Submitted to *Production and Operations Management*
manuscript POM-Jun-15-OA-0438.R2

Priority Rules for Multi-Task Due-Date Scheduling Under Varying Processing Costs

Yunjian Xu
Engineering Systems & Design, Singapore University of Technology and Design, Singapore, yunjian_xu@sutd.edu.sg

Cong Shi
Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109, shicong@umich.edu

Izak Duenyas
Technology and Operations, Ross School of Business, University of Michigan, Ann Arbor, MI 48109, duenyas@umich.edu

We study the scheduling of multiple tasks under varying processing costs and derive a priority rule for optimal scheduling policies. Each task has a due date, and a non-completion penalty cost is incurred if the task is not completely processed before its due date. We assume that the task arrival process is stochastic and the processing rate is capacitated. Our work is motivated by both traditional and emerging application domains, such as construction industry and freelance consulting industry. We establish the optimality of *Shorter Slack time and Longer remaining Processing time (SSLP) principle* that determines the priority among active tasks. Based on the derived structural properties, we also propose an effective cost-balancing heuristic policy and demonstrate the efficacy of the proposed policy through extensive numerical experiments. We believe our results provide operators/managers valuable insights on how to devise effective service scheduling policies under varying costs.

Key words: multi-task due-date scheduling, varying processing costs, make-to-order, priority rule
History: Received May 2015; revisions received December 2015, February 2016; accepted March 2016 by Michael Pinedo after two revisions.

1. Introduction

We study the scheduling of multiple processors to perform multiple tasks under varying (and possibly random) processing costs. The task arrival processes can be random and intertemporally correlated. Each task can be processed at a limited rate before its due date (which is sometimes referred to as deadline in the literature). Failure to meet the due dates will result in non-completion penalty costs. The critical feature of our model is that the processing cost can also be random and nonstationary. In many practical settings, the processing cost of each task may fluctuate over time due to varying labor, energy or raw material costs. The objective is to find a feasible scheduling policy to process incoming tasks so as to minimize the total expected costs, i.e., the sum of the expected processing cost and the expected non-completion penalty for not finishing tasks before their pre-specified due dates. The main focus of this paper is to derive priority rules for optimal scheduling policies and also to develop effective heuristic policies based on these rules.

Our work is primarily motivated by the scheduling problem faced by subcontractors in the construction industry. There are many firms that specialize on one aspect of construction (e.g., cabinet or garage installation, kitchen remodeling, etc.) that are used by the contractor who is responsible for the whole construction project on an as-needed basis. Typically, the subcontractors will get requests that have strict due dates (with penalties for non-completion), and they need to decide how to schedule their work force across the multiple projects that are waiting to be completed before their due dates. The subcontractor has access to a labor pool that has varying availability (i.e., the number of total workers available in any given day varies) and also different pay rates depending on the day that the work has to be done (e.g., to complete the project faster the manager can get the workers to work on a Sunday at a much higher labor cost). Depending on the nature of the job, energy and materials costs can significantly vary across time as well. Thus, the scheduler has to consider all jobs competing for his available capacity and decide how to schedule his work force across the different jobs. This paper focuses on the case in which the subcontractor cannot allocate parallel capacity to a job on the same period to speed up the job. For example, installing a garage door cannot be significantly speeded up if the scheduler sends multiple teams instead of a single team to install the garage door. Similarly, installing kitchen cabinets usually involves one installation team and sending two teams to a house does not speed up the work (except in the rare house that has multiple kitchens). This is in contrast to other activities like painting or flooring where sending more people to one location can accelerate the job.

A similar problem arises in the freelance consulting industry (e.g., HourlyNerd). Unlike the regular consulting companies, which hire employees for the whole year and allocate them to client projects, other consulting companies hire freelance consultants only on an as-needed basis and allocate them to individual clients. The problem is that the availability of the number of freelance consultants available to the firm at any point in time varies, resulting in the firm having to pay higher amounts in certain periods to secure consulting capacity to serve their clients' needs. For example, in the metro Detroit area, there are several firms that specialize on lean consulting. Most of these firms do not have the lean consultants on staff but only contact them on an as-needed basis. Often the lean consultants are affiliated with these companies but also have other work that they do on their own. Depending on the time of year, and industry demand trends, the daily rate that the firm has to pay for a consultant's services to allocate them to a given client can vary significantly (one of the co-authors of this paper has consulted himself and charged highly varying rates to these firms depending on his level of availability).

The difficult question that the firms face is to decide which job(s) should have teams allocated to them on any given day. A common rule of thumb is to allocate teams according to the earliest due date first rule. In this paper, we show that the earliest due date first rule is not necessarily optimal when the firm faces varying costs over time. Instead, we develop an alternative priority rule and establish its optimality.

1.1. Main results and contributions of this paper

The major results and contributions of this paper are summarized as follows.

