Near Delay-Optimal Scheduling of Batch Jobs in Multi-Server Systems

Yin Sun, C. Emre Koksal, and Ness B. Shroff

February 2, 2017

Abstract

We study a class of scheduling problems, where each job is divided into a batch of unit-size tasks and these tasks can be executed in parallel on multiple servers with New-Better-than-Used (NBU) service time distributions. While many delay optimality results are available for single-server queueing systems, generalizing these results to the multi-server case has been challenging. This motivated us to investigate near delay-optimal scheduling of batch jobs in multi-server queueing systems. We consider three low-complexity scheduling policies: the Fewest Unassigned Tasks first (FUT) policy, the Earliest Due Date first (EDD) policy, and the First-Come, First-Served (FCFS) policy. We prove that for arbitrary number, batch sizes, arrival times, and due times of the jobs, these scheduling policies are near delay-optimal in stochastic ordering for minimizing three classes of delay metrics among all causal and non-preemptive policies. In particular, the FUT policy is within a constant additive delay gap from the optimum for minimizing the mean average delay, and the FCFS policy within twice of the optimum for minimizing the mean maximum delay and the mean p-norm of delay. The key proof tools are several novel sample-path orderings, which can be used to compare the sample-path delay of different policies in a near-optimal sense.

Index Terms

Scheduling, multi-server queueing systems, near delay optimality, New-Better-than-Used distribution, sample-path methods, stochastic ordering.

Yin Sun and C. Emre Koksal are with the Department of Electrical and Computer Engineering, the Ohio State University, Columbus, OH. Email: sunyin02@gmail.com, koksal.2@osu.edu.

Ness B. Shroff is with the Departments of Electrical and Computer Engineering and Computer Science and Engineering, the Ohio State University, Columbus, OH. Email: shroff.11@osu.edu.

I. INTRODUCTION

Achieving low service latency is imperative in cloud computing and data center systems. As the user population and data volume scale up, the running times of service jobs grow quickly. In practice, long-running jobs are broken into a batch of short tasks which can be executed separately by a number of parallel servers [1]. We study the scheduling of such batch jobs in multi-server queueing systems. The jobs arrive over time according to a general arrival process, where the number, batch sizes, arrival times, and due times of the jobs are *arbitrarily* given. A scheduler determines how to assign the tasks of incoming jobs to the servers, based on the casual information (the history and current information) of the system. We assume that each task is of one unit of work and can be flexibly assigned to any server. The service times of the tasks follow New-Better-than-Used (NBU) distributions, and are *independent* across the servers and *i.i.d.* across the tasks assigned to the same server. Our goal is to seek for low-complexity scheduling policies that optimize the delay performance of the jobs.

Many delay-optimal scheduling results have been established in single-server queueing systems. In the deterministic scheduling models, the preemptive Shortest Remaining Processing Time first (SRPT) policy, the preemptive Earliest Due Date first (EDD) policy, and the First-Come, First-Served (FCFS) policy were proven to minimize the average delay [2], [3], maximum lateness [4], and maximum delay [5], respectively. In addition, index policies are shown to be delay-optimal in many stochastic scheduling problems, e.g., [6]–[10].

Unfortunately, generalizing these delay-optimal results to the parallel-server case is quite challenging. In particular, minimizing the average delay in deterministic scheduling problems with more than one servers is NP-hard [11]. Similarly, delay-optimal stochastic scheduling in multi-class multi-server queueing systems is deemed to be notoriously difficult [12], [13]. Prior attempts on solving the delay-optimal stochastic scheduling problem have met with little success, except in some limiting regions such as large system limits, e.g., [14], and heavy traffic limits, e.g., [15], [16]. However, these results may not apply either outside of these limiting regions, or for a non-stationary job arrival process under which the steady state distribution of the system does not exist.

A. Our Contributions

Because of the difficulty in establishing delay optimality in the considered queueing systems, we investigate near delay-optimal scheduling in this paper. We prove that for arbitrarily given job parameters (including the number, batch sizes, arrival times, and due times of the jobs), the Fewest Unassigned Tasks first (FUT) policy, the EDD policy, and the FCFS policy are near delay-optimal in stochastic ordering for minimizing average delay, maximum lateness, and maximum delay, respectively, among all causal and

non-preemptive[1] policies. If the job parameters satisfy some additional conditions, we can further show that these policies are near delay-optimal in stochastic ordering for minimizing three general classes of delay metrics. In particular, we show that the FUT policy is within a constant additive delay gap from the optimum for minimizing the mean average delay (Theorem 2)[2]; and the FCFS policy within twice of the optimum for minimizing the mean maximum delay and mean p-norm of delay (Theorems 5 and 6).

We develop a unified sample-path method to establish these results, where the key proof tools are several novel sample-path orderings for comparing the delay performance of different policies in a near-optimal sense. These sample-path orderings are very general because they do not need to specify the queueing system model, and hence can be potentially used for establishing near delay optimality results in other queueing systems.

B. Organization of the Paper

We describe the system model and problem formulation in Section II, together with the notations that we will use throughout the paper. Some near delay optimality results are introduced in Section III, which are proven in Section IV. Finally, some conclusion discussions are presented in Section V.

II. MODEL AND FORMULATION

A. Notations and Definitions

We will use lower case letters such as x and $\boldsymbol{x}$, respectively, to represent deterministic scalars and vectors. In the vector case, a subscript will index the components of a vector, such as x_i. We use $x_{[i]}$ and $x_{(i)}$, respectively, to denote the i-th largest and the i-th smallest components of $\boldsymbol{x}$. For any n-dimensional vector $\boldsymbol{x}$, let $\boldsymbol{x}_{\uparrow} = (x_{(1)}, \ldots, x_{(n)})$ denote the increasing rearrangement of $\boldsymbol{x}$. Let $\boldsymbol{0}$ denote the vector with all 0 components.

Random variables and vectors will be denoted by upper case letters such as X and $\mathbf{X}$, respectively, with the subscripts and superscripts following the same conventions as in the deterministic case. Throughout the paper, "increasing/decreasing" and "convex/concave" are used in the non-strict sense. LHS and RHS denote, respectively, "left-hand side" and "right-hand side".

For any n-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the elementwise vector ordering $x_i \leq y_i$, $i = 1, \ldots, n$, is denoted by $\boldsymbol{x} \leq \boldsymbol{y}$. Further, $\boldsymbol{x}$ is said to be *majorized* by $\boldsymbol{y}$, denoted by $\boldsymbol{x} \prec \boldsymbol{y}$, if (i) $\sum_{i=1}^{j} x_{[i]} \leq \sum_{i=1}^{j} y_{[i]}$,

---

[1]We consider task-level non-preemptive policies: Processing of a task cannot be interrupted until the task is completed. However, after completing a task from one job, a server can switch to process another task from any job.

[2]As we will see in Section III, the FUT policy is a nice approximation of the Shortest Remaining Processing Time (SRPT) first policy [2], [3].
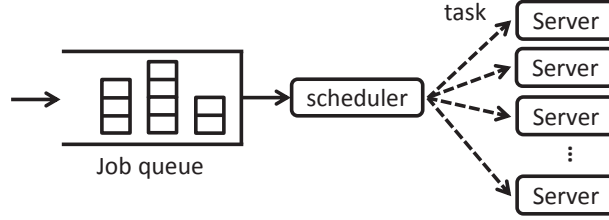
Fig. 1: A centralized queueing system, where a scheduler assigns the tasks of arrived jobs to the servers.

$j = 1, \ldots, n - 1$ and (ii) $\sum_{i=1}^{n} x_{[i]} = \sum_{i=1}^{n} y_{[i]}$ [18]. In addition, $\boldsymbol{x}$ is said to be *weakly majorized by $\boldsymbol{y}$ from below*, denoted by $\boldsymbol{x} \prec_{\mathrm{w}} \boldsymbol{y}$, if $\sum_{i=1}^{j} x_{[i]} \leq \sum_{i=1}^{j} y_{[i]}$, $j = 1, \ldots, n$; $\boldsymbol{x}$ is said to be *weakly majorized by $\boldsymbol{y}$ from above*, denoted by $\boldsymbol{x} \prec^{\mathrm{w}} \boldsymbol{y}$, if $\sum_{i=1}^{j} x_{(i)} \geq \sum_{i=1}^{j} y_{(i)}$, $j = 1, \ldots, n$ [18]. A function that preserves the majorization order is called a Schur convex function. Specifically, $f : \mathbb{R}^n \to \mathbb{R}$ is termed *Schur convex* if $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ for all $\boldsymbol{x} \prec \boldsymbol{y}$ [18]. A function $f : \mathbb{R}^n \to \mathbb{R}$ is termed *symmetric* if $f(\boldsymbol{x}) = f(\boldsymbol{x}_\uparrow)$ for all $\boldsymbol{x}$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is termed *sub-additive* if $f(\boldsymbol{x}+\boldsymbol{y}) \leq f(\boldsymbol{x})+f(\boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{y}$. The composition of functions ϕ and f is denoted by $\phi \circ f(\boldsymbol{x}) = \phi(f(\boldsymbol{x}))$. Define $x \wedge y = \min\{x, y\}$.

Let $\mathcal{A}$ and $\mathcal{S}$ denote sets and events, with $|\mathcal{S}|$ denoting the cardinality of $\mathcal{S}$. For all random variable X and event $\mathcal{A}$, let $[X|\mathcal{A}]$ denote a random variable whose distribution is identical to the conditional distribution of X for given $\mathcal{A}$. A random variable X is said to be stochastically smaller than another random variable Y, denoted by $X \leq_{\mathrm{st}} Y$, if $\Pr(X > x) \leq \Pr(Y > x)$ for all $x \in \mathbb{R}$. A set $\mathcal{U} \subseteq \mathbb{R}^n$ is called *upper*, if $\boldsymbol{y} \in \mathcal{U}$ whenever $\boldsymbol{y} \geq \boldsymbol{x}$ and $\boldsymbol{x} \in \mathcal{U}$. A random vector $\boldsymbol{X}$ is said to be *stochastically smaller* than another random vector $\boldsymbol{Y}$, denoted by $\boldsymbol{X} \leq_{\mathrm{st}} \boldsymbol{Y}$, if $\Pr(\boldsymbol{X} \in \mathcal{U}) \leq \Pr(\boldsymbol{Y} \in \mathcal{U})$ for all upper sets $\mathcal{U} \subseteq \mathbb{R}^n$. If $\boldsymbol{X} \leq_{\mathrm{st}} \boldsymbol{Y}$ and $\boldsymbol{X} \geq_{\mathrm{st}} \boldsymbol{Y}$, then $\boldsymbol{X}$ and $\boldsymbol{Y}$ follow the same distribution, denoted by $\boldsymbol{X} =_{\mathrm{st}} \boldsymbol{Y}$. We remark that $\boldsymbol{X} \leq_{\mathrm{st}} \boldsymbol{Y}$ if, and only if,

$$\mathbb{E}[\phi(\boldsymbol{X})] \leq \mathbb{E}[\phi(\boldsymbol{Y})] \tag{1}$$

holds for all increasing $\phi : \mathbb{R}^n \to \mathbb{R}$ for which the expectations in (1) exist.

B. Queueing System Model

Consider a queueing system with m parallel servers, as shown in Fig. 1, which starts to operate at time $t = 0$. A sequence of n jobs arrive at time instants $a_1, \ldots, a_n$ and are stored in a job queue, where n is an arbitrary positive integer, no matter finite or infinite, and $0 = a_1 \leq a_2 \leq \cdots \leq a_n$. The i-th arrived job, called job i, brings with it a batch of k_i tasks. Each task is of one unit of work, and can be flexibly assigned to any server. Job i is completed when all k_i tasks of job i are completed. The maximum job

size is[3]

$$k_{\max} = \max_{i=1,\ldots,n} k_i. \tag{2}$$

A scheduler assigns the tasks of arrived jobs to the servers over time. The service times of the tasks are assumed to be *independent* across the servers and *i.i.d.* across the tasks assigned to the same server. Let X_l represent the random service time of server l. The service rate of server l is $\mu_l = 1/\mathbb{E}[X_l]$, which may vary across the servers. Denote $\boldsymbol{X} = (X_1, \ldots, X_m)$. We consider the classes of NBU task service time distributions.

Definition 1. Consider a non-negative random variable X with complementary cumulative distribution function (CDF) $\bar{F}(x) = \Pr[X > x]$. Then, X is **New-Better-than-Used (NBU)** if for all $t, \tau \geq 0$

$$\bar{F}(\tau + t) \leq \bar{F}(\tau)\bar{F}(t). \tag{3}$$

C. Scheduling Policies

A scheduling policy, denoted by π, determines the task assignments in the system. We consider the class of **causal** policies, in which scheduling decisions are made based on the history and current state of the system; the realization of task service time is unknown until the task is completed (unless the service time is deterministic). In practice, service preemption is costly and may leads to complexity and reliability issues [19], [20]. Motivated by this, we assume that **preemption is not allowed**. Hence, if a server starts to process a task, it must complete this task before switching to process another task. We use Π to denote the set of causal and non-preemptive policies. Let us define several sub-classes of policies within Π:

A policy is said to be **non-anticipative**, if the scheduling decisions are made without using any information about the future job arrivals; and is said to be **anticipative**, if it has access to the parameters (a_i, k_i, d_i) of future arriving jobs. For periodic and pre-planned services, future job arrivals can be predicted in advance. To cover these scenarios, we abuse the definition of causal policies a bit and include anticipative policies into the policy space Π. However, the policies that we propose in this paper are all non-anticipative.

A policy is said to be **work-conserving**, if no server is idle when there are tasks waiting to be processed.

---

[3]If $n \to \infty$, then the max operator in (2) is replaced by sup.

The goal of this paper is to design low-complexity non-anticipative scheduling policies that are near delay-optimal among all policies in Π, even compared to the anticipative policies with knowledge about future arriving jobs.

D. Delay Metrics

Each job i has a **due time** $d_i \in [0, \infty)$, also called due date, which is the time that job i is promised to be completed [21]. Completion of a job after its due time is allowed, but then a penalty is incurred. Hence, the due time can be considered as a soft deadline.

