n \geq 1$, and let $G \in \mathcal{C}_{u.o.f}$ be a uniform out-forest. Let $\pi \in \Psi$ be a policy which follows the LT rule all the time except at the first decision epoch. Then,*

$$LT(G) \leq_{st} \pi(G).$$

The proof of Theorem 5.1 can finally be shown by a forward induction using Lemma 5.4 (see the proof of Theorem 3.1).

Consider now the optimal scheduling within the class of r-uniform out-forests. We will show that LT policy is optimal for any profile.

Theorem 5.5 *For any profile $\{a_n, M_n\}_{n=1}^{\infty}$ and any r-uniform out-forest $G \in \mathcal{C}_{r.o.f}$,*

$$\forall \pi \in \Psi : \qquad LT(G) \leq_{st} \pi(G). \tag{5.2}$$

Moreover, LT is the only policy that satisfies relation (5.2).

In order to prove the above theorem, we need the following lemma. The assertion of the lemma can be shown by mimicking the proof of Theorem 2 of Coffman and Liu [3]. The detailed proof is omitted.

Lemma 5.6 *Let $G \in \mathcal{C}_{r.o.f}$ be an arbitrary r-uniform forest, and u, v two roots of G such that $T(u) \succ_r T(v)$. Then for any policy $\pi \in \Psi$, there exists a policy $\pi_* \in \Psi$ such that for any profile $\{a_n, M_n\}_{n=1}^{\infty}$,*

$$\pi_*(G - \{u\}) \leq_{st} \pi(G - \{v\}).$$

Moreover, the inequality is strict if $T(u) \succ_r T(v)$ and $T(u) \not\prec_r T(v)$.

Proof of Theorem 5.5. We prove by induction on n that for any $G \in \mathcal{C}_{r.o.f}$ such that $|G| \leq n$, the relation

$$LT(G) \leq_{st} \pi(G) \tag{5.3}$$

holds for any profile.

If G is a singleton, i.e., $|G| = 1$, then the proof of (5.3) is simple. Indeed, the only possible difference between LT and π is that LT assigns the task to the fastest available processor, say processor q_1, whereas π assigns the task to another available processor, say processor q_2, with $\alpha_{q_1} \geq \alpha_{q_2}$. Trivially, the makespan of G under LT (whose distribution

is exponential with parameter α_{q_1}) is stochastically smaller than the makespan of G under π (whose distribution is exponential with parameter α_{q_2}). The relation (5.3) thus holds for all $|G| = 1$.

Assume that for some $n \geq 1$, relation (5.3) holds for all G such that $|G| \leq n$. Consider now the r-uniform out-forests G such that $|G| = n + 1$.

Fix the task graph $G \in \mathcal{C}_{r.o.f}$, the profile $\{a_n, M_n\}_{n=1}^{\infty}$, and the policy $\pi \in \Psi$. Denote by $\{d_n\}_{n=1}^{\infty}$ the sequence of decision epochs of π for finishing tasks in G, with $d_1 = 0$. Without loss of generality, we assume that d_2 corresponds to a task completion. The other cases can be analyzed similarly.

We construct three intermediate policies π_1, π_2, π_3 in the following way:

- Policies π_1, π_2 and π_3 follow the LT rule all the time except at the first decision epoch.
- At time $d_1 = 0$,
  - let $v_1, v_2, \cdots, v_k$ be the roots of G that are assigned by π to the processors $q_1, q_2, \cdots, q_k$, respectively;
  - policy π_1 assigns the roots $u_1, u_2, \cdots, u_k$ of G to the processors $q_1, q_2, \cdots, q_k$, respectively, such that $T(u_i) \succ_r T(v_i)$, $1 \leq i \leq k$;
  - policy π_2 assigns the roots $u_1, u_2, \cdots, u_k$ of G to the available processors $p_1, p_2, \cdots, p_k$, respectively, such that $p_i \leq q_i$, $1 \leq i \leq k$;
  - policy π_3 assigns root $u_{\phi(i)}$ of G to processor $p_{\psi(i)}$, $1 \leq i \leq k$, where ϕ and ψ are two permutations on $\{1, 2, \cdots, k\}$ such that

$$T(u_{\phi(1)}) \succ_r T(u_{\phi(2)}) \succ_r \cdots \succ_r T(u_{\phi(k)}), \qquad p_{\psi(1)} < p_{\psi(2)} < \cdots < p_{\psi(k)}.$$