We study the scheduling of multiple tasks under demand and cost uncertainty in order to minimize the expected total cost (the sum of the processing cost and non-completion penalty). (A non-completion penalty cost will be incurred if a task is not completely processed at its due date.) We derive a priority rule for optimal scheduling policies that provides managers valuable insight into designing effective heuristic policies. We demonstrate (via Example 1) that giving priority to a task with the earliest due date could be sub-optimal, even if it is feasible to complete all tasks before their due dates. We further argue that even earliest due date together with shortest slack time does not guarantee the priority of a task (cf. Example 2). Here, the *slack time* (or the laxity interchangeably) of a task is defined as the difference between the remaining time before its due date and its remaining processing time. Somewhat surprisingly, under a mild assumption that each task's non-completion penalty cost is convex in its remaining processing time at its due date, we establish the optimality of *Shorter Slack time and Longer remaining Processing time (SSLP) principle*: priority should be given to a task with shorter slack time and longer remaining processing time, regardless of future system dynamics, e.g., the dynamic processes that describe future task arrivals and time-varying processing cost.

It is worth noting that if a task j has shorter (or the same) slack time and a later due date than another task i, then task j must have longer remaining processing time than i, and according to the SSLP principle task j has the priority. Among two tasks with the same slack time, our result suggests that priority should be given to the task with *later due date*. The optimality of the Latest Due date First (LDF) principle among jobs with the same slack time is in sharp contrast to the common sense that the Earliest Due date First (EDF) policy is optimal. Although the LDF principle has been mentioned in the literature (Chen and Yih (1996), Farzan and Ghodsi (2002)), the authors are not aware of any formal results that establish the optimality of LDF. We believe that our results provide managers valuable insights on how to devise effective service scheduling policies under demand and cost uncertainty.

The intuition behind the SSLP principle is conceptually simple. Consider two tasks with the same slack time and the same non-completion penalty. Priority should be given to the task with longer remaining processing time (and a later due date), which leads to a larger number of small unfinished tasks in the future (cf. the discussion following Example 1). Facing time-varying (and possibly stochastic) processing costs and processing rate constraint (i.e., each task can be processed only by a limited amount in each time period), the operator/manager prefers to have many small unfinished tasks that could be simultaneously processed when the processing cost becomes lower. In other words, the SSLP principle enlarges the admissible set of actions in future periods, and therefore reduces the long-term expected cost by enabling the operator/manager to better explore cheaper resources that may become available in the future.

We note that computing exact optimal policies by brute-force dynamic programming is intractable, since the number of system states grows exponentially with the number of tasks. Based on the structural properties of optimal policies derived above, we devise an effective cost-balancing heuristic policy, and our numerical results show that the proposed policy consistently outperforms existing earliest-due-date-first-based (EDF) greedy policies for a large set of demand and parameter instances. We observe that the cost improvement of our SSLP-based policy over the benchmark EDF-based policy is most significant when the processing cost has a stationary or decreasing trend. This is mainly because in such scenarios our SSLP-based policy tends to save many small unfinished jobs (with shorter remaining process time) that could be simultaneously processed later with lower cost, which is in line with our theoretical results.

1.2. Literature review

Our work is closely related to two streams of on-going research, namely, production/service scheduling and stochastic MTO systems.

Production or service scheduling. Our work is intimately related to the large body of literature on the scheduling of multiple jobs in production/service systems with tardiness penalties (see, e.g., Baker and Scudder (1990), Cheng and Gupta (1989), Keskinocak and Tayur (2004), Bülbül et al. (2007)). Closer to the present work, a few recent works study the scheduling in make-to-order environments with order due dates (see, e.g., Pundoor and Chen (2005), Leung et al. (2006), Chen (2010), Leung and Chen (2013)). We note that this literature usually considers the scheduling of a fixed set of orders, while our model allows for random order arrival processes. While the aforementioned literature usually adopts mathematical programming based approaches to study the scheduling of make-to-order systems, to deal with random order arrivals we resort to *dynamic programming* which is a standard approach for sequential decision problems under uncertainty.

There also exists a substantial body of literature on due date scheduling models that incorporates random order arrival processes in the context of real-time dynamic systems and queuing systems. For single-processor periodic task scheduling systems, it is well-known that simple scheduling algorithms, such as the earliest due date first (EDF) policies (see, e.g., Liu and Layland (1973)) and the least-laxity-first (LLF) policies (see, e.g., Dertouzos (1974)) are optimal, in the sense that both the EDF and LLF policies must be able to complete all tasks before their due dates, provided that completion of all tasks before due dates is feasible. For a variety of single-server queuing systems, Panwar et al. (1988) showed that a variation of the EDF policy maximizes the fraction of customers served within their respective due dates, and Pinedo (1983) characterized simple policies that minimize the expected weighted number of late jobs. We note, however, that when the completion of all tasks is not feasible, EDF and LLF policies may perform poorly (see Locke (1986)). Indeed, in this "overload" setting there does not exist an optimal "on-line" policy which makes scheduling decision based only on the states of active tasks that have arrived (see G. Buttazzo (1995)). There is also a literature on due date scheduling of multiple processors (see, e.g., Davis and Burns (2011) for a comprehensive survey). To our knowledge, no characterization on optimal scheduling policies is provided for the multi-processor setting considered in this paper.