For each job i, C_i is the job completion time, $D_i = C_i - a_i$ is the delay, $L_i = C_i - d_i$ is the lateness after the due time d_i, and $T_i = \max[C_i - d_i, 0]$ is the tardiness (or positive lateness). Define the vectors $\boldsymbol{a} = (a_1, \ldots, a_n)$, $\boldsymbol{d} = (d_1, \ldots, d_n)$, $\boldsymbol{C} = (C_1, \ldots, C_n)$, $\boldsymbol{D} = (D_1, \ldots, D_n)$, $\boldsymbol{L} = (L_1, \ldots, L_n)$, and $\boldsymbol{C}_\uparrow = (C_{(1)}, \ldots, C_{(n)})$. Let $\boldsymbol{c} = (c_1, \ldots, c_n)$ and $\boldsymbol{c}_\uparrow = (c_{(1)}, \ldots, c_{(n)})$, respectively, denote the realizations of $\boldsymbol{C}$ and $\boldsymbol{C}_\uparrow$. All these quantities are functions of the scheduling policy π.

Several important delay metrics are introduced in the following: For any policy π, the **average delay** $D_{\mathrm{avg}} : \mathbb{R}^n \to \mathbb{R}$ is defined by[4]

$$D_{\mathrm{avg}}(\boldsymbol{C}(\pi)) = \frac{1}{n} \sum_{i=1}^{n} [C_i(\pi) - a_i]. \tag{4}$$

The **maximum lateness** $L_{\max} : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$L_{\max}(\boldsymbol{C}(\pi)) = \max_{i=1,2,\ldots,n} L_i(\pi) = \max_{i=1,2,\ldots,n} [C_i(\pi) - d_i], \tag{5}$$

if $d_i = a_i$, $L_{\max}$ reduces to the **maximum delay** $D_{\max} : \mathbb{R}^n \to \mathbb{R}$, i.e.,

$$D_{\max}(\boldsymbol{C}(\pi)) = \max_{i=1,2,\ldots,n} D_i(\pi) = \max_{i=1,2,\ldots,n} [C_i(\pi) - a_i]. \tag{6}$$

The p-**norm of delay** $D_p : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$D_p(\boldsymbol{C}(\pi)) = ||\boldsymbol{D}(\pi)||_p = \left[\sum_{i=1}^{n} D_i(\pi)^p \right]^{1/p}, \quad p \geq 1. \tag{7}$$

In general, a **delay metric** can be expressed as a function $f(\boldsymbol{C}(\pi))$ of the job completion time vector

---

[4]If $n \to \infty$, then a $\limsup$ operator is enforced on the RHS of (4), and the $\max$ operator in (5) and (6) is replaced by $\sup$.

$C(\pi)$, where $f : \mathbb{R}^n \to \mathbb{R}$ is increasing. In this paper, we consider three classes of delay metric functions:

$$\mathcal{D}_{\text{sym}} = \{f : f \text{ is symmetric and increasing}\},$$

$$\mathcal{D}_{\text{Sch-1}} = \{f : f(\boldsymbol{x} + \boldsymbol{d}) \text{ is Schur convex and increasing in } \boldsymbol{x}\},$$

$$\mathcal{D}_{\text{Sch-2}} = \{f : f(\boldsymbol{x} + \boldsymbol{a}) \text{ is Schur convex and increasing in } \boldsymbol{x}\}.$$

Consider two alternative expressions of the delay metric function f:

$$f_1(\boldsymbol{L}(\pi)) = f(\boldsymbol{L}(\pi) + \boldsymbol{d}) = f(\boldsymbol{C}(\pi)), \tag{8}$$

$$f_2(\boldsymbol{D}(\pi)) = f(\boldsymbol{D}(\pi) + \boldsymbol{a}) = f(\boldsymbol{C}(\pi)). \tag{9}$$

Then, $f_1(\boldsymbol{L}(\pi))$ is Schur convex in the lateness vector $\boldsymbol{L}(\pi)$ for each $f \in \mathcal{D}_{\text{Sch-1}}$, and $f_2(\boldsymbol{D}(\pi))$ is Schur convex in the delay vector $\boldsymbol{D}(\pi)$ for each $f \in \mathcal{D}_{\text{Sch-2}}$. Furthermore, every convex and symmetric function is Schur convex. Using these properties, it is easy to show that

$$D_{\text{avg}} \in \mathcal{D}_{\text{sym}} \cap \mathcal{D}_{\text{Sch-1}} \cap \mathcal{D}_{\text{Sch-2}},$$

$$L_{\text{max}} \in \mathcal{D}_{\text{Sch-1}}, D_{\text{max}} \in \mathcal{D}_{\text{Sch-2}}, D_p \in \mathcal{D}_{\text{Sch-2}}.$$

E. Near Delay Optimality

Define $\mathcal{I} = \{n, (a_i, k_i, d_i)_{i=1}^n\}$ as the parameters of the jobs, which include the number, batch sizes, arrival times, and due times of the jobs. The job parameters $\mathcal{I}$ and random task service times are determined by two external processes, which are *mutually independent* and do not change according to the scheduling policy. For delay metric function f and policy space Π, a policy $P \in \Pi$ is said to be **delay-optimal in stochastic ordering**, if for all $\pi \in \Pi$ and $\mathcal{I}$

$$[f(\boldsymbol{C}(P))|P, \mathcal{I}] \leq_{\text{st}} [f(\boldsymbol{C}(\pi))|\pi, \mathcal{I}]. \tag{10}$$

or equivalently, if for all $\mathcal{I}$

$$\mathbb{E}[\phi \circ f(\boldsymbol{C}(P))|P, \mathcal{I}] = \min_{\pi \in \Pi} \mathbb{E}[\phi \circ f(\boldsymbol{C}(\pi))|\pi, \mathcal{I}] \tag{11}$$

holds for all increasing function $\phi : \mathbb{R} \to \mathbb{R}$ for which the conditional expectations in (11) exist. The equivalence between (10) and (11) follows from (1). For notational simplicity, we will omit the mention of the condition that policy π or policy P is adopted in the system in the rest of the paper. Note that this definition of delay optimality is quite strong because the optimal policy P needs to satisfy (10) and (11) for *arbitrarily given* job parameters $\mathcal{I}$.
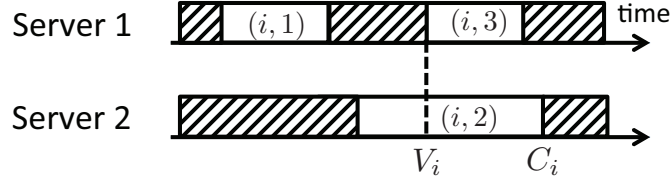
Fig. 2: An illustration of V_i and C_i. There are 2 servers, and job i has 3 tasks denoted by $(i, 1)$, $(i, 2)$, $(i, 3)$. Tasks $(i, 1)$ and $(i, 3)$ are assigned to Server 1, and Task $(i, 2)$ is assigned to Server 2. By time V_i, all tasks of job i have started service. By time C_i, all tasks of job i have completed service. Therefore, $V_i \le C_i$.

In multi-class multi-server queueing systems, delay optimality is extremely difficult to achieve, even with respect to some definitions of delay optimality that are weaker than (10). In deterministic scheduling problems, it was shown that minimizing the average delay in multi-server queueing systems is NP-hard [11]. Similarly, delay-optimal stochastic scheduling in multi-server queueing systems with multiple job classes is deemed to be notoriously difficult [12], [13], [15]. This motivated us to study whether there exist policies that can come close to delay optimality. We will show that this is possible in the following sense:

Define V_i as the earliest time that all tasks of job i have entered the servers. In other words, all tasks of job i are either completed or under service at time V_i. One illustration of V_i is provided in Fig. 2, from which it is easy to see

$$V_i \le C_i. \tag{12}$$

Denote $\boldsymbol{V} = (V_1, \ldots, V_n)$, $\boldsymbol{V}_\uparrow = (V_{(1)}, \ldots, V_{(n)})$. Let $\boldsymbol{v} = (v_1, \ldots, v_n)$ and $\boldsymbol{v}_\uparrow = (v_{(1)}, \ldots, v_{(n)})$ be the realization of $\boldsymbol{V}$ and $\boldsymbol{V}_\uparrow$, respectively. All these quantities are functions of the scheduling policy π. A policy $P \in \Pi$ is said to be **near delay-optimal in stochastic ordering**, if for all $\pi \in \Pi$ and $\mathcal{I}$

$$[f(\boldsymbol{V}(P))|\mathcal{I}] \le_{\mathrm{st}} [f(\boldsymbol{C}(\pi))|\mathcal{I}], \tag{13}$$

or equivalently, if for all $\mathcal{I}$

$$\mathbb{E}[\phi \circ f(\boldsymbol{V}(P))|\mathcal{I}] \le \min_{\pi \in \Pi} \mathbb{E}[\phi \circ f(\boldsymbol{C}(\pi))|\mathcal{I}] \le \mathbb{E}[\phi \circ f(\boldsymbol{C}(P))|\mathcal{I}] \tag{14}$$

holds for all increasing function ϕ for which the conditional expectations in (14) exist. There exist many ways to approximate (10) and (11), and obtain various forms of near delay optimality. We find that the form in (13) and (14) is convenient, because it is analytically provable and leads to tight sub-optimal delay gap, as we will see in the subsequent sections.

Algorithm 1: Fewest Unassigned Tasks first (FUT).

---

1 $Q := \emptyset$;   // Q is the set of jobs in the queue
2 **while** *the system is ON* **do**
3   **if** *job i arrives* **then**
4        $\xi_i := k_i$;   // job i has ξ_i remaining tasks
5        $\gamma_i := k_i$;   // job i has γ_i unassigned tasks
6        $Q := Q \cup \{i\}$;
7   **end**
8   **while** $\sum_{i \in Q} \gamma_i > 0$ *and there are idle servers* **do**
9   Pick any idle server l;
10       $j := \arg\min\{\gamma_i : i \in Q, \gamma_i > 0\}$;
11   Allocate a task of job j on server l;
12       $\gamma_j := \gamma_j - 1$;
13   **end**
14   **if** *a task of job i is completed* **then**
15       $\xi_i := \xi_i - 1$;
16   **if** $\xi_i = 0$ **then** $Q := Q/\{i\}$;
17   **end**
18 **end**

III. MAIN RESULTS

In this section, we provide some near delay optimality results in the form of (13) and (14). These results will be proven in Section IV by using a unified sample-path method.

A. Average Delay D_{avg} and the Delay Metrics in $\mathcal{D}_{sym}$

Let us first consider the following **Fewest Unassigned Tasks first (FUT)** policy, which is also described in Algorithm 1: Each idle server is allocated to process a task from the job with the earliest due time among all jobs with unassigned tasks. The delay performance of policy FUT is characterized in the following theorem.

Theorem 1. *If the task service times are NBU, independent across the servers, and i.i.d. across the tasks assigned to the same server, then for all $f \in \mathcal{D}_{\mathrm{sym}}$, $\pi \in \Pi$, and $\mathcal{I}$*

$$[f(\boldsymbol{V}(\mathrm{FUT}))|\mathcal{I}] \leq_{\mathrm{st}} [f(\boldsymbol{C}(\pi))|\mathcal{I}]. \tag{15}$$

If there are multiple idle servers, the choices of the idle server in Step 9 of Algorithm 1 are non-unique. The result of Theorem 1 holds for any server selection method in Step 9. In practice, one can pick the idle server with the fastest service rate.

Let us characterize the sub-optimality delay gap of policy FUT. For mean average delay, i.e., $f(\cdot) = D_{\mathrm{avg}}(\cdot)$, it follows from (15) that

$$\mathbb{E}[D_{\mathrm{avg}}(\boldsymbol{V}(\mathrm{FUT}))|\mathcal{I}] \leq \min_{\pi \in \Pi} \mathbb{E}\left[D_{\mathrm{avg}}(\boldsymbol{C}(\pi))\,|\,\mathcal{I}\right] \leq \mathbb{E}[D_{\mathrm{avg}}(\boldsymbol{C}(\mathrm{FUT}))|\mathcal{I}]. \tag{16}$$

The delay gap between the LHS and RHS of (16) is given by

$$\mathbb{E}[D_{\mathrm{avg}}(\boldsymbol{C}(\mathrm{FUT})) - D_{\mathrm{avg}}(\boldsymbol{V}(\mathrm{FUT}))|\mathcal{I}]$$
$$= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left(C_i(\mathrm{FUT}) - V_i(\mathrm{FUT})\right)\Big|\mathcal{I}\right]. \tag{17}$$

Recall that m is the number of servers, and k_i is the number of tasks in job i. At time $V_i(\mathrm{FUT})$, all tasks of job i have started service: If $k_i > m$, then job i has at most m incomplete tasks that are under service at time $V_i(\mathrm{FUT})$; if $k_i \leq m$, then job i has at most k_i incomplete tasks that are under service at time $V_i(\mathrm{FUT})$. Therefore, in policy FUT, at most $k_i \wedge m = \min\{k_i, m\}$ tasks of job i are under service during the time interval $[V_i(\mathrm{FUT}), C_i(\mathrm{FUT})]$. Using this and the property of NBU distributions, we can obtain

Theorem 2. *Let* $\mathbb{E}[X_l] = 1/\mu_l$, *and without loss of generality* $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_m$. *If the task service times are NBU, independent across the servers, i.i.d. across the tasks assigned to the same server, then for all* $\mathcal{I}$

$$\mathbb{E}[D_{\mathrm{avg}}(\boldsymbol{C}(\mathrm{FUT}))|\mathcal{I}] - \min_{\pi \in \Pi}\mathbb{E}\left[D_{\mathrm{avg}}(\boldsymbol{C}(\pi))|\mathcal{I}\right] \leq \frac{1}{n}\sum_{i=1}^{n}\sum_{l=1}^{k_i \wedge m}\frac{1}{\sum_{j=1}^{l}\mu_j} \tag{18}$$
$$\leq \frac{\ln(k_{\max} \wedge m) + 1}{\mu_1}, \tag{19}$$

where $k_{\max}$ *is the maximum job size in* (2)*, and* $x \wedge y = \min\{x, y\}$.