We will show that

$$\pi_1(G) \leq_{st} \pi(G), \tag{5.4}$$

$$\pi_2(G) \leq_{st} \pi_1(G), \tag{5.5}$$

$$\pi_3(G) \leq_{st} \pi_2(G). \tag{5.6}$$

Under the assumption that d_2 corresponds to a task completion, we have that

$$\pi_1(G) = d_2 + \frac{1}{\lambda} \sum_{i=1}^{k} \alpha_{q_i} LT(G - \{u_i\}), \tag{5.7}$$

$$\pi(G) \;=\; d_2 + \frac{1}{\lambda}\sum_{i=1}^{k} a_{q_i}\pi(G-\{v_i\}). \tag{5.8}$$

where $\lambda = \sum_{i=1}^{k} a_{q_i}$. Owing to Lemma 5.6, for all $1 \le i \le k$, there is a policy π_*^i such that $\pi_*^i(G-\{u_i\}) \le_{st} \pi(G-\{v_i\})$. Using further the inductive assumption entails that

$$LT(G-\{u_i\}) \le_{st} \pi_*^i(G-\{u_i\}) \le_{st} \pi(G-\{v_i\}), \qquad 1 \le i \le k. \tag{5.9}$$

Inequalities (5.7), (5.8) and (5.9) readily imply (5.4).

Consider relation (5.5). We first show that for all $1 \le i \le k$,

$$LT(G-\{u_i\}) \le_{st} LT(G) \tag{5.10}$$

Define ρ^i to be the idling scheduling policy for task graph $G-\{u_i\}$ such that at any time a task $v \in G-\{u_i\}$ is assigned to a processor q if and only if $v \in G$ is assigned to processor q under LT for the task graph G. It is easy to see that policy ρ^i is idling as the processors running task u_i of G under LT are idled under policy ρ^i which assigns the tasks in $G-\{u_i\}$. Thus, one can construct a nonidling policy ρ_*^i of task graph $G-\{u_i\}$ by removing the idling in ρ^i. Using further the inductive assumption entails that

$$LT(G-\{u_i\}) \le_{st} \rho_*^i(G-\{u_i\}) \le_{st} \rho^i(G-\{u_i\}) =_{st} LT(G).$$

Relation (5.10) thus holds.

Now turn back to the proof of relation (5.5). For $0 \le i \le k$, we define γ_i to be a scheduling policy such that γ_i follows the LT rule all the time except at the first decision epoch. At time 0, policy γ_i assigns the roots $u_1, u_2, \cdots, u_k$ of G to the available processors $p_1, p_2, \cdots, p_i, q_{i+1}, q_{i+2}, \cdots, q_k$, respectively. Clearly, γ_0 is identical to π_1, and γ_k is identical to π_2.

In view of (5.10), we can show, as in Lemma 3.5, that for all $1 \le i \le k$,

$$\gamma_i(G) \le_{st} \gamma_{i-1}(G).$$

Therefore, relation (5.5) holds.