Whereas in the aforementioned literature the processing cost is usually assumed to be constant over the entire operation interval, the key distinction in our setting is that the processing cost is allowed to be time-variant and random, which is a more general and realistic assumption. We show by example that under time-variant processing cost and processing capacity constraint, EDF policies could be highly sub-optimal. Somewhat surprisingly, we further establish the optimality of a priority rule (the SSLP principle), which sometimes gives priority to tasks with *later due date*, for example, among a set of tasks with the same slack time. To our knowledge, the present work is the first that rigorously investigates priority rules for due-date scheduling of multiple processors under varying processing costs and random task arrivals.

Stochastic MTO systems. Dellaert and Melo (1995) gave heuristic procedures for a stochastic lot-sizing problem in a MTO manufacturing system. Duenyas and Van Oyen (1996) designed a heuristic scheduling policy for a MTO system with heterogeneous classes of customers in a queueing context. He et al. (2002) studied a MTO inventory production system consisting of a warehouse and a workshop and also discussed the value of information in this simple supply chain. More recently, Buchbinder et al. (2013) studied an online MTO variant of the classical joint replenishment problem. There is also a stream of research devoted to the study of joint MTS/MTO systems (see, e.g., Carr and Duenyas (2000), Gupta and Wang (2007), Dobson and Yano (2002), Youssef et al. (2004), Iravani et al. (2012)). The aforementioned literature mainly deals with procurement, job assignment or production-line assignment decisions in order to minimize the waiting (penalty) costs. The present work, on the other hand, deals with a due date scheduling problem whose state space is much larger (since for each active task the manager has to keep track the start time, the end time, the required processing time, and how much it has been processed). The resulting dynamic program is much more complicated to analyze. We contribute to the existing literature by prescribing a set of optimal priority rules for such complex due-date-based MTO systems under varying costs.

There is another major stream of scheduling literature on MTO systems that falls outside the scope of this paper, which focuses on the impact of lead time and due date quotation. Kaminsky and Kaya (2008) considered a scheduling model where a manufacturer in a MTO production environment, is penalized for long lead time and for missing due dates. Afèche et al. (2013) studied revenue-maximizing tariffs that depend on realized lead times where a provider is serving multiple time-sensitive types of customers. Zhao et al. (2012) studied and compared the profits between uniform quotation model and differentiated quotation model in MTO manufacturing industries. Benjaafar et al. (2011) considered a supplier with finite production capacity and stochastic production times, where customers provide advance demand information (ADI) to the supplier by announcing orders ahead of their due dates. In this stream of research, the authors are interested in the due date quoting, whereas we take the due dates of tasks as given and are interested in the optimal sequencing of these tasks.

1.3. Structure of this paper

The remainder of the paper is organized as follows. In §2, we present the mathematical formulation of our service scheduling problem. In §3, we characterize some important structural properties of optimal policies under our capacity settings. In §4, we propose an effective heuristic policy and also carry out an extensive numerical study. In §5, we establish the SSLP principle in a more general setting with heterogeneous non-completion penalty, and show the optimality of EDF policies in an alternative setting without processing rate constraint. Finally, we conclude our paper and point out some plausible future research avenues in §6.

2. Due-Date Scheduling with Random Processing Cost

We study the scheduling of multiple due-date-constrained tasks in a stochastic environment with random arrival, stochastic processing cost, and limited processing rate. Our model focuses on a capacitated MTO system, where the production/service only starts after orders are placed by customers. We consider a periodic-review system over a planning horizon of $T+1$ periods (possibly infinite), where time periods are indexed by $t=0,1,\ldots,T$. There is a capacity constraint N_t on the total amount of service in each period t.