If $\mathbb{E}[X_l] \leq 1/\mu$ for $l = 1, \ldots, m$, then the sub-optimality delay gap of policy FUT is of the order $O(\ln(k_{\max} \wedge m))/\mu$. As the number of servers m increases, this sub-optimality delay gap is upper bounded by $[\ln(k_{\max}) + 1]/\mu$ which is independent of m.

Theorem 1 and Theorem 2 tell us that for arbitrary number, batch sizes, arrival times, and due times of the jobs, the FUT policy is near delay-optimal for minimizing the average delay D_{avg} in the mean.

The FUT policy is a nice approximation of the Shortest Remaining Processing Time (SRPT) first policy [2], [3]: The FUT policy utilizes the number of unassigned tasks of a job to approximate the remaining processing time of this job, and is within a small additive sub-optimality gap from the optimum for minimizing the mean average delay in the scheduling problem that we consider.

Due to the difficulty of delay minimization in multi-class multi-server queueing systems, attempts on finding a tight additive sub-optimal delay gap in these systems has met with little success. In [12], [13], closed-form upper bounds on the sub-optimality gaps of the Smith's rule and the Gittin's index rule were established for the cases that all jobs arrive at time zero. In Corollary 2 of [15], the authors studied the optimal control in multi-server queueing systems with stationary arrivals of multiple classes of jobs, and used the achievable region method to obtain an additive sub-optimality delay gap, which is of the order $O(mk_{\max})$. The model and methodology in these studies are significantly different from those in this paper.

B. Maximum Lateness $L_{\max}$ and the Delay Metrics in $\mathcal{D}_{sym} \cup \mathcal{D}_{Sch\text{-}1}$

Next, we investigate the maximum lateness $L_{\max}$ and the delay metrics in $\mathcal{D}_{\mathrm{sym}} \cup \mathcal{D}_{\mathrm{Sch\text{-}1}}$. We consider a **Earliest Due Date first (EDD)** policy: Each idle server is allocated to process a task from the job with the earliest due time among all jobs with unassigned tasks. Policy EDD can be obtained from Algorithm 1 by revising step 10 as $j := \arg\min\{d_i : i \in Q, \gamma_i > 0\}$. The delay performance of policy EDD is characterized in the following theorem.

Theorem 3. *If the task service times are NBU, independent across the servers, and i.i.d. across the tasks assigned to the same server, then for all $\pi \in \Pi$, and $\mathcal{I}$*

$$[L_{\max}(\boldsymbol{V}(\mathrm{EDD}))|\mathcal{I}] \leq_{\mathrm{st}} [L_{\max}(\boldsymbol{C}(\pi)) \,|\, \mathcal{I}]. \tag{20}$$

If the job sizes satisfy certain conditions, policy EDD is identical with policy FUT. In this case, policy EDD is near delay-optimal in stochastic ordering for minimizing both the average delay $D_{\mathrm{avg}}(\cdot)$ and the maximum lateness $L_{\max}(\cdot)$. Interestingly, in this case policy EDD is near delay-optimal in stochastic ordering for minimizing all delay metrics in $\mathcal{D}_{\mathrm{sym}} \cup \mathcal{D}_{\mathrm{Sch\text{-}1}}$, as stated in the following theorem:

Theorem 4. *If $k_1 = \ldots = k_n = 1$ (or $d_1 \leq d_2 \leq \ldots \leq d_n$ and $k_1 \leq k_2 \leq \ldots \leq k_n$) and $L_{\max}$ is replaced by any $f \in \mathcal{D}_{\mathrm{sym}} \cup \mathcal{D}_{\mathrm{Sch\text{-}1}}$, Theorem 3 still holds.*

C. Maximum Delay $D_{\max}$ and the Delay Metrics in $\mathcal{D}_{sym} \cup \mathcal{D}_{Sch\text{-}2}$

We now study maximum delay $D_{\max}$ and the delay metrics in $\mathcal{D}_{\mathrm{sym}} \cup \mathcal{D}_{\mathrm{Sch\text{-}2}}$. We consider a **First-Come, First-Served (FCFS)** policy: Each idle server is allocated to process a task from the job with the earliest arrival time among all jobs with unassigned tasks. Policy FCFS can be obtained from Algorithm 1 by revising step 10 as $j := \arg\min\{a_i : i \in Q, \gamma_i > 0\}$. The delay performance of FCFS is characterized as follows:

Corollary 1. *If the task service times are NBU, independent across the servers, and i.i.d. across the tasks assigned to the same server, then for all $\pi \in \Pi$, and $\mathcal{I}$*

$$[D_{\max}(\boldsymbol{V}(\text{FCFS}))|\mathcal{I}] \leq_{\text{st}} [D_{\max}(\boldsymbol{C}(\pi))|\mathcal{I}]. \tag{21}$$

In addition, if the job sizes satisfy certain conditions, policy FCFS is identical with policy FUT. Hence, policy FCFS is near delay-optimal in stochastic ordering for minimizing both the average delay $D_{\text{avg}}(\cdot)$ and the maximum delay $D_{\max}(\cdot)$. In this case, we can further show that policy FCFS is near delay-optimal in distribution for minimizing all delay metrics in $\mathcal{D}_{\text{sym}} \cup \mathcal{D}_{\text{Sch-2}}$, as stated in the following:

Corollary 2. *If $k_1 \leq k_2 \leq \ldots \leq k_n$ and $D_{\max}$ is replaced by any $f \in \mathcal{D}_{\text{sym}} \cup \mathcal{D}_{\text{Sch-2}}$, Theorem 1 still holds.*

The proofs of Corollaries 1 and 2 are omitted, because they follow directly from Theorems 3 and 4 by setting $d_i = a_i$ for all job i.

In the sequel, we show that policy FCFS is within twice of the optimum for minimizing the mean maximum delay $\mathbb{E}[D_{\max}(\boldsymbol{C}(\pi))|\mathcal{I}]$ and mean p-norm of delay $\mathbb{E}[\|\boldsymbol{D}(\pi)\|_p|\mathcal{I}]$ for $p \geq 1$.[5]

Lemma 1. *For arbitrarily given job parameters $\mathcal{I}$, if (i) the task service times are NBU and i.i.d. across the tasks and servers, (ii) $f(\boldsymbol{x} + \boldsymbol{a})$ is sub-additive and increasing in $\boldsymbol{x}$, and (iii) two policies $P, \pi \in \Pi$ satisfy*

$$[f(\boldsymbol{V}(P))|\mathcal{I}] \leq_{\text{st}} [f(\boldsymbol{C}(\pi))|\mathcal{I}], \tag{22}$$

then

$$[f(\boldsymbol{C}(P))|\mathcal{I}] \leq_{\text{st}} [2f(\boldsymbol{C}(\pi))|\mathcal{I}]. \tag{23}$$

Proof. See Appendix A. □

By applying Lemma 1 in Corollaries 1 and 2, it is easy to obtain

---

[5]Because of the maximum operator over all jobs, $\mathbb{E}[D_{\max}(\boldsymbol{C}(\pi))|\mathcal{I}]$ will likely grow to infinite as the number of jobs n increases. Hence, it is difficult to obtain an additive sub-optimality gap for minimizing $\mathbb{E}[D_{\max}(\boldsymbol{C}(\pi))|\mathcal{I}]$ that remains constant for all n.

Theorem 5. *If the task service times are NBU and i.i.d. across the tasks and servers, then for all $\mathcal{I}$*

$$[D_{\max}(\boldsymbol{C}(\text{FCFS}))|\mathcal{I}] \leq_{\text{st}} [2D_{\max}(\boldsymbol{C}(\pi))|\mathcal{I}], \; \forall \; \pi \in \Pi.$$

$$\mathbb{E}[D_{\max}(\boldsymbol{C}(\text{FCFS}))|\mathcal{I}] \leq \min_{\pi \in \Pi} 2\mathbb{E}\left[D_{\max}(\boldsymbol{C}(\pi))|\mathcal{I}\right].$$

Theorem 6. *If (i) $k_1 \leq k_2 \leq \ldots \leq k_n$, (ii) the task service times are NBU and i.i.d. across the tasks and servers, (iii) $f(\boldsymbol{x} + \boldsymbol{a})$ is sub-additive and increasing in $\boldsymbol{x}$, (iv) $f \in \mathcal{D}_{\text{sym}} \cup \mathcal{D}_{\text{Sch-2}}$, then for all $\mathcal{I}$*

$$[f(\boldsymbol{C}(\text{FCFS}))|\mathcal{I}] \leq_{\text{st}} [2f(\boldsymbol{C}(\pi))|\mathcal{I}], \; \forall \; \pi \in \Pi.$$

$$\mathbb{E}[f(\boldsymbol{C}(\text{FCFS}))|\mathcal{I}] \leq \min_{\pi \in \Pi} 2\mathbb{E}\left[f(\boldsymbol{C}(\pi))|\mathcal{I}\right].$$

There exist a fair large class of delay metrics satisfying the conditions (iii) and (iv) of Theorem 6. For example, consider the p-norm of delay $D_p(\boldsymbol{C}(\pi)) = D_p(\boldsymbol{D}(\pi) + \boldsymbol{a}) = ||\boldsymbol{D}(\pi)||_p$ for $p \geq 1$. It is known that any norm is sub-additive, i.e., $||\boldsymbol{x} + \boldsymbol{y}|| \leq ||\boldsymbol{x}|| + ||\boldsymbol{y}||$. In addition, $||\boldsymbol{D}(\pi)||_p$ is symmetric and convex in the delay vector $\boldsymbol{D}(\pi)$, and hence is Schur-convex in $\boldsymbol{D}(\pi)$. Therefore, $f = D_p$ satisfies the conditions (iii) and (iv) of Theorem 6 for all $p \geq 1$.

IV. PROOFS OF THE MAIN RESULTS

We propose a unified sample-path method to prove the main results in the previous section. This method contains three steps:

1. **Sample-Path Orderings:** We first introduce several novel sample-path orderings (Propositions 2, 4, and Corollary 4). Each of these sample-path orderings can be used to obtain a delay inequality for comparing the delay performance (i.e., average delay, maximum lateness, or maximum delay) of different policies in a near-optimal sense.

2. **Sufficient Conditions for Sample-Path Orderings:** In order to minimize delay, it is important to execute the tasks as fast as possible. Motivated by this, we introduce a weak work-efficiency ordering to compare the efficiency of task executions in different policies. By combining this weak work-efficiency ordering with appropriate priority rules (i.e., FUT, EDD, FCFS) for job services, we obtain several sufficient conditions (Propositions 5-6) of the sample-path orderings in *Step 1*. In addition, if more than one sufficient conditions are satisfied simultaneously, we are able to obtain delay inequalities for comparing more general classes of delay metrics achieved by different policies (Propositions 7-8).

3. **Coupling Arguments:** We use coupling arguments to prove that for NBU task service times, the weak work-efficiency ordering is satisfied in the sense of stochastic ordering. By combining this

with the priority rules (i.e., FUT, EDD, FCFS) for job services, we are able to prove the sufficient conditions in *Step 2* in the sense of stochastic ordering. By this, the main results of this paper are proven.

This sample-path method is quite general. In particular, *Step 1* and *Step 2* do not need to specify the queueing system model, and can be potentially used for establishing near delay optimality results in other queueing systems. One application of this sample-path method is in [25], which considered the problem of scheduling in queueing systems with replications.

A. Step 1: Sample-path Orderings

We first propose several sample-path orderings to compare the delay performance of different scheduling policies. Let us first define the system state of any policy $\pi \in \Pi$.

Definition 2. At any time instant $t \in [0, \infty)$, the system state of policy π is specified by a pair of n-dimensional vectors $\boldsymbol{\xi}_\pi(t) = (\xi_{1,\pi}(t), \ldots, \xi_{n,\pi}(t))$ and $\boldsymbol{\gamma}_\pi(t) = (\gamma_{1,\pi}(t), \ldots, \gamma_{n,\pi}(t))$ with non-negative components, where n is the total number of jobs and can be either finite or infinite. The components of $\boldsymbol{\xi}_\pi(t)$ and $\boldsymbol{\gamma}_\pi(t)$ are interpreted as follows: If job i is present in the system at time t, then $\xi_{i,\pi}(t)$ is the number of *remaining* tasks (which are either stored in the queue or being executed by the servers) of job i, and $\gamma_{i,\pi}(t)$ is the number of *unassigned* tasks (which are stored in the queue and not being executed by the servers) of job i; if job i is not present in the system at time t (i.e., job i has not arrived at the system or has departed from the system), then $\xi_{i,\pi}(t) = \gamma_{i,\pi}(t) = 0$. Hence, for all $i = 1, \ldots, n$, $\pi \in \Pi$, and $t \in [0, \infty)$

$$0 \leq \gamma_{i,\pi}(t) \leq \xi_{i,\pi}(t) \leq k_i. \tag{24}$$

Let $\{\boldsymbol{\xi}_\pi(t), \boldsymbol{\gamma}_\pi(t), t \in [0, \infty)\}$ denote the state process of policy π in a probability space $(\Omega, \mathcal{F}, P)$, which is assumed to be right-continuous. The realization of the state process on a sample path $\omega \in \Omega$ can be expressed as $\{\boldsymbol{\xi}_\pi(\omega, t), \boldsymbol{\gamma}_\pi(\omega, t), t \in [0, \infty)\}$. To ease the notational burden, we will omit ω henceforth and reuse $\{\boldsymbol{\xi}_\pi(t), \boldsymbol{\gamma}_\pi(t), t \in [0, \infty)\}$ to denote the realization of the state process on a sample path. Because the system starts to operate at time $t = 0$, there is no job in the system before time $t = 0$. Hence, $\boldsymbol{\xi}_\pi(0^-) = \boldsymbol{\gamma}_\pi(0^-) = \mathbf{0}$.