Finally, we consider relation (5.6). Owing to Lemma 5.6 and the inductive assumption, for all roots u, v of G, if $T(u) \succ_r T(v)$ then there exists a policy $\rho_{u,v}$ such that

$$LT(G-\{u\}) \le_{st} \rho_{u,v}(G-\{u\}) \le_{st} LT(G-\{v\}).$$

Therefore, the condition $T(u_{\sigma(1)}) \succ_r T(u_{\sigma(2)}) \succ_r \cdots \succ_r T(u_{\sigma(k)})$ implies that

$$LT(G - \{u_{\sigma(1)}\}) \leq_{st} LT(G - \{u_{\sigma(2)}\}) \leq_{st} \cdots \leq_{st} LT(G - \{u_{\sigma(k)}\}). \tag{5.11}$$

Relation (5.11) together with the condition $p_{\nu(1)} < p_{\nu(2)} < \cdots < p_{\nu(k)}$ readily yields relation (5.6) (cf. proof of Lemma 3.3).

In the construction of policies π_1, π_2, π_3, if $u_1, \cdots, u_k$ are the roots of G having the largest subtrees, and if processors $p_1, \cdots, p_k$ are the fastest available processors at time 0, then policy π_3 is the LT policy. Therefore, relations (5.4), (5.5) and (5.6) imply

$$LT(G) \leq_{st} \pi(G).$$

Hence, relation (5.3) holds for r-uniform out-forests G such that $|G| = n+1$. By induction, (5.3) holds for all r-uniform out-forests $G \in \mathcal{C}_{r.o.f.}$

Moreover, from Lemma 5.6 and relation (5.10), it is not difficult to see that if a policy π violates the LT rule, then, with positive probability $LT(G) < \pi(G)$. Therefore, LT is the only policy that satisfies relation (5.2). ∎

References

[1] J. Bruno, "On Scheduling Tasks with Exponential Service Times and In-Tree Precedence Constraints", *Acta Informatica*, **22** (1985), pp. 139–148.

[2] K. M. Chandy, P. F. Reynolds, "Scheduling Partially Ordered Tasks with Probabilistic Execution Times", *Operating System Review*, **9** (1975), pp. 169–177.

[3] E. G. Coffman, Z. Liu, "On the Optimal Stochastic Scheduling of Out-Forests", Rapport de Recherche INRIA No. 1156, 1990. To appear in *Operations Research*, January 1992.

[4] D. Dolev, M. K. Warmuth, "Scheduling Precedence Graphs of Bounded Height", *J. of Algorithms*, **5** (1984), pp. 48–59.

[5] D. Dolev, M. K. Warmuth, "Scheduling Flat Graphs", *SIAM J. on Comput.*, **14** (1985), pp. 638–657.

[6] D. Dolev, M. K. Warmuth, "Profile Scheduling of Opposing Forests and Level Orders", *SIAM J. Alg. Disc. Meth.*, **6** (1985), pp. 665–687.

[7] E. Frostig, "A Stochastic Scheduling Problem with Intree Precedence Constraints", *Operations Research*, **36** (1988), pp. 937–943.

[8] M. R. Garey, D. S. Johnson, R. E. Tarjan, M. Yanakakis, "Scheduling Opposite Forests", *SIAM J. Alg. Disc. Meth.*, **4** (1983), pp. 72–93.

[9] C. H. Papadimitriou, J. N. Tsitsiklis, "On Stochastic Scheduling with In-Tree Precedence Constraints", *SIAM J. Comput.*, **16** (1987), pp. 1–6.

[10] C. H. Papadimitriou, M. Yannakakis, "Scheduling Interval-Ordered Tasks", *SIAM J. Comput.*, **8** (1979), pp. 405–409.

[11] M. Pinedo, G. Weiss, "Scheduling Jobs with Exponentially Distributed Processing Times and Intree Precedence Constraints on Two Parallel Machines", *Operations Research*, **33** (1985), pp. 1381–1388.

[12] V. Strassen, "The Existence of Probability Measures with Given Marginals", *Ann. Math. Sta.*, **36** (1965), pp. 423–439.

[13] J. D. Ullman, "NP-Complete Scheduling Problems" *J. Comput. System Sci.*, **10** (1975), pp. 384–393.

ISSN 0249 - 6399