Symbol	Type	Description
f_t	State	The information set that is available at the beginning of period t.
F_t	Parameter	The finite set of all possible f_t.
I_t	State	The set of active tasks available for processing in period t.
a^i	State	The arrival time of an active task $i \in I_t$.
b^i	State	The due date of an active task $i \in I_t$.
m^i	State	The required processing time to complete an active task $i \in I_t$.
n_t^i	State	The time already spent processing an active task $i \in I_t$.
x_t^i	State	In period t, the remaining time before task i's due date, $b^i - t$.
y_t^i	State	In period t, the remaining processing time of task i, $m^i - n_t^i$.
z_t^i	State	The state of an active task $i \in I_t$: $z_t^i = (x_t^i, y_t^i)$.
$\mathbf{z}_t$	State	The state of all active tasks in period t: $\mathbf{z}_t = \{z_t^i : i \in I_t\}$.
s_t^i	State	The slack of an active task $i \in I_t$, $x_t^i - y_t^i$.
u_t^i	Decision	The processing amount allocated to an active task $i \in I_t$ in period t.
$\mathbf{u}_t$	Decision	The action vector taken in period t: $\mathbf{u}_t = \{u_t^i : i \in I_t\}$.
$U(\mathbf{z}_t, f_t)$	State	The set of admissible action vectors in period t.
D_t	State	The set of newly arrival tasks at the beginning of period t.
W_t	State	The external information available at the beginning of period t.
J_t^1	State	The set of tasks that are successfully completed in period t.
J_t^2	State	The set of tasks that are not successfully completed in their due date t.
N_t	Parameter	The total processing capacity available in period t.
c^i	Parameter	The per-unit processing cost of an active task $i \in I_t$.
$\epsilon(f_t)$	Parameter	The (random) perturbation on tasks' per-unit processing cost under f_t.
$q(\cdot)$	Parameter	The mapping from the remaining processing time of a task (at its due date) to its non-completion penalty.

Table 1 Summary of Major Notation

Demand structure. The demand sets, denoted by $D_0, \ldots, D_T$, are random, where the demand D_t is the *set* of all new tasks available to be processed in period t (e.g., the newly requested installation/remodeling projects in period t). More specifically, for each new task $i \in D_t$, we use a^i to denote its arrival time (We set a^i to be t for each new arriving task i in period t), $b^i \le T+1$ to denote its due date, and m^i to denote its required processing time to complete the task. We assume that the task i can be processed during any periods within $[a^i, b^i)$. We use n_t^i to track the time already spent processing the existing unfinished task i between its arrival a^i and time t. (Note that if i is a new task arriving in period t, then $n_t^i = 0$.)

As part of the model, we assume that at the beginning of each period t, we are given what we call an *information set* that is denoted by f_t. The information set f_t contains all the information that is available at the beginning of period t. More specifically, the information set f_t contains the realized demand sets $(D_0, \ldots, D_t)$, and possibly some more (external) information denoted by $(W_0, \ldots, W_t)$ (e.g., the state of economy, energy and raw material costs) that may have influence on demand and cost. The information set f_t in each period t belongs to a finite set consisting of all possible information sets F_t. In addition, we assume that in each period t there is a known conditional joint distribution of the future demands $(D_{t+1}, \ldots, D_T)$, which is determined by f_t. This demand model is very general, allowing for nonstationarity and correlation between the demands of different periods. In particular, it includes not only independent demand processes, but also most time-series demand models such as autoregressive (AR) and autoregressive moving average (ARMA) demand models, demand forecast updating models such as Martingale models for forecast evolution (MMFE) (e.g., Heath and Jackson (1994)), demand processes with advance demand information (ADI) (e.g., Gallego and Özer (2001)), as well as economic-state driven demand processes such as Markov modulated demand processes with state transition matrix.

State of the system. In each period t, let I_t be the set of *active* tasks available for processing. The state of the system consists of the information set f_t and the states of all active tasks. It is clear that the state of the active tasks in period t can by fully described by a set of quadruplets $\{(a^i, b^i, m^i, n_t^i) : i \in I_t\}$. However, this state variable is not convenient to work with. It turns out that we can keep track of the *effective* state by using only two variables. Now, for each task $i \in I_t$, we introduce $x_t^i \triangleq b^i - t$ as the *remaining time before its due date*, where $\triangleq$ means "defined as". We also introduce $y_t^i \triangleq m^i - n_t^i$ as the *remaining processing time* required to complete the task. The state of the active tasks in period t is therefore given by

$$\mathbf{z}_t \triangleq \left\{ z_t^i \triangleq (x_t^i, y_t^i) : i \in I_t \right\}.$$

In period t, for every task $i \in I_t$, we define its slack time $s_t^i \triangleq x_t^i - y_t^i$, which is the difference between the remaining time before the due date and the remaining time required to complete the task. A task's slack time is the maximum allowance time for the system to stay idle if the task has to be eventually processed before its due date.

System dynamics. We now proceed to describe the dynamics of our model. At the beginning of period t, the information set f_t (containing the new task arrivals D_t) is revealed to the manager.