The following proposition provides one condition (25) for comparing the average delay $D_{\text{avg}}(\boldsymbol{c}(\pi))$ of different policies on a sample path, which was firstly used in [3] to prove the optimality of the preemptive SRPT policy to minimize the average delay in single-server queueing systems.

Proposition 1. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

$$\sum_{i=j}^{n} \xi_{[i],P}(t) \leq \sum_{i=j}^{n} \xi_{[i],\pi}(t), \ \forall \ j = 1, 2, \ldots, n, \tag{25}$$

holds for all $t \in [0, \infty)$,[6] where $\xi_{[i],\pi}(t)$ is the i-th largest component of $\boldsymbol{\xi}_{\pi}(t)$, then

$$c_{(i)}(P) \leq c_{(i)}(\pi), \ \forall \ i = 1, 2, \ldots, n, \tag{26}$$

where $c_{(i)}(\pi)$ is the i-th smallest component of $\boldsymbol{c}(\pi)$.[7] Hence,

$$D_{\text{avg}}(\boldsymbol{c}(P)) \leq D_{\text{avg}}(\boldsymbol{c}(\pi)). \tag{27}$$

Proof. Suppose that there are l unfinished jobs at time t in policy π, then $\sum_{i=l+1}^{n} \xi_{[i],\pi}(t) = 0$. By (25), we get $\sum_{i=l+1}^{n} \xi_{[i],P}(t) = 0$ and hence there are at most l unfinished jobs in policy P. In other words, there are at least as many unfinished jobs in policy π as in policy P at any time $t \in [0, \infty)$. This implies (26), because the sequence of job arrival times $a_1, a_2, \ldots, a_n$ are invariant under any policy. In addition, (27) follows from (26), which completes the proof. □

The sample-path ordering (25) is quite insightful. According to Proposition 1, if (25) holds for all policies $\pi \in \Pi$ and all sample paths $\omega \in \Omega$, then policy P is sample-path delay-optimal for minimizing the average delay $D_{\text{avg}}(\boldsymbol{c}(\pi))$. Interestingly, Proposition 1 is also necessary: If (25) does not hold at some time t, then one can construct an arrival process after time t such that (26) and (27) do not hold [3].

The sample-path ordering (25) has been successfully used to analyze single-server queueing systems [3]. However, applying this condition in multi-server queueing systems is challenging. Due to this fundamental difficulty, we consider an alternative method to relax the sample-path ordering (25) and seek for near delay optimality.

Proposition 2. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

$$\sum_{i=j}^{n} \gamma_{[i],P}(t) \leq \sum_{i=j}^{n} \xi_{[i],\pi}(t), \ \forall \ j = 1, 2, \ldots, n, \tag{28}$$

holds for all $t \in [0, \infty)$, where $\gamma_{[i],\pi}(t)$ is the i-th largest component of $\boldsymbol{\gamma}_{\pi}(t)$, then

$$v_{(i)}(P) \leq c_{(i)}(\pi), \ \forall \ i = 1, 2, \ldots, n, \tag{29}$$

---

[6]In majorization theory [18], (25) is equivalent to "$\boldsymbol{\xi}_{\pi}(t)$ is weakly supermajorized by $\boldsymbol{\xi}_P(t)$, i.e., $\boldsymbol{\xi}_{\pi}(t) \prec^{\text{w}} \boldsymbol{\xi}_P(t)$".

[7]In other words, $c_{(i)}(\pi)$ is the earliest time in policy π by which i jobs have been completed.

where $v_{(i)}(P)$ is the i-th smallest component of $\boldsymbol{v}(P)$.[8] Hence,

$$D_{\mathrm{avg}}(\boldsymbol{v}(P)) \leq D_{\mathrm{avg}}(\boldsymbol{c}(\pi)). \tag{30}$$

Proof. See Appendix B. □

Hence, by relaxing the sample-path ordering (25) as (28), a relaxed delay inequality (30) is obtained which can be used to compare the average delay of policy P and policy π in a near-optimal sense.

Similarly, we develop two sample-path orderings in the following two lemmas to compare the maximum lateness $L_{\mathrm{max}}(\cdot)$ achieved by different policies.

Proposition 3. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

$$\sum_{i:d_i \leq \tau} \xi_{i,P}(t) \leq \sum_{i:d_i \leq \tau} \xi_{i,\pi}(t), \ \forall \ \tau \in [0, \infty), \tag{31}$$

holds for all $t \in [0, \infty)$, then

$$L_{\mathrm{max}}(\boldsymbol{c}(P)) \leq L_{\mathrm{max}}(\boldsymbol{c}(\pi)). \tag{32}$$

Proof. See Appendix C. □

Proposition 4. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

$$\sum_{i:d_i \leq \tau} \gamma_{i,P}(t) \leq \sum_{i:d_i \leq \tau} \xi_{i,\pi}(t), \ \forall \ \tau \in [0, \infty), \tag{33}$$

holds for all $t \in [0, \infty)$, then

$$L_{\mathrm{max}}(\boldsymbol{v}(P)) \leq L_{\mathrm{max}}(\boldsymbol{c}(\pi)). \tag{34}$$

Proof. See Appendix C. □

If $d_i = a_i$ for all i, the maximum lateness $L_{\mathrm{max}}(\cdot)$ reduces to the maximum delay $D_{\mathrm{max}}(\cdot)$. Hence, we can obtain

Corollary 3. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

$$\sum_{i:a_i \leq \tau} \xi_{i,P}(t) \leq \sum_{i:a_i \leq \tau} \xi_{i,\pi}(t), \ \forall \ \tau \in [0, \infty), \tag{35}$$

---

[8]In other words, $v_{(i)}(P)$ is the earliest time in policy P that there exist i jobs whose tasks have all started service.

holds for all $t \in [0, \infty)$, then

$$D_{\max}(\boldsymbol{c}(P)) \leq D_{\max}(\boldsymbol{c}(\pi)). \tag{36}$$

Corollary 4. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

$$\sum_{i:a_i \leq \tau} \gamma_{i,P}(t) \leq \sum_{i:a_i \leq \tau} \xi_{i,\pi}(t), \ \forall \ \tau \in [0, \infty), \tag{37}$$

holds for all $t \in [0, \infty)$, then

$$D_{\max}(\boldsymbol{v}(P)) \leq D_{\max}(\boldsymbol{c}(\pi)). \tag{38}$$

The proofs of Corollaries 3-4 are omitted, because they follow directly from Propositions 3-4 by setting $d_i = a_i$ for all i.

The sample-path orderings in Propositions 1-4 and Corollaries 3-4 are of similar forms. Their distinct features are

- In the sample-path orderings (25) and (28) corresponding to the average delay $D_{\mathrm{avg}}(\cdot)$, the summations are taken over the jobs with the fewest remaining/unassigned tasks;
- In the sample-path orderings (31) and (33) corresponding to the maximum lateness $L_{\max}(\cdot)$, the summations are taken over the jobs with the earliest due times;
- In the sample-path orderings (35) and (37) corresponding to the maximum delay $D_{\max}(\cdot)$, the summations are taken over the jobs with the earliest arrival times.

These features are tightly related to the priority rules for minimizing the corresponding delay metrics near optimally: The priority rule for minimizing the average delay $D_{\mathrm{avg}}(\cdot)$ is FUT first; the priority rule for minimizing the maximum lateness $L_{\max}(\cdot)$ is EDD first; the priority rule for minimizing the maximum delay $D_{\max}(\cdot)$ is FCFS. Hence, *the summations in these sample-path orderings are taken over the high priority jobs*. This is one key insight behind these sample-path orderings.

A number of popular sample-path methods — such as forward induction, backward induction, and interchange arguments [17] — have been successfully used to establish delay optimality results in single-server queueing systems [2], [4], [5]. However, it appears to be difficult to directly generalize these methods and characterize the sub-optimal delay gap from the optimum. On the other hand, the sample-path orderings (25), (28), (31), (33), (35), and (37) provide an interesting unified framework for sample-path delay comparisons, aiming for both delay optimality and near delay optimality. To the best of our knowledge, except for (25) developed in [3], the sample-path orderings (28), (31), (33), (35), and (37) have not appeared before.
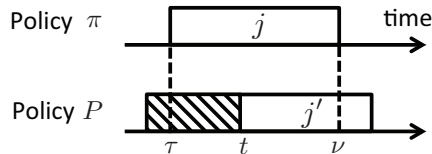
Fig. 3: Illustration of the weak work-efficiency ordering, where the service duration of a task is indicated by a rectangle. Task j starts service at time τ and completes service at time ν in policy π, and one corresponding task j' starts service at time $t \in [\tau, \nu]$ in policy P.

B. Step 2: Sufficient Conditions for Sample-path Orderings

Next, we provide several sufficient conditions for the sample-path orderings (28), (33), and (37). In addition, we will also develop several sufficient conditions for comparing more general delay metrics in $\mathcal{D}_{\text{sym}}$, $\mathcal{D}_{\text{Sch-1}}$, and $\mathcal{D}_{\text{Sch-2}}$.

1) Weak Work-efficiency Ordering: In single-server queueing systems, the service delay is largely governed by the work conservation law (or its generalizations): At any time, the expected total amount of time for completing the jobs in the queue is invariant among all work-conserving policies [22]–[24]. However, this work conservation law does not hold in queueing systems with parallel servers, where it is difficult to efficiently utilize the full service capacity (some servers can be idle if all tasks are being processed by the remaining servers). In order to minimize delay, the system needs to execute the tasks as fast as possible. In the sequel, we introduce an ordering to compare the efficiency of task executions in different policies in a near-optimal sense, which is called *weak work-efficiency ordering*.[9]

Definition 3. Weak Work-efficiency Ordering: For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, policy P is said to be **weakly more work-efficient than** policy π, if the following assertion is true: For each task j executed in policy π, if

1. In policy π, task j starts service at time τ and completes service at time ν ($\tau \leq \nu$),
2. In policy P, the queue is not empty (there exist unassigned tasks in the queue) during $[\tau, \nu]$,

then there exists one corresponding task j' in policy P which starts service during $[\tau, \nu]$.

An illustration of this weak work-efficiency ordering is provided in Fig. 3. Note that the weak work-efficient ordering requires the service starting time of task j' in policy P to be within the service duration of its corresponding task j in policy π. This is the key feature that enables establishing tight sub-optimality delay gap later on.

---

[9]Work-efficiency orderings are also used in [25] to study queueing systems with replications.

2) Sufficient Conditions for Sample-path Orderings: Using weak work-efficiency ordering, we can obtain the following sufficient condition for the sample-path ordering (28) associated to the average delay $D_{\mathrm{avg}}(\cdot)$.

Proposition 5. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

1. *Policy P is weakly more work-efficient than policy π,*

2. *In policy P, each task starting service is from the job with the fewest unassigned tasks among all jobs with unassigned tasks,*

then (28)-(30) hold.

Proof. See Appendix D. □

Similarly, one sufficient condition is obtained for the sample-path orderings (33) for comparing the maximum lateness $L_{\mathrm{max}}(\cdot)$ of different policies.

Proposition 6. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

1. *Policy P is weakly more work-efficient than policy π,*

2. *In policy P, each task starting service is from the job with the earliest due time among all jobs with unassigned tasks,*

then (33) and (34) hold.

Proof. See Appendix E. □

3) More General Delay Metrics: We now investigate more general delay metrics in $\mathcal{D}_{\mathrm{sym}}$, $\mathcal{D}_{\mathrm{Sch\text{-}1}}$, and $\mathcal{D}_{\mathrm{Sch\text{-}2}}$. First, Proposition 5 can be directly generalized to all delay metrics in $\mathcal{D}_{\mathrm{sym}}$.

Proposition 7. *If the conditions of Proposition 5 are satisfied, then for all $f \in \mathcal{D}_{\mathrm{sym}}$*

$$f(\boldsymbol{v}(P)) \leq f(\boldsymbol{c}(\pi)).$$

Proof. See Appendix F. □

In some scenarios, there exists a policy P simultaneously satisfying the sufficient conditions in Proposition 5 (or Proposition 7) and Proposition 6. In this case, we can obtain a delay inequality for comparing any delay metric in $\mathcal{D}_{\mathrm{sym}} \cup \mathcal{D}_{\mathrm{Sch\text{-}1}}$.

Proposition 8. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

1. *Policy P is weakly more work-efficient than policy π,*

2. *In policy P, each task starting service is from the job with the fewest unassigned tasks among all jobs with unassigned tasks, and this job is also the job with the earliest due time among all jobs with unassigned tasks,*

then for all $f \in \mathcal{D}_{\text{sym}} \cup \mathcal{D}_{\text{Sch-1}}$

$$f(\boldsymbol{v}(P)) \leq f(\boldsymbol{c}(\pi)). \tag{39}$$

Proof sketch of Proposition 8. For any $f \in \mathcal{D}_{\text{sym}}$, (29) and (39) follow from Propositions 5 and 7. For any $f \in \mathcal{D}_{\text{Sch-1}}$, we construct an n-dimensional vector $\boldsymbol{v}'$ and show that

$$\boldsymbol{v}(P) - \boldsymbol{d} \prec \boldsymbol{v}' - \boldsymbol{d} \leq \boldsymbol{c}(\pi) - \boldsymbol{d}, \tag{40}$$

where the first majorization ordering in (40) follows from the rearrangement inequality [18, Theorem 6.F.14], [26], and the second inequality in (40) is proven by using (29). This further implies

$$\boldsymbol{v}(P) - \boldsymbol{d} \prec_{\text{w}} \boldsymbol{c}(\pi) - \boldsymbol{d}. \tag{41}$$

Using this, we can show that (39) holds for all $f \in \mathcal{D}_{\text{Sch-1}}$. The details are provided in Appendix G. $\square$

Using this proposition, we can obtain

Corollary 5. *For any given job parameters $\mathcal{I}$ and a sample path of two policies $P, \pi \in \Pi$, if*

1. $k_1 = \ldots = k_n = 1$ *(or $d_1 \leq d_2 \leq \ldots \leq d_n$ and $k_1 \leq k_2 \leq \ldots \leq k_n$),*

2. *Policy P is weakly more work-efficient than policy π,*

3. *In policy P, each task starting service is from the job with the earliest due time among all jobs with unassigned tasks,*

then for all $f \in \mathcal{D}_{\text{sym}} \cup \mathcal{D}_{\text{Sch-1}}$

$$f(\boldsymbol{v}(P)) \leq f(\boldsymbol{c}(\pi)). \tag{42}$$

Proof. If $k_1 = \ldots = k_n = 1$, each job has only one task. Hence, the job with the earliest due time is also one job with the fewest unassigned tasks. If $d_1 \leq d_2 \leq \ldots \leq d_n$, $k_1 \leq k_2 \leq \ldots \leq k_n$, and each task starting service is from the job with the earliest due time among all jobs with unassigned tasks, then each task starting service is also from the job with the fewest unassigned tasks among all jobs with unassigned tasks. By using Proposition 8, Corollary 5 follows. This completes the proof. $\square$

C. Step 3: Coupling Arguments

The following coupling lemma plays one important role in the proof of our main results.

Lemma 2. *Consider two policies $P, \pi \in \Pi$. If policy P is work-conserving, the task service times are NBU, independent across the servers, and i.i.d. across the tasks assigned to the same server, then there exist two state processes $\{\boldsymbol{\xi}_{P_1}(t), \boldsymbol{\gamma}_{P_1}(t), t \in [0, \infty)\}$ and $\{\boldsymbol{\xi}_{\pi_1}(t), \boldsymbol{\gamma}_{\pi_1}(t), t \in [0, \infty)\}$ of policy P_1 and policy π_1, such that*

1. *The state process $\{\boldsymbol{\xi}_{P_1}(t), \boldsymbol{\gamma}_{P_1}(t), t \in [0, \infty)\}$ of policy P_1 has the same distribution with the state process $\{\boldsymbol{\xi}_P(t), \boldsymbol{\gamma}_P(t), t \in [0, \infty)\}$ of policy P,*

2. *The state process $\{\boldsymbol{\xi}_{\pi_1}(t), \boldsymbol{\gamma}_{\pi_1}(t), t \in [0, \infty)\}$ of policy π_1 has the same distribution with the state process $\{\boldsymbol{\xi}_\pi(t), \boldsymbol{\gamma}_\pi(t), t \in [0, \infty)\}$ of policy π,*

3. *Policy P_1 is weakly more work-efficient than policy π_1 with probability one.*

Proof. See Appendix H. □

Now, we are ready to prove our main results.

Proof of Theorem 1. According to lemma 2, for any policy $\pi \in \Pi$, there exist two state processes $\{\boldsymbol{\xi}_{\mathrm{FUT}_1}(t), \boldsymbol{\gamma}_{\mathrm{FUT}_1}(t), t \in [0, \infty)\}$ and $\{\boldsymbol{\xi}_{\pi_1}(t), \boldsymbol{\gamma}_{\pi_1}(t), t \in [0, \infty)\}$ of policy FUT_1 and policy π_1, such that (i) the state process $\{\boldsymbol{\xi}_{\mathrm{FUT}_1}(t), \boldsymbol{\gamma}_{\mathrm{FUT}_1}(t), t \in [0, \infty)\}$ of policy FUT_1 has the same distribution with the state process $\{\boldsymbol{\xi}_{\mathrm{FUT}}(t), \boldsymbol{\gamma}_{\mathrm{FUT}}(t), t \in [0, \infty)\}$ of policy FUT, (ii) the state process $\{\boldsymbol{\xi}_{\pi_1}(t), \boldsymbol{\gamma}_{\pi_1}(t), t \in [0, \infty)\}$ of policy π_1 has the same distribution with the state process $\{\boldsymbol{\xi}_\pi(t), \boldsymbol{\gamma}_\pi(t), t \in [0, \infty)\}$ of policy π, and (iii) policy FUT_1 is weakly more work-efficient than policy π_1 with probability one. By (iii) and Proposition 7, for all $f \in \mathcal{D}_{\mathrm{sym}}$

$$\Pr[f(\boldsymbol{V}(\mathrm{FUT}_1)) \leq f(\boldsymbol{C}(\pi_1))|\mathcal{I}] = 1.$$

By (i), $f(\boldsymbol{V}(\mathrm{FUT}_1))$ has the same distribution with $f(\boldsymbol{V}(\mathrm{FUT}))$. By (ii), $f(\boldsymbol{C}(\pi_1))$ has the same distribution with $f(\boldsymbol{C}(\pi))$. Using the property of stochastic ordering [27, Theorem 1.A.1], we can obtain (15). This completes the proof. □

Proof of Theorem 2. After Theorem 1 is established, Theorem 2 is proven in Appendix I. □

Proof of Theorem 3. According to lemma 2, for any policy $\pi \in \Pi$, there exist two state processes $\{\boldsymbol{\xi}_{\mathrm{EDD}_1}(t), \boldsymbol{\gamma}_{\mathrm{EDD}_1}(t), t \in [0, \infty)\}$ and $\{\boldsymbol{\xi}_{\pi_1}(t), \boldsymbol{\gamma}_{\pi_1}(t), t \in [0, \infty)\}$ of policy EDD_1 and policy π_1, such that (i) the state process $\{\boldsymbol{\xi}_{\mathrm{EDD}_1}(t), \boldsymbol{\gamma}_{\mathrm{EDD}_1}(t), t \in [0, \infty)\}$ of policy EDD_1 has the same distribution with the state process $\{\boldsymbol{\xi}_{\mathrm{EDD}}(t), \boldsymbol{\gamma}_{\mathrm{EDD}}(t), t \in [0, \infty)\}$ of policy EDD, (ii) the state process $\{\boldsymbol{\xi}_{\pi_1}(t), \boldsymbol{\gamma}_{\pi_1}(t), t \in [0, \infty)\}$ of policy π_1 has the same distribution with the state process $\{\boldsymbol{\xi}_\pi(t), \boldsymbol{\gamma}_\pi(t), t \in [0, \infty)\}$ of policy

π, and (iii) policy EDD$_1$ is weakly more work-efficient than policy π_1 with probability one. By (iii) and Proposition 6,

$$\Pr[L_{\max}(\boldsymbol{V}(\text{EDD}_1)) \leq L_{\max}(\boldsymbol{C}(\pi_1))|\mathcal{I}] = 1.$$

By (i), $L_{\max}(\boldsymbol{V}(\text{EDD}_1))$ has the same distribution with $L_{\max}(\boldsymbol{V}(\text{EDD}))$. By (ii), $L_{\max}(\boldsymbol{C}(\pi_1))$ has the same distribution with $L_{\max}(\boldsymbol{C}(\pi))$. Using the property of stochastic ordering [27, Theorem 1.A.1], we can obtain (20). This completes the proof. □

Proof of Theorem 4. By replacing Proposition 6 with Proposition 5 in the proof of Theorem 3, Theorem 4 is proven. □

V. Conclusions

In this paper, we present a comprehensive study on near delay-optimal scheduling of batch jobs in queueing systems with parallel servers. We prove that for NBU task service time distributions and for arbitrarily given job parameters, the FUT policy, EDD policy, and the FCFS policy are near delay-optimal in stochastic ordering for minimizing average delay, maximum lateness, and maximum delay, respectively, among all causal and non-preemptive policies. If the job parameters satisfy some additional conditions, we can further show that these policies are near delay-optimal in stochastic ordering for minimizing three general classes of delay metrics.

The key tools for showing these results are some novel sample-path conditions for comparing the delay performance of different policies. These sample-path conditions do not need to specify the queueing system model and hence can potentially be applied to obtain near delay-optimal results for other queueing systems. One application of these sample-path conditions is in [25], which considered the problem of scheduling batch jobs in multi-server queueing systems with replications.

Acknowledgements

Appendix A
Proof of Lemma 1

We will need the following lemma:

Lemma 3. *Suppose that $X_1, \ldots, X_m$ are non-negative independent random variables, $\chi_1, \ldots, \chi_m$ are arbitrarily given non-negative constants, $R_l = [X_l - \chi_l | X_l > \chi_l]$ for $l = 1, \ldots, m$, then $R_1, \ldots, R_m$ are mutually independent.*

Proof. For all constants $t_l \geq 0$, $l = 1, \ldots m$, we have

$$\Pr[R_l > t_l, l = 1, \ldots, m]$$

$$= \Pr[X_l - \chi_l > t_l, l = 1, \ldots, m | X_l > \chi_l, l = 1, \ldots, m]$$

$$= \frac{\Pr[X_l > t_l + \chi_l, l = 1, \ldots, m]}{\Pr[X_l > \chi_l, l = 1, \ldots, m]}$$

$$= \frac{\prod_{l=1}^{m} \Pr[X_l > t_l + \chi_l]}{\prod_{l=1}^{m} \Pr[X_l > \chi_l]}$$

$$= \prod_{l=1}^{m} \Pr[X_l - \chi_l > t_l | X_l > \chi_l]$$

$$= \prod_{l=1}^{m} \Pr[R_l > t_l]. \tag{43}$$

Hence, $R_1, \ldots, R_m$ are mutually independent. $\qquad\square$

Proof of lemma 1. Define a function $g_2(\boldsymbol{x}) = f(\boldsymbol{x} + \boldsymbol{a})$. Because $g_2(\boldsymbol{x}) = f(\boldsymbol{x} + \boldsymbol{a})$ is sub-additive in $\boldsymbol{x}$, i.e., $g_2(\boldsymbol{x} + \boldsymbol{y}) \leq g_2(\boldsymbol{x}) + g_2(\boldsymbol{y})$, we can obtain

$$f(\boldsymbol{C}(P))$$

$$= g_2([\boldsymbol{V}(P) - \boldsymbol{a}] + [\boldsymbol{C}(P) - \boldsymbol{V}(P)])$$

$$\leq g_2(\boldsymbol{V}(P) - \boldsymbol{a}) + g_2(\boldsymbol{C}(P) - \boldsymbol{V}(P))$$

$$= f(\boldsymbol{V}(P)) + g_2(\boldsymbol{C}(P) - \boldsymbol{V}(P)). \tag{44}$$

At time $V_i(P)$, all tasks of job i have started service. If $k_i > m$, then job i has at most m incomplete tasks at time $V_i(P)$ because there are only m servers. If $k_i \leq m$, then job i has at most k_i incomplete tasks at time $V_i(P)$. Therefore, at most $k_i \wedge m$ tasks of job i are in service at time $V_i(P)$. Let $X_{i,l}$ denote the random service time of the l-th remaining task of job i. We will prove that for all $\mathcal{I}$

$$[\boldsymbol{C}(P) - \boldsymbol{V}(P) | \mathcal{I}] \leq_{\text{st}} \left[\left(\max_{l=1,\ldots,k_1 \wedge m} X_{1,l}, \ldots, \max_{l=1,\ldots,k_n \wedge m} X_{n,l} \right) \middle| k_1, \ldots, k_n \right]. \tag{45}$$

Without loss of generality, suppose that at time $V_i(P)$, the remaining tasks of job i are being executed by the set of servers $\mathcal{S}_i \subseteq \{1, \ldots, m\}$, which satisfies $|\mathcal{S}_i| \leq k_i \wedge m$. Hence, the $X_{i,l}$'s are *i.i.d.* In policy P, let $\chi_{i,l}$ denote the amount of time that was spent on executing the l-th remaining task of job i before

time $V_i(P)$, and let $R_{i,l}$ denote the remaining service time to complete this task after time $V_i(P)$. Then, $R_{i,l}$ can be expressed as $R_{i,l} = [X_{i,l} - \chi_{i,l} | X_{i,l} > \chi_{i,l}]$. Suppose that in policy P, $V_{j_i}(P)$ associated with job j_i is the i-th smallest component of the vector $\boldsymbol{V}(P)$, i.e., $V_{j_i}(P) = V_{(i)}(P)$. We prove (45) by using an inductive argument supported by Theorem 6.B.3 of [27], which contains two steps.

Step 1: We first show that for all $\mathcal{I}$

$$[C_{j_1}(P) - V_{j_1}(P)|\mathcal{I}] \leq_{\text{st}} \left[\max_{l=1,\ldots,k_{j_1} \wedge m} X_{j_1,l} \middle| \mathcal{I} \right]. \tag{46}$$

Because the $X_{i,l}$'s are *i.i.d.* NBU, for all realizations of j_1 and $\chi_{j_1,l}$, we have

$$[R_{j_1,l}|j_1, \chi_{j_1,l}] \leq_{\text{st}} [X_{j_1,l}|j_1], \ \forall \ l \in \mathcal{S}_{j_1}. \tag{47}$$

Lemma 3 tells us that the $R_{j_1,l}$'s are conditional independent for any given realization of $\{\chi_{j_1,l}, l \in \mathcal{S}_{j_1}\}$. Hence, for all realizations of j_1, $\mathcal{S}_{j_1}$, and $\{\chi_{j_1,l}, l \in \mathcal{S}_{j_1}\}$

$$[C_{j_1}(P) - V_{j_1}(P)|j_1, \mathcal{S}_{j_1}, \{\chi_{j_1,l}, l \in \mathcal{S}_{j_1}\}]$$

$$= [\max_{l \in \mathcal{S}_{j_1}} R_{j_1,l}|j_1, \mathcal{S}_{j_1}, \{\chi_{j_1,l}, l \in \mathcal{S}_{j_1}\}]$$

$$\leq_{\text{st}} [\max_{l \in \mathcal{S}_{j_1}} X_{j_1,l}|j_1, \mathcal{S}_{j_1}] \tag{48}$$

$$\leq [\max_{l=1,\ldots,k_{j_1} \wedge m} X_{j_1,l}|j_1], \tag{49}$$

where in (48) we have used (47) and Theorem 6.B.14 of [27], and in (49) we have used $|\mathcal{S}_i| \leq k_i \wedge m$. Because j_1, $\mathcal{S}_{j_1}$, and $\{\chi_{j_1,l}, l \in \mathcal{S}_{j_1}\}$ are random variables which depend on the job parameters $\mathcal{I}$, by using Theorem 6.B.16(e) of [27], (46) is proven.