Let $\tilde{I}_t$ be the set of active tasks available for processing before the demands arrive in period t. (We assume that $\tilde{I}_0 = 0$.) Then the manager updates the state of active tasks to be $I_t = \tilde{I}_t \cup D_t$. Next, the manager decides the processing amount allocated to each active task $i \in I_t$, denoted by the decision variable u_t^i. We define $\mathbf{u}_t \triangleq \{u_t^i : i \in I_t\}$ as our actions or decisions, and $U_t(\mathbf{z}_t, f_t)$ as the admissible set of actions depending on the capacity scenario (defined later in this section). The distribution of f_{t+1} (over the set F_{t+1}) is determined by the current information set f_t and does not depend on the action taken in period t, $\mathbf{u}_t$.

It follows that $y_{t+1}^i = y_t^i - u_t^i$, i.e., the remaining process time for task i in the next period $t+1$ is reduced by how much we process task i in period t, u_t^i. Also, it is clear that $x_{t+1}^i = x_t^i - 1$, i.e., the remaining time before its due date reduces by 1 period (independent of our action). We remove the task i from the active set I_t at the end of period t if $x_{t+1}^i = 0$ (in which we run out of time) or $y_{t+1}^i = 0$ (in which the task has been completely processed).

Let J_t^1 be the set of tasks successfully completed in period t, i.e., $J_t^1 \triangleq \{j \in I_t : y_{t+1}^j = 0\}$. Let J_t^2 be the set of tasks that we are unable to fully process but have already reached their due dates in period t, i.e., $J_t^2 \triangleq \{j \in I_t : x_{t+1}^j = 0 \text{ and } y_{t+1}^j > 0\}$. Then the state transition is $\tilde{I}_{t+1} = I_t \setminus J_t^1 \setminus J_t^2$.

Capacity constraints. Since the production or processing rate is limited (e.g., installation or remodeling projects), we assume that for each task, zero or one unit of a task can be processed in each time period. Furthermore, the total processing capacity assigned to all active tasks in period t is upper bounded by N_t. Formally, at a state $\mathbf{z}_t$, the feasible action space is defined as

$$U_t(\mathbf{z}_t, f_t) \triangleq \left\{ \mathbf{u}_t : u_t^i \in \{0,1\} \text{ for all } i \in I_t \text{ and } \sum_{i \in I_t} u_t^i \le N_t \right\}.$$

We also consider the uncapacitated counterpart model in §5.2.

Cost structure. For each decision u_t^i, we incur a linear processing cost $(c^i + \epsilon(f_t))u_t^i$, where c^i is the per-unit processing cost of task i and $\epsilon(f_t)$ is a perturbation term depending on the information set f_t. For simplicity, we consider aggregate (non-idiosyncratic) random shock $\epsilon(f_t)$ to reflect the macroeconomic impact of cost volatility in a given application (e.g., the state of economy, energy or raw material costs in the installation or remodeling example). We note that no assumption is needed on the distribution of the aggregate random shock, i.e., the mapping $\epsilon(\cdot)$ is allowed to be arbitrary. In practice, the manager can use past observable costs (from markets or firms' own database) or forecasted costs (from markets or forecasting agencies) to determine an appropriate choice of $\epsilon(\cdot)$. For example, if the data suggest that the costs are independent, one may use Normal or other general-purpose distributions such as Weibull to fit the random error distribution. If the data suggest a clear trend, one may use linear regression model to determine the mapping $\epsilon(\cdot)$ assuming normality of the random error term (see Draper and Smith (1998)).

The total processing cost in period t is given by

$$\sum_{i \in I_t} (c^i + \epsilon(f_t)) u_t^i. \tag{1}$$

For each task $j \in J_t^2$ that is not completed before its due date, the firm incurs a nonnegative penalty cost that is a function of its remaining processing time, denoted by $q(y_t^j - u_t^j)$, with $q(0) = 0$. The function $q(\cdot)$ maps the remaining processing time of each task to its non-completion penalty. In our setting, if a particular task is not completed by the due date, the firm may face severe penalties, including loss of profit and goodwill, or customers compensation expenses for the inconvenience. In practice, the remaining process time at due date implies an extra delay is required after the due date. The firm often needs to request the workers to work overtime, or resort to hiring additional subcontractors from other firms to wrap up the task at more expensive rates. Moreover, the more incomplete the task remains at its due date, the heavier penalty the firm has to pay to compensate customers for the inconvenience. For example, if a housing project is 98% complete (only requiring some additional finishing touches), the residents may generally move in and request no compensation. However, if it is only 80% complete and the residents are unable to move in, they may need to stay in a hotel for an extended period of time and the firm may have to pay all the hotel expenses. In such scenarios, as the percentage completion decreases, the service provider tends to incur higher costs. Thus we model the penalty associated with the non-completion (of task j), $q(y_t^j - u_t^j)$, to be a general convex (including linear as a special case) function of its remaining processing time. We note that this general convex tardiness cost assumption has been used in the scheduling literature (see Federgruen and Mosheiov (1997) and the references therein). The total non-completion penalty cost incurred in period t is then given by $\sum_{j \in J_t^2} q(y_t^j - u_t^j)$.