Step 2: We show that for all given $\mathcal{I}$ *and* $\{C_{j_i}(P), V_{j_i}(P), i = 1, \ldots h\}$

$$[C_{j_{h+1}}(P) - V_{j_{h+1}}(P)|\mathcal{I}, \{C_{j_i}(P), V_{j_i}(P), i = 1, \ldots h\}] \leq_{\text{st}} \left[\max_{l=1,\ldots,k_{j_{h+1}} \wedge m} X_{j_{h+1},l} \middle| \mathcal{I} \right]. \tag{50}$$

Similar with (49), for all realizations of j_{h+1}, $\mathcal{S}_{j_{h+1}}$, and $\{\chi_{j_{h+1},l}, l \in \mathcal{S}_{j_{h+1}}\}$, we can get

$$[C_{j_{h+1}}(P) - V_{j_{h+1}}(P)|j_{h+1}, \mathcal{S}_{j_{h+1}}, \{\chi_{j_{h+1},l}, l \in \mathcal{S}_{j_{h+1}}\}] \leq_{\text{st}} [\max_{l=1,\ldots,k_{j_{h+1}} \wedge m} X_{j_{h+1},l}|j_{h+1}].$$

Because j_{h+1}, $\mathcal{S}_{j_{h+1}}$, and $\{\chi_{j_{h+1},l}, l \in \mathcal{S}_{j_{h+1}}\}$ are random variables which are determined by $\mathcal{I}$ and $\{C_{j_i}(P), V_{j_i}(P), i = 1, \ldots h\}$, by using Theorem 6.B.16(e) of [27], (50) is proven. By (46), (50), and

Theorem 6.B.3 of [27], we have

$$[\boldsymbol{C}(P) - \boldsymbol{V}(P)|\mathcal{I}]$$

$$\leq_{\text{st}} \left[\left(\max_{l=1,\ldots,k_1 \wedge m} X_{1,l}, \ldots, \max_{l=1,\ldots,k_n \wedge m} X_{n,l} \right) \middle| \mathcal{I} \right]$$

$$= \left[\left(\max_{l=1,\ldots,k_1 \wedge m} X_{1,l}, \ldots, \max_{l=1,\ldots,k_n \wedge m} X_{n,l} \right) \middle| k_1, \ldots, k_n \right].$$

Hence, (45) holds in policy P.

In policy π, k_i tasks of job i must be accomplished to complete job i. Hence, using the above arguments again, yields

$$\left[\left(\max_{l=1,\ldots,k_1 \wedge m} X_{1,l}, \ldots, \max_{l=1,\ldots,k_n \wedge m} X_{n,l} \right) \middle| k_1, \ldots, k_n \right] \leq_{\text{st}} [\boldsymbol{C}(\pi) - \boldsymbol{a}|\mathcal{I}]. \tag{51}$$

Combining (46) and (51), we have

$$[\boldsymbol{C}(P) - \boldsymbol{V}(P)|\mathcal{I}] \leq_{\text{st}} [\boldsymbol{C}(\pi) - \boldsymbol{a}|\mathcal{I}].$$

Because g_2 is increasing, by Theorem 6.B.16(a) of [27]

$$[g_2(\boldsymbol{C}(P) - \boldsymbol{V}(P))|\mathcal{I}] \leq_{\text{st}} [g_2(\boldsymbol{C}(\pi) - \boldsymbol{a})|\mathcal{I}] = [f(\boldsymbol{C}(\pi))|\mathcal{I}]. \tag{52}$$

By substituting (22) and (52) into (44), (23) is proven. This completes the proof. $\qquad\square$

APPENDIX B

PROOF OF PROPOSITION 2

Let j be any integer chosen from $\{1, \ldots, n\}$, and y_j be the number of jobs that have arrived by the time $c_{(j)}(\pi)$, where $y_j \geq j$. Because j jobs are completed by the time $c_{(j)}(\pi)$ in policy π, there are exactly $(y_j - j)$ incomplete jobs in the system at time $c_{(j)}(\pi)$. By the definition of the system state, we have $\xi_{[i],\pi}(c_{(j)}(\pi)) = 0$ for $i = y_j - j + 1, \ldots, n$. Hence,

$$\sum_{i=y_j-j+1}^{n} \xi_{[i],\pi}(c_{(j)}(\pi)) = 0.$$

Combining this with (28), yields that policy P satisfies

$$\sum_{i=y_j-j+1}^{n} \gamma_{[i],P}\left(c_{(j)}(\pi)\right) \leq 0. \tag{53}$$

Further, the definition of the system state tells us that $\gamma_{i,P}(t) \geq 0$ holds for all $i = 1, \ldots, n$ and $t \geq 0$. Hence, we have

$$\gamma_{[i],P}\left(c_{(j)}(\pi)\right) = 0, \ \forall \ i = y_j - j + 1, \ldots, n. \tag{54}$$

Therefore, there are at most $y_j - j$ jobs which have unassigned tasks at time $c_{(j)}(\pi)$ in policy P. Because the sequence of job arrival times $a_1, a_2, \ldots, a_n$ are invariant under any policy, y_j jobs have arrived by the time $c_{(j)}(\pi)$ in policy P. Thus, there are at least j jobs which have no unassigned tasks at the time $c_{(j)}(\pi)$ in policy P, which can be equivalently expressed as

$$v_{(j)}(P) \leq c_{(j)}(\pi). \tag{55}$$

Because j is arbitrarily chosen, (55) holds for all $j = 1, \ldots, n$, which is exactly (29). In addition, (30) follows from (29), which completes the proof.

APPENDIX C

PROOFS OF PROPOSITION 3 AND PROPOSITION 4

Proof of Proposition 3. Let w_i be the index of the job associated with the job completion time $c_{(i)}(P)$. In order to prove (32), it is sufficient to show that for each $j = 1, 2, \ldots, n$,

$$c_{w_j}(P) - d_{w_j} \leq \max_{i=1,2,\ldots,n} [c_i(\pi) - d_i]. \tag{56}$$

We prove (56) by contradiction. *For this, let us assume that*

$$c_i(\pi) < c_{w_j}(P) \tag{57}$$

holds for all job i satisfying $a_i \leq c_{w_j}(P)$ and $d_i \leq d_{w_j}$. That is, if job i arrives before time $c_{w_j}(P)$ and its due time is no later than d_{w_j}, then job i is completed before time $c_{w_j}(P)$ in policy π. Define

$$\tau_j = \max_{i: a_i \leq c_{w_j}(P), d_i \leq d_{w_j}} c_i(\pi). \tag{58}$$

According to (57) and (58), we can obtain

$$\tau_j < c_{w_j}(P). \tag{59}$$

On the other hand, (58) tells us that all job i satisfying $d_i \leq d_{w_j}$ and $a_i \leq c_{w_j}(P)$ are completed by

time τ_j in policy π. By this, the system state of policy π satisfies

$$\sum_{i:d_i \leq d_{w_j}} \xi_{i,\pi}(\tau_j) = 0.$$

Combining this with (31), yields

$$\sum_{i:d_i \leq d_{w_j}} \xi_{i,P}(\tau_j) \leq 0. \tag{60}$$

Further, the definition of the system state tells us that $\xi_{i,P}(t) \geq 0$ for all $i = 1, \ldots, n$ and $t \geq 0$. Using this and (60), we get that job w_j satisfies

$$\xi_{w_j,P}(\tau_j) = 0.$$

That is, all tasks of job w_j are completed by time τ_j in policy P. Hence, $c_{w_j}(P) \leq \tau_j$, where contradicts with (59). Therefore, there exists at least one job i satisfying the conditions $a_i \leq c_{w_j}(P)$, $d_i \leq d_{w_j}$, and $c_{w_j}(P) \leq c_i(\pi)$. This can be equivalently expressed as

$$c_{w_j}(P) \leq \max_{i:a_i \leq c_{w_j}(P), d_i \leq d_{w_j}} c_i(\pi). \tag{61}$$

Hence, for each $j = 1, 2, \ldots, n$,

$$
\begin{aligned}
c_{w_j}(P) - d_{w_j} &\leq \max_{i:a_i \leq c_{w_j}(P), d_i \leq d_{w_j}} c_i(\pi) - d_{w_j} \\
&\leq \max_{i:a_i \leq c_{w_j}(P), d_i \leq d_{w_j}} [c_i(\pi) - d_i] \\
&\leq \max_{i=1,2,\ldots,n} [c_i(\pi) - d_i].
\end{aligned}
$$

This implies (32). Hence, Proposition 3 is proven. $\qquad \square$

The proof of Proposition 4 is almost identical with that of Proposition 3, and hence is not repeated here. The only difference is that $c_{w_j}(P)$ and $\boldsymbol{\xi}_P(\tau_j)$ in the proof of Proposition 3 should be replaced by $v_{w_j}(P)$ and $\boldsymbol{\gamma}_P(\tau_j)$, respectively.

APPENDIX D
PROOF OF PROPOSITION 5

The following two lemmas are needed to prove Proposition 5:

Lemma 4. *Suppose that under policy P, $\{\boldsymbol{\xi}'_P, \boldsymbol{\gamma}'_P\}$ is obtained by allocating b_P unassigned tasks to the servers in the system whose state is $\{\boldsymbol{\xi}_P, \boldsymbol{\gamma}_P\}$. Further, suppose that under policy π, $\{\boldsymbol{\xi}'_\pi, \boldsymbol{\gamma}'_\pi\}$ is obtained*

by completing b_π tasks in the system whose state is $\{\boldsymbol{\xi}_\pi, \boldsymbol{\gamma}_\pi\}$. If $b_P \geq b_\pi$, condition 2 of Proposition 5 is satisfied in policy P, and

$$\sum_{i=j}^{n} \gamma_{[i],P} \leq \sum_{i=j}^{n} \xi_{[i],\pi}, \; \forall \; j = 1, 2, \ldots, n,$$

then

$$\sum_{i=j}^{n} \gamma'_{[i],P} \leq \sum_{i=j}^{n} \xi'_{[i],\pi}, \; \forall \; j = 1, 2, \ldots, n. \tag{62}$$

Proof. If $\sum_{i=j}^{n} \gamma'_{[i],P} = 0$, then the inequality (62) follows naturally. If $\sum_{i=j}^{n} \gamma'_{[i],P} > 0$, then there exist unassigned tasks which have not been assigned to any server. In policy P, each task allocated to the servers is from the job with the minimum positive $\gamma_{i,P}$. Hence, $\sum_{i=j}^{n} \gamma'_{[i],P} = \sum_{i=j}^{n} \gamma_{[i],P} - b_P \leq \sum_{i=j}^{n} \xi_{[i],\pi} - b_\pi \leq \sum_{i=j}^{n} \xi'_{[i],\pi}$. $\qquad\square$

Lemma 5. *Suppose that, under policy P, $\{\boldsymbol{\xi}'_P, \boldsymbol{\gamma}'_P\}$ is obtained by adding a job with b tasks to the system whose state is $\{\boldsymbol{\xi}_P, \boldsymbol{\gamma}_P\}$. Further, suppose that, under policy π, $\{\boldsymbol{\xi}'_\pi, \boldsymbol{\gamma}'_\pi\}$ is obtained by adding a job with b tasks to the system whose state is $\{\boldsymbol{\xi}_\pi, \boldsymbol{\gamma}_\pi\}$. If*

$$\sum_{i=j}^{n} \gamma_{[i],P} \leq \sum_{i=j}^{n} \xi_{[i],\pi}, \; \forall \; j = 1, 2, \ldots, n,$$

then

$$\sum_{i=j}^{n} \gamma'_{[i],P} \leq \sum_{i=j}^{n} \xi'_{[i],\pi}, \; \forall \; j = 1, 2, \ldots, n.$$

Proof. Without loss of generalization, we suppose that after the job arrival, b is the l-th largest component of $\boldsymbol{\gamma}'_P$ and the m-th largest component of $\boldsymbol{\xi}'_\pi$, i.e., $\gamma'_{[l],P} = \xi'_{[m],\pi} = b$. We consider the following four cases:

Case 1: $l < j, m < j$. We have $\sum_{i=j}^{n} \gamma'_{[i],P} = \sum_{i=j-1}^{n} \gamma_{[i],P} \leq \sum_{i=j-1}^{n} \xi_{[i],\pi} = \sum_{i=j}^{n} \xi'_{[i],\pi}$.

Case 2: $l < j, m \geq j$. We have $\sum_{i=j}^{n} \gamma'_{[i],P} = \sum_{i=j-1}^{n} \gamma_{[i],P} \leq b + \sum_{i=j}^{n} \gamma_{[i],P} \leq b + \sum_{i=j}^{n} \xi_{[i],\pi} = \sum_{i=j}^{n} \xi'_{[i],\pi}$.

Case 3: $l \geq j, m < j$. We have $\sum_{i=j}^{n} \gamma'_{[i],P} = b + \sum_{i=j}^{n} \gamma_{[i],P} \leq \sum_{i=j-1}^{n} \gamma_{[i],P} \leq \sum_{i=j-1}^{n} \xi_{[i],\pi} = \sum_{i=j}^{n} \xi'_{[i],\pi}$.

Case 4: $l \geq j, m \geq j$. We have $\sum_{i=j}^{n} \gamma'_{[i],P} = b + \sum_{i=j}^{n} \gamma_{[i],P} \leq b + \sum_{i=j}^{n} \xi_{[i],\pi} = \sum_{i=j}^{n} \xi'_{[i],\pi}$. $\qquad\square$

We now use Lemma 4 and Lemma 5 to prove Proposition 5.

Proof of Proposition 5. Assume that no task is completed at the job arrival times a_i for $i = 1, \ldots, n$.

This does not lose any generality, because if a task is completed at time $t_j = a_i$, Proposition 5 can be proven by first proving for the case $t_j = a_i + \epsilon$ and then taking the limit $\epsilon \to 0$. We prove (28) by induction.

Step 1: We will show that (28) *holds during* $[0, a_2)$.[10]

Because $\boldsymbol{\xi}_P(0^-) = \boldsymbol{\gamma}_P(0^-) = \boldsymbol{\xi}_\pi(0^-) = \boldsymbol{\gamma}_\pi(0^-) = \mathbf{0}$, (28) holds at time 0^-. Job 1 arrives at time $a_1 = 0$. By Lemma 5, (28) holds at time 0. Let t be an arbitrarily chosen time during $(0, a_2)$. Suppose that b_π tasks start execution and also complete execution during $[0, t]$ in policy π. We need to consider two cases:

Case 1: The queue is not empty (there exist unassigned tasks in the queue) during $[0, t]$ in policy P. By the weak work-efficiency ordering condition, no fewer than b_π tasks start execution during $[0, t]$ in policy P. Because (28) holds at time 0, by Lemma 4, (28) also holds at time t.

Case 2: The queue is empty (all tasks in the system are in service) by time $t' \in [0, t]$ in policy P. Because $t \in (0, a_2)$ and there is no task arrival during $(0, a_2)$, there is no task arrival during $(t', t]$. Hence, it must hold that all tasks in the system are in service at time t. Then, the system state of policy P satisfies $\sum_{i=j}^n \gamma_{[i], P}(t) = 0$ for all $j = 1, 2, \ldots, n$ at time t. Hence, (28) holds at time t.

In summary of these two cases, (28) holds for all $t \in [0, a_2)$.

Step 2: Assume that for some integer $i \in \{2, \ldots, n\}$, *the conditions of Proposition 5 imply that* (28) *holds for all* $t \in [0, a_i)$. *We will prove that the conditions of Proposition 5 imply that* (28) *holds for all* $t \in [0, a_{i+1})$.

Let t be an arbitrarily chosen time during (a_i, a_{i+1}). We modify the task completion times in policy π as follows: For each pair of corresponding task j and task j' mentioned in the definition of the weak work-efficiency ordering, if

- In policy π, task j starts execution at time $\tau \in [0, a_i]$ and completes execution at time $\nu \in (a_i, t]$,
- In policy P, the queue is not empty (there exist unassigned tasks in the queue) during $[\tau, \nu]$,
- In policy P, the corresponding task j' starts execution at time $t' \in [0, a_i]$,

then the completion time of task j is modified from ν to a_i^- in policy π, as illustrated in Fig. 4.

This modification satisfies the following three claims:

1. The system state of policy π at time t remains the same before and after this modification;
2. Policy P is still weakly more work-efficient than policy π after this modification;
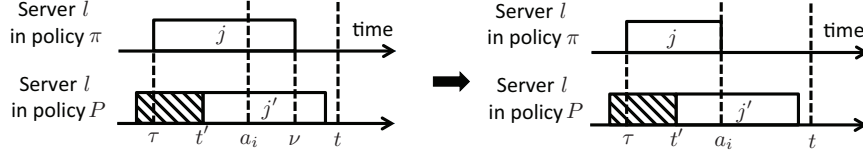
---

[10]Note that $a_1 = 0$.

Fig. 4: Illustration of the modification of task completion times in policy π: If in policy π, task j starts execution at time $\tau \in [0, a_i]$ and completes execution at time $\nu \in (a_i, t]$, and in policy P, task j' starts execution at time $t' \in [0, a_i]$, then the completion time of task j is changed from ν to a_i^- in policy π.

3. If b_π tasks complete execution during $[a_i, t]$ on the modified sample path of policy π, and the queue is not empty (there exist unassigned tasks in the queue) during $[a_i, t]$ in policy P, then no fewer than b_π tasks start execution during $[a_i, t]$ in policy P.

We now prove these three claims. Claim 1 follows from the fact that the tasks completed during $[0, t]$ remain the same before and after this modification. It is easy to prove Claim 2 by checking the definition of work-efficiency ordering. For Claim 3, notice that if a task j starts execution and completes execution during $[a_i, t]$ on the modified sample path of policy π, then by Claim 2, its corresponding task j' must start execution during $[a_i, t]$ in policy P. On the other hand, if a task j starts execution during $[0, a_i]$ and completes execution during $[a_i, t]$ on the modified sample path of policy π, then by the modification, its corresponding task j' must start execution during $[a_i, t]$ in policy P. By combining these two cases, Claim 3 follows.

We use these three claims to prove the statement of *Step 2*. According to Claim 2, policy P is weakly more work-efficient than policy π after the modification. By the assumption of *Step 2*, (28) holds during $[0, a_i)$ for the modified sample path of policy π. Job j arrives at time a_i. By Lemma 5, (28) holds at time a_i for the modified sample path of policy π. Suppose that b_π tasks complete execution during $[a_i, t]$ on the modified sample path of policy π. We need to consider two cases:

Case 1: The queue is not empty (there exist unassigned tasks in the queue) during $[a_i, t]$ in policy P. By Claim 3, no fewer than b_π tasks start execution during $[a_i, t]$ in policy P. Because (28) holds at time a_i, by Lemma 4, (28) also holds at time t for the modified sample path of policy π.

Case 2: The queue is empty (all tasks in the system are in service) at time $t' \in [a_i, t]$ in policy P. Because $t \in (a_i, a_{i+1})$ and $t' \in [a_i, t]$, there is no task arrival during $(t', t]$. Hence, it must hold that all tasks in the system are in service at time t. Then, the system state of policy P satisfies $\sum_{i=j}^{n} \gamma_{[i], P}(t) = 0$ for all $j = 1, 2, \ldots, n$ at time t. Hence, (28) holds at time t for the modified sample path of policy π.

In summary of these two cases, (28) holds at time t for the modified sample path of policy π. By Claim 1, the system state of policy π at time t remains the same before and after this modification.

Hence, (28) holds at time t for the original sample path of policy π. Therefore, if the assumption of *Step 2* is true, then (28) holds for all $t \in [0, a_{i+1})$.

By induction, (28) holds at time $t \in [0, \infty)$. Then, (29) and (30) follow from Proposition 2. This completes the proof. $\square$

APPENDIX E

PROOF OF PROPOSITION 6

The proof of Proposition 6 requires the following two lemmas:

Lemma 6. *Suppose that, in policy P, $\{\boldsymbol{\xi}'_P, \boldsymbol{\gamma}'_P\}$ is obtained by allocating b_P unassigned tasks to the servers in the system whose state is $\{\boldsymbol{\xi}_P, \boldsymbol{\gamma}_P\}$. Further, suppose that, in policy π, $\{\boldsymbol{\xi}'_\pi, \boldsymbol{\gamma}'_\pi\}$ is obtained by completing b_π tasks in the system whose state is $\{\boldsymbol{\xi}_\pi, \boldsymbol{\gamma}_\pi\}$. If $b_P \geq b_\pi$, condition 2 of Proposition 6 is satisfied in policy P, and*

$$\sum_{i:d_i \leq \tau} \gamma_{i,P} \leq \sum_{i:d_i \leq \tau} \xi_{i,\pi}, \ \forall \ \tau \in [0, \infty),$$

then

$$\sum_{i:d_i \leq \tau} \gamma'_{i,P} \leq \sum_{i:d_i \leq \tau} \xi'_{i,\pi}, \ \forall \ \tau \in [0, \infty). \tag{63}$$

Proof. If $\sum_{i:d_i \leq \tau} \gamma'_{i,P} = 0$, then the inequality (63) follows naturally. If $\sum_{i:d_i \leq \tau} \gamma'_{i,P} > 0$, then there exist some unassigned tasks in the queue. In policy P, each task allocated to the servers is from the job with the earliest due time. Hence, $\sum_{i:d_i \leq \tau} \gamma'_{i,P} = \sum_{i:d_i \leq \tau} \gamma_{i,P} - b_P \leq \sum_{i:d_i \leq \tau} \xi_{i,\pi} - b_\pi \leq \sum_{i:d_i \leq \tau} \xi'_{i,\pi}.$ $\square$

Lemma 7. *Suppose that under policy P, $\{\boldsymbol{\xi}'_P, \boldsymbol{\gamma}'_P\}$ is obtained by adding a job with b tasks and due time d to the system whose state is $\{\boldsymbol{\xi}_P, \boldsymbol{\gamma}_P\}$. Further, suppose that under policy π, $\{\boldsymbol{\xi}'_\pi, \boldsymbol{\gamma}'_\pi\}$ is obtained by adding a job with b tasks and due time d to the system whose state is $\{\boldsymbol{\xi}_\pi, \boldsymbol{\gamma}_\pi\}$. If*

$$\sum_{i:d_i \leq \tau} \gamma_{i,P} \leq \sum_{i:d_i \leq \tau} \xi_{i,\pi}, \ \forall \ \tau \in [0, \infty),$$

then

$$\sum_{i:d_i \leq \tau} \gamma'_{i,P} \leq \sum_{i:d_i \leq \tau} \xi'_{i,\pi}, \ \forall \ \tau \in [0, \infty).$$

Proof. If $d \leq \tau$, then $\sum_{i:d_i \leq \tau} \gamma'_{i,P} \leq \sum_{i:d_i \leq \tau} \gamma_{i,P} + b \leq \sum_{i:d_i \leq \tau} \xi_{i,\pi} + b \leq \sum_{i:d_i \leq \tau} \xi'_{i,\pi}.$

If $d > \tau$, then $\sum_{i:d_i \leq \tau} \gamma'_{i,P} \leq \sum_{i:d_i \leq \tau} \gamma_{i,P} \leq \sum_{i:d_i \leq \tau} \xi_{i,\pi} \leq \sum_{i:d_i \leq \tau} \xi'_{i,\pi}.$ $\square$

The proof of Proposition 6 is almost identical with that of Proposition 5, and hence is not repeated here. The only difference is that Lemma 4 and Lemma 5 in the proof of Proposition 5 should be replaced by Lemma 6 and Lemma 7, respectively.

APPENDIX F
PROOF OF PROPOSITION 7

We have proven that (29) holds under the conditions of Proposition 7. Note that (29) can be equivalently expressed in the following vector form:

$$\boldsymbol{v}_\uparrow(P) \leq \boldsymbol{c}_\uparrow(\pi).$$

Because any $f \in \mathcal{D}_{\text{sym}}$ is a symmetric and increasing function, we can obtain

$$f(\boldsymbol{v}(P)) = f(\boldsymbol{v}_\uparrow(P))$$

$$\leq f(\boldsymbol{c}_\uparrow(\pi)) = f(\boldsymbol{c}(\pi)).$$

This completes the proof.

APPENDIX G
PROOF OF PROPOSITION 8

In the proof of Proposition 8, we need to use the following rearrangement inequality:

Lemma 8. *[18, Theorem 6.F.14] Consider two n-dimensional vectors $(x_1,\ldots,x_n)$ and $(y_1,\ldots,y_n)$. If $(x_i - x_j)(y_i - y_j) \leq 0$ for two indices i and j where $1 \leq i < j \leq n$, then*

$$(x_1-y_1,\ldots,x_j-y_i,\ldots,x_i-y_j,\ldots,x_n-y_n)$$

$$\prec(x_1-y_1,\ldots,x_i-y_i,\ldots,x_j-y_j,\ldots,x_n-y_n).$$

Proof of Proposition 8. For $f \in \mathcal{D}_{\text{sym}}$, (29) and (39) follow from Proposition 5 and Proposition 7.

For $f \in \mathcal{D}_{\text{Sch-1}}$, (39) is proven in 3 steps, which are described as follows:

Step 1: We will show that

$$\boldsymbol{v}(P) - \boldsymbol{d} \prec_{\text{w}} \boldsymbol{c}(\pi) - \boldsymbol{d}. \tag{64}$$

According to Eq. (1.A.17) and Theorem 5.A.9 of [18], it is sufficient to show that there exists an n-dimensional vector $\boldsymbol{v}'$ such that

$$\boldsymbol{v}(P) - \boldsymbol{d} \prec \boldsymbol{v}' - \boldsymbol{d} \leq \boldsymbol{c}(\pi) - \boldsymbol{d}. \tag{65}$$

Vector $\boldsymbol{v}'$ is constructed as follows: First, the components of the vector $\boldsymbol{v}'$ is a rearrangement (or permutation) of the components of the vector $\boldsymbol{v}(P)$, which can be equivalently expressed as

$$v'_{(i)} = v_{(i)}(P), \ \forall \ i = 1, \ldots, n. \tag{66}$$

Second, for each $j = 1, \ldots, n$, if the completion time $c_j(\pi)$ of job j is the i_j-th smallest component of $\boldsymbol{c}(\pi)$, i.e.,

$$c_j(\pi) = c_{(i_j)}(\pi), \tag{67}$$

then v'_j associated with job j is the i_j-th smallest component of $\boldsymbol{v}'$, i.e.,

$$v'_j = v'_{(i_j)}. \tag{68}$$

Combining (29) and (66)-(68), yields

$$v'_j = v'_{(i_j)} = v_{(i_j)}(P) \leq c_{(i_j)}(\pi) = c_j(\pi)$$

for $j = 1, \ldots, n$. This implies $\boldsymbol{v}' \leq \boldsymbol{c}(\pi)$, and hence the second inequality in (65) is proven.