Combining the above two types of cost, the total cost associated with period t is therefore

$$C_t(\mathbf{z}_t, f_t, \mathbf{u}_t) \triangleq \sum_{i \in I_t} (c^i + \epsilon(f_t)) u_t^i + \sum_{j \in J_t^2} q(y_t^j - u_t^j). \tag{2}$$

We remark that one may also consider the idle costs of processors and holding costs of completed tasks. However, to keep the model succinct and bring out the important insights, we decide not to model them in our setting, since these costs tend to be much smaller than the processing and non-completion penalty costs. For example, the idling machines do not consume electricity in a construction project (so the idle costs are negligible). Also, there is typically no penalty for finishing a construction project earlier (so the holding costs of completed tasks are also negligible).

Objective. Our objective is to find a feasible scheduling policy $\pi = (\mathbf{u}_0, \mathbf{u}_1, \ldots)$ to minimize the average per-period cost for this MTO system. Given any initial states $(\mathbf{z}_0, f_0)$, the expected total cost over interval $[t, T]$ induced by a policy π is given by

$$V_t^{\pi}(\mathbf{z}_t, f_t) = \mathbb{E}\left\{\sum_{k=t}^{T} C_t(\mathbf{z}_k, f_k, \mathbf{u}_k)\right\},$$

where the expectation is taken over the distribution of $(f_{t+1}, \ldots, f_T)$, conditioned on the current information set f_t. Our objective is to minimize the expected total cost over the finite planning horizon $[0, T]$.

Dynamic Programming Formulation. The dynamic programming formulation is

$$V_t^{\pi}(\mathbf{z}_t, f_t) = \min_{\mathbf{u}_t \in U_t(\mathbf{z}_t, f_t)} \left\{\mathbb{E}\left[C_t(\mathbf{z}_t, f_t, \mathbf{u}_t)\right] + \mathbb{E}\left[V_{t+1}^{\pi}(\mathbf{z}_{t+1}, f_{t+1}) \mid f_t\right]\right\}, \tag{3}$$

with boundary conditions $V_{T+1}^{\pi}(\cdot,\cdot) = 0$. In Eq. (3), the expectation is over f_{t+1} (the information set in period $t+1$), conditioned on the current information set f_t. In every period t, the system state of the dynamic program consists of the information set f_t and the state of all active tasks $\mathbf{z}_t$. Note that the size of state space grows exponentially with the number of active tasks. Even for a simple case where $|F_t| = 1$ for every t (i.e., there is no randomness in the demand and cost processes), the processing time of each task is less than 4 periods, and each task has to be completed within 8 periods, the size of the system state space scales with $32^{\max_t |I_t|}$, where $|I_t|$ is the number of active tasks in period t. Reasonable numbers of active tasks lead to very high dimensions, which makes brute-force dynamic programming computationally intractable.

3. Priority Rules for Optimal Scheduling Policies

In this section, we study the priority rule among active tasks and derive the main result of this paper, the SSLP principle. First, we provide a simple counter-example to show that the EDF (earliest due date first) scheduling may be suboptimal.

Example 1 (A simple counter-example). Consider the following problem with capacities $N_0 = 1$ and $N_t = 2$ for all $t \geq 1$. In period 0, suppose that there are two tasks in the set I_0 and that no task will arrive in the future. The states of the two tasks are $z_0^1 = (2, 1)$ and $z_0^2 = (3, 2)$. That is, task 1 can be processed in the two periods 0 and 1, and requires to be processed for one time period, while task 2 has a due date at the beginning of period 3, and requires to be processed for two time periods. The two tasks have the same slack time, and task 2 has longer remaining processing time.

We assume that the per-unit processing cost is uniformly 1 for all tasks in period 0, is zero in period 1, and is uniformly 2 in period 2. We also assume that the per-unit non-completion penalty is much higher than the highest per-unit processing cost 2, and therefore it is optimal to finish all tasks before their due dates.

Since the processing cost in period 0 is higher than that in period 1, and is lower than that in period 2, it is straightforward to check that under the unique optimal policy, task 2 is processed in period 0, and both tasks are processed in period 1. This optimal policy yields a total processing cost of 1 and no non-completion penalty. We note that this optimal policy is strictly better than any EDF policy that gives priority to task 1 in period 0. This is because under EDF, task 1 is finished at the end of period 0, which makes it impossible to utilize the free resource in period 1 to process task 1. □

In the above counter-example, priority should be given to task 2 with longer remaining processing time. This is because processing task 1 in period 0 restricts the admissible set of actions in period 1: the operator could process at most one task in period 1 if task 1 were processed in period 0. Facing time-varying (and possibly stochastic) processing costs, the manager prefers to have a larger number of small unfinished tasks that could be simultaneously processed when the processing cost becomes lower.

The above example motivates us to consider a different policy based on *slack time* and *remaining processing time*. Recall that in period t, for every task $i \in I_t$, its slack time is defined by $s_t^i \triangleq x_t^i - y_t^i$, which is the maximum allowance time for the system to stay idle if the task has to be eventually processed.

Definition 1. In period t, for two tasks i and j in the set I_t, we say $i \preccurlyeq j$ (task j has priority to task i) if $s_t^i \geq s_t^j$, $y_t^i \leq y_t^j$ and at least one of these inequalities strictly holds. If tasks $i \preccurlyeq j$ or $j \preccurlyeq i$, we call the tasks i and j *comparable*. □

It is clear that the relation $\preccurlyeq$ is reflexive, antisymmetric, and transitive, and therefore is a partial order (among the states of all tasks in the set I_t). It is worth noting that if task j has shorter slack time and a later due date than task i, i.e., if $s_t^i \geq s_t^j$ and $b_t^i \leq b_t^j$, then task j must have longer remaining processing time than i, and therefore $i \preccurlyeq j$. For two tasks i and j that are comparable (according to Definition 1), priority should always be given to task j, regardless of future demand

and cost processes. This result is referred to as the Shorter Slack Time and Longer Remaining Processing Time (SSLP) principle, which will be formulated and proved later in this section.

In period t with task state $\mathbf{z}_t$, there are only two cases where the two tasks i and j are *incomparable*, namely, (a) $s_t^i \geq s_t^j$ and $y_t^i > y_t^j$; (b) $s_t^i > s_t^j$, $y_t^i \geq y_t^j$. In this case, which task should have a higher priority is a decision that depends on future system dynamics. The following example demonstrates that shorter slack time together with earlier due date does not guarantee priority along every sample path.

Example 2. In period t, suppose that there are two tasks in the set I_t, and that no task will arrive in the future. The states of the two tasks are $z_t^1 = (2,1)$ and $z_t^2 = (4,2)$. The total available capacity in each period is given by: $N_t = N_{t+2} = N_{t+3} = 1$ and $N_{t+1} = 2$. They are incomparable: task 1 has shorter slack time and shorter remaining processing time (and of course, an earlier due date).

The per unit processing cost in period t is zero. Consider two different scenarios after period t. First, suppose that the per unit processing cost is zero in periods $t+2$ and $t+3$, but is $q(2)$ (the non-completion penalty incurred to task 2 if it were not processed at all) in period $t+1$. As a result, it is optimal to process task 1 in period t, and process task 2 in periods $t+2$ and $t+3$. On the other hand, if the per unit processing cost is zero in period $t+1$, but is $q(2)$ in periods $t+2$ and $t+3$, then the operator should process task 2 in period t, and process both tasks in period $t+1$. □

We are now in a position to formulate the Shorter Slack time and Longer remaining Processing Time (SSLP) principle. For any given feasible policy π that violates the SSLP principle, we will define an *interchanging policy* $\tilde{\pi}$ that gives priority to a task with shorter slack time and longer remaining processing time, and then show that the SSLP-based interchanging policy $\tilde{\pi}$ is feasible and can improve the expected total cost compared to original feasible policy.

Definition 2 (An SSLP-based Interchanging Policy). Assume that in period τ with active tasks $\mathbf{z}_\tau$, task j has priority to i (i.e., $i \preccurlyeq j$), and that a policy $\pi = \{\mathbf{u}_\tau, \mathbf{u}_{\tau+1}, \ldots, \mathbf{u}_T\}$ processes task i but not j. An interchanging policy $\tilde{\pi} = \{\tilde{\mathbf{u}}_\tau, \tilde{\mathbf{u}}_{\tau+1}, \ldots, \tilde{\mathbf{u}}_T\}$ (generated from policy π with respect to tasks i and j at state $\mathbf{z}_\tau$) processes the same tasks as the original policy π (i.e., copies π), except that:

1. In period τ, the interchanging policy $\tilde{\pi}$ processes task j but not i at task state $\mathbf{z}_\tau$.
2. If there exists a period $t \in (\tau, b_\tau^i)$ such that the original policy π processes task j but not i, then the interchanging policy $\tilde{\pi}$ processes task i but not j. This case results in a complete swap of tasks i and j between π and $\tilde{\pi}$.