The remaining task is to prove the first inequality in (65). First, consider the case that the due times $d_1, \ldots, d_n$ of the n jobs are different from each other. The vector $\boldsymbol{v}(P)$ can be obtained from $\boldsymbol{v}'$ by the following procedure: For each $j = 1, \ldots, n$, define a set

$$S_j = \{i : a_i \leq v_j(P), d_i < d_j\}. \tag{69}$$

If there exist two jobs i and j which satisfy $i \in S_j$ and $v'_i > v'_j$, we interchange the components v'_i and v'_j in vector $\boldsymbol{v}'$. Repeat this interchange operation, until such two jobs i and j satisfying $i \in S_j$ and $v'_i > v'_j$ cannot be found. Therefore, at the end of this procedure, if job i arrives before $v_j(P)$ and job i has an earlier due time than job j, then $v'_i < v'_j$, which is exactly the priority rule of job service satisfied by policy P. Therefore, the vector $\boldsymbol{v}(P)$ is obtained at the end of this procedure. In each interchange operation of this procedure, $(v'_i - v'_j)(d_i - d_j) \leq 0$ is satisfied before the interchange of v'_i and v'_j. By Lemma 8 and the transitivity of the ordering of majorization, we can obtain $\boldsymbol{v}(P) - \boldsymbol{d} \prec \boldsymbol{v}' - \boldsymbol{d}$, which is the first inequality in (65).
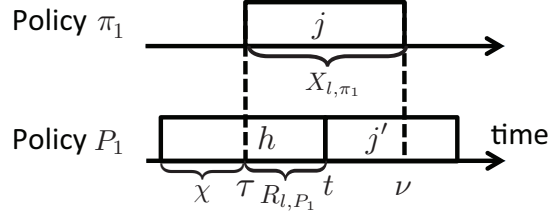
Fig. 5: Illustration of the weak work-efficiency ordering between policy π_1 and policy P_1.

Next, consider the case that two jobs i and j have identical due time $d_i = d_j$. Hence, $(v_i' - v_j')(d_i - d_j) = 0$. In this case, the service order of job i and job j are indeterminate in policy P. Nonetheless, by Lemma 8, the service order of job i and job j does not affect the first inequality in (65). Hence, the first inequality in (65) holds even when $d_i = d_j$.

Finally, (64) follows from (65).

Step 3: We use (64) *to prove Proposition 8.* For any $f \in \mathcal{D}_{\text{Sch-1}}$, $f(\boldsymbol{x} + \boldsymbol{d})$ is increasing and Schur convex. According to Theorem 3.A.8 of [18], for all $f \in \mathcal{D}_{\text{Sch-1}}$, we have

$$
\begin{aligned}
&f(\boldsymbol{v}(P)) \\
=&f[(\boldsymbol{v}(P) - \boldsymbol{d}) + \boldsymbol{d}] \\
\leq&f[(\boldsymbol{c}(\pi) - \boldsymbol{d}) + \boldsymbol{d}] \\
=&f(\boldsymbol{c}(\pi)).
\end{aligned}
$$

This completes the proof. □

APPENDIX H
PROOF OF LEMMA 2

We use coupling to prove Lemma 2: We construct two policies P_1 and π_1 such that policy P_1 satisfies the same queueing discipline with policy P, and policy π_1 satisfies the same queueing discipline with policy π. Hence, policy P_1 is work-conserving. The task and job completion times of policy P_1 (policy π_1) have the same distribution with those of policy P (policy π). Because the state process is determined by the job parameters $\mathcal{I}$ and the task/job completion events, the state process $\{\boldsymbol{\xi}_{P_1}(t), \boldsymbol{\gamma}_{P_1}(t), t \in [0, \infty)\}$ of policy P_1 has the same distribution with the state process $\{\boldsymbol{\xi}_P(t), \boldsymbol{\gamma}_P(t), t \in [0, \infty)\}$ of policy P, and the state process $\{\boldsymbol{\xi}_{\pi_1}(t), \boldsymbol{\gamma}_{\pi_1}(t), t \in [0, \infty)\}$ of policy π_1 has the same distribution with the state process $\{\boldsymbol{\xi}_\pi(t), \boldsymbol{\gamma}_\pi(t), t \in [0, \infty)\}$ of policy π.

Next, we show that policy P_1 and policy π_1 can be constructed such that policy P_1 is weakly more work-efficient than policy π_1 with probability one. Let us consider any task j executed in policy π_1. Suppose that in policy π_1 task j starts service in server l at time τ and completes service at time ν, and the queue is not empty during $[\tau, \nu]$ in policy P_1. Because policy P_1 is work-conserving, all servers are busy in policy P_1 during $[\tau, \nu]$. In particular, server l is busy in policy P_1 during $[\tau, \nu]$. Suppose that in policy P_1, server l has spent a time duration χ ($\chi \geq 0$) on executing a task h before time τ. Let R_{l,P_1} denote the remaining service time of server l for executing task h after time τ in policy P_1. Let $X_{l,\pi_1} = \nu - \tau$ denote the service time of task j in policy π_1 and $X_{l,P_1} = \chi + R_{l,P_1}$ denote the service time of task h in policy P_1. Then, the complementary CDF of R_{l,P_1} is given by

$$\Pr[R_{l,P_1} > s] = \Pr[X_{l,P_1} - \chi > s | X_{l,P_1} > \chi].$$

Because the task service times are NBU, we can obtain that for all $s, \chi \geq 0$

$$\Pr[X_{l,P_1} - \chi > s | X_{l,P_1} > \chi] = \Pr[X_{l,\pi_1} - \chi > s | X_{l,\pi_1} > \chi] \leq \Pr[X_{l,\pi_1} > s],$$

and hence $R_{l,P_1} \leq_{\text{st}} X_{l,\pi_1}$. By Theorem 1.A.1 of [27], the random variables R_{l,P_1} and X_{l,π_1} can be constructed such that $R_{l,P_1} \leq X_{l,\pi_1}$ with probability one. That is, in policy P_1 server l completes executing task h at time $t = \tau + R_{l,P_1}$, which is earlier than time $\nu = \tau + X_{l,\pi_1}$ with probability one. Because in policy P_1 server l is kept busy during $[\tau, \nu]$, a new task, say task j', will start execution on server l at time $t \in [\tau, \nu]$ with probability one.

In the above coupling arguments, conditioned on every possible realization of policy P_1 and policy π_1 before the service of task j starts, we can construct the service of task j in policy π_1 and the service of the corresponding task j' in policy P_1 such that the requirement of weak work-efficiency ordering is satisfied for this pair of tasks. Next, following the proof of [27, Theorem 6.B.3], one can continue this procedure to progressively construct the service of all tasks in policy π_1 and policy P_1. By this, we obtain that policy P_1 is weakly more work-efficient than policy π_1 with probability one, which completes the proof.

APPENDIX I
PROOF OF THEOREM 2

Consider the time difference $C_i(\text{FUT}) - V_i(\text{FUT})$. At time $V_i(\text{FUT})$, all tasks of job i are completed or in service. if $k_i > m$, then job i has at most m incomplete tasks that are in service at time $V_i(\text{FUT})$, see Fig. 2 for an illustration; if $k_i \leq m$, then job i has at most k_i incomplete tasks that are in service at

time $V_i(\text{FUT})$. Therefore, in policy FUT, no more than $k_i \wedge m = \min\{k_i, m\}$ tasks of job i are completed during the time interval $[V_i(\text{FUT}), C_i(\text{FUT})]$.

Suppose that at time $V_i(\text{FUT})$, the remaining tasks of job i are being executed by the set of servers $\mathcal{S}_i \subseteq \{1, \ldots, m\}$, which satisfies $|\mathcal{S}_i| \leq k_i \wedge m$. Let χ_l denote the amount of time that server $l \in \mathcal{S}_i$ has spent on executing a task of job i by time $V_i(\text{FUT})$ in policy FUT. Let R_l denote the remaining service time of server $l \in \mathcal{S}_i$ for executing this task after time $V_i(\text{FUT})$. Then, R_l can be expressed as $R_l = [X_l - \chi_l | X_l > \chi_l]$. Because the X_l's are independent NBU random variables with mean $\mathbb{E}[X_l] = 1/\mu_l$, for all realizations of χ_l

$$[R_l | \chi_l] \leq_{\text{st}} X_l, \ \forall \ l \in \mathcal{S}_i.$$

In addition, Theorem 3.A.55 of [27] tells us that

$$X_l \leq_{\text{icx}} Z_l, \ \forall \ l \in \mathcal{S}_i,$$

where $\leq_{\text{icx}}$ is the increasing convex order defined in [27, Chapter 4] and the Z_l's are independent exponential random variables with mean $\mathbb{E}[Z_l] = \mathbb{E}[X_l] = \mu_l$. Hence,

$$[R_l | \chi_l] \leq_{\text{icx}} Z_l, \ \forall \ l \in \mathcal{S}_i.$$

Lemma 3 tells us that the R_l's are conditional independent for any given realization of $\{\chi_l, l \in \mathcal{S}_i\}$. Hence, by Corollary 4.A.16 of [27], for all realizations of $\mathcal{S}_i$ and $\{\chi_l, l \in \mathcal{S}_i\}$

$$\left[\max_{l \in \mathcal{S}_i} R_l \big| \mathcal{S}_i, \{\chi_l, l \in \mathcal{S}_i\}\right] \leq_{\text{icx}} \left[\max_{l \in \mathcal{S}_i} Z_l \big| \mathcal{S}_i\right]. \tag{70}$$

Then,

$$\mathbb{E}[C_i(\text{FUT}) - V_i(\text{FUT}) | \mathcal{S}_i, \{\chi_l, l \in \mathcal{S}_i\}]$$

$$= \mathbb{E}\left[\max_{l \in \mathcal{S}_i} R_l \bigg| \mathcal{S}_i, \{\chi_l, l \in \mathcal{S}_i\}\right]$$

$$\leq \mathbb{E}\left[\max_{l \in \mathcal{S}_i} Z_l \bigg| \mathcal{S}_i\right] \tag{71}$$

$$\leq \mathbb{E}\left[\max_{l=1, \ldots, k_i \wedge m} Z_l\right] \tag{72}$$

$$\leq \sum_{l=1}^{k_i \wedge m} \frac{1}{\sum_{j=1}^{l} \mu_j}, \tag{73}$$

where (71) is due to (70) and Eq. (4.A.1) of [27], (72) is due to $\mu_1 \leq \ldots \leq \mu_M$, $|\mathcal{S}_i| \leq k_i \wedge m$, and the fact that $\max_{l=1, \ldots, k_i \wedge m} Z_l$ is independent of $\mathcal{S}_i$, and (73) is due to the property of exponential distributions.

Because $\mathcal{S}_i$ and $\{\chi_l, l \in \mathcal{S}_i\}$ are random variables which are determined by the job parameters $\mathcal{I}$, taking the conditional expectation for given $\mathcal{I}$ in (73), yields

$$\mathbb{E}[C_i(\text{FUT}) - V_i(\text{FUT})|\mathcal{I}] \leq \sum_{l=1}^{k_i \wedge m} \frac{1}{\sum_{j=1}^{l} \mu_j}.$$

By taking the average over all n jobs, (18) is proven. In addition, it is known that for each $k = 1, 2, \ldots,$

$$\sum_{l=1}^{k} \frac{1}{l} \leq \ln(k) + 1.$$

By this, (19) holds. This completes the proof.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *USENIX OSDI*, Dec. 2004, pp. 137–150.

[2] L. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, pp. 687–690, 1968.

[3] D. R. Smith, "A new proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, pp. 197–199, 1978.

[4] J. R. Jackson, "Scheduling a production line to minimize maximum tardiness," management Science Research Report, University of California, Los Angeles, CA, 1955.

[5] F. Baccelli, Z. Liu, and D. Towsley, "Extremal scheduling of parallel processing with and without real-time constraints," *J. ACM*, vol. 40, no. 5, pp. 1209–1237, Nov. 1993.

[6] W. E. Smith, "Various optimizers for single-stage production," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 59–66, 1956.

[7] M. H. Rothkopf, "Scheduling with random service times," *Management Science*, vol. 12, no. 9, pp. 707–713, 1966.

[8] J. C. Gittins, "Bandit processes and dynamic allocation indices," *Journal of the Royal Statistical Society, Series B*, pp. 148–177, 1979.

[9] G. P. Klimov, "Time-sharing service systems. I," *Theory of Probability & Its Applications*, vol. 19, no. 3, pp. 532–551, 1975.

[10] J. Gittins, K. Glazebrook, and R. Weber, *Multi-Armed Bandit Allocation Indices*, 2nd ed. Wiley, 2011.

[11] S. Leonardi and D. Raz, "Approximating total flow time on parallel machines," in *ACM STOC*, 1997.

[12] G. Weiss, "Turnpike optimality of smith's rule in parallel machines stochastic scheduling," *Math. Oper. Res.*, vol. 17, no. 2, pp. 255–270, May 1992.

[13] ——, "On almost optimal priority rules for preemptive scheduling of stochastic jobs on parallel machines," *Advances in Applied Probability*, vol. 27, no. 3, pp. 821–839, 1995.

[14] L. Ying, R. Srikant, and X. Kang, "The power of slightly more than one sample in randomized load balancing," in *2015 IEEE INFOCOM*, April 2015, pp. 1131–1139.

[15] M. Dacre, K. Glazebrook, and J. Nio-Mora, "The achievable region approach to the optimal control of stochastic systems," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 4, pp. 747–791, 1999.

[16] A. L. Stolyar, "Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic," *The Annals of Applied Probability*, vol. 14, no. 1, pp. 1–53, 2004.

[17] Z. Liu, P. Nain, and D. Towsley, "Sample path methods in the control of queues," *Queueing Systems*, vol. 21, no. 3, pp. 293–335.

[18] A. W. Marshall, I. Olkin, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications*, 2nd ed. Springer, 2011.

[19] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: Distributed, low latency scheduling," in *ACM SOSP*, 2013, pp. 69–84.

[20] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in *EuroSys*, Bordeaux, France, 2015.

[21] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 4th ed. Springer, 2012.

[22] L. Kleinrock, *Queueing Systems*. John Wiley and Sons, 1975, vol. 1& 2.

[23] J. Nino-Mora, "Conservation laws and related applications," in *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[24] J. C. Gittins, K. Glazebrook, and R. Weber, *Multi-armed Bandit Allocation Indices*, 2nd ed. Wiley, Chichester, NY, 2011.

[25] Y. Sun, C. E. Koksal, and N. B. Shroff, "On delay-optimal scheduling in queueing systems with replications." [Online]. Available: http://arxiv.org/abs/1603.07322

[26] C.-S. Chang and D. D. Yao, "Rearrangement, majorization and stochastic scheduling," *Math. of Oper. Res*, 1993.

[27] M. Shaked and J. G. Shanthikumar, *Stochastic Orders*. Springer, 2007.