3. If there does not exist a period $t \in (\tau, b^i_\tau)$ such that the original policy π processes task j but not i, then it implies that for each period $t \in (\tau, b^i_\tau)$, the original policy π always processes task i whenever it processes j. In this case, the interchanging policy $\tilde{\pi}$ will take the same action as the original policy π after period τ. $\square$

Two simple (illustrative) scenarios on the interchanging policy defined above are given in Table 2 (corresponding to item 2 above) and Table 3 (corresponding to item 3 above), where the state evolution of two tasks under the actions taken by two policies are presented. In period τ, task j has priority over i and policy π processes i but not j. In Table 2, the original policy π processes task j but not i in period $\tau+1$. According to Definition 2, the interchanging policy $\tilde{\pi}$ processes task i but not j in period $t = \tau+1$. In Table 3, on the other hand, such a period t does not exist: the original policy π does not process task i or j in period $\tau+1$. In this case, the interchanging policy $\tilde{\pi}$ will follow exactly what the original policy π does after period τ. We finally note that in the second scenario, the interchanging policy $\tilde{\pi}$ results in a non-completion penalty of $2q(1)$, which cannot be higher the non-completion penalty resulting from the original policy π, $q(2)$, due to the convexity of the non-completion penalty function $q(\cdot)$.

Period (s)	τ	$\tau+1$	$\tau+2$	$\tau+3$	$\tau+4$
Capacity (N_s)	1	1	1	0	0
Task $j:(x^j_s, y^j_s)$	(4,3)	(3,3)	(2,2)	(1,2)	penalized $q(2)$
Task $i:(x^i_s, y^i_s)$	(3,2)	(2,1)	(1,1)	completed	–
Policy $\pi:(u^j_s, u^i_s)$	(0,1)	(1,0)	(0,1)	(0,0)	(0,0)
Policy $\tilde{\pi}:(\tilde{u}^j_s, \tilde{u}^i_s)$	**(1,0)**	**(0,1)**	(0,1)	(0,0)	(0,0)
Task $j:(x^j_s, y^j_s)$	(4,3)	(3,2)	(2,2)	(1,2)	penalized $q(2)$
Task $i:(x^j_s, y^j_s)$	(3,2)	(2,2)	(1,1)	completed	–

Table 2 A simple scenario for which the period $t \in (\tau, b^i_\tau)$ defined in Definition 2 exists: $t = \tau+1$.

Period (s)	τ	$\tau+1$	$\tau+2$	$\tau+3$	$\tau+4$
Capacity (N_s)	1	2	0	0	0
Task $j:(x^j_s, y^j_s)$	(4,3)	(3,3)	(2,2)	(1,2)	penalized $q(2)$
Task $i:(x^i_s, y^i_s)$	(3,2)	(2,1)	completed	–	–
Policy $\pi:(u^j_s, u^i_s)$	(0,1)	(1,1)	(0,0)	(0,0)	(0,0)
Policy $\tilde{\pi}:(\tilde{u}^j_s, \tilde{u}^i_s)$	**(1,0)**	(1,1)	(0,0)	(0,0)	(0,0)
Task $j:(x^j_s, y^j_s)$	(4,3)	(3,2)	(2,1)	(1,1)	penalized $q(1)$
Task $i:(x^j_s, y^j_s)$	(3,2)	(2,2)	(1,1)	penalized $q(1)$	–

Table 3 A simple scenario for which the period $t \in (\tau, b^i_\tau)$ defined in Definition 2 does not exist.

We are now ready to state our main result below.

Theorem 1 **(The SSLP Principle)**. *For the make-to-order multi-task scheduling problem under varying processing costs, suppose that task j has priority to i (cf. Definition 1) at task state $\mathbf{z}_\tau$, and let $\tilde{\pi}$ be the interchanging policy generated by a policy π. The interchanging policy $\tilde{\pi}$ cannot incur higher total costs than the original policy π along any sample path.*

Theorem 1 suggests that among two tasks with the same slack time, priority should be given to the task with *later* due date. (Note that our result holds *sample-pathwise*.) This is in sharp contrast to the earliest due date first (EDF) policy commonly implemented in practice. Even though this priority rule is only a partial characterization of optimal policies, it sheds some light on this complex scheduling problem, and managers can use this insight to design effective heuristics.

Now we focus on the proof of Theorem 1 in the remainder of this section. The proof of Theorem 1 is an immediate consequence of Lemma 1 (showing feasibility of the interchanging policy $\tilde{\pi}$) and Lemma 2 (showing improvement of $\tilde{\pi}$ over the original policy π).

Lemma 1. *An interchanging policy $\tilde{\pi}$ is feasible.*

We note that the feasibility of policy $\tilde{\pi}$ after period t follows from Definition 2 (if such a period t does not exist, we can set $t = \infty$). To argue the feasibility of the interchanging policy $\tilde{\pi}$, we show in the proof that

1. in period t when the original policy π first processes task j but not i, it is feasible for the interchanging policy $\tilde{\pi}$ to process task i;
2. in period $k = \tau+1, \ldots, t-1$, whenever the original policy π processes task j, it is feasible for the interchanging policy $\tilde{\pi}$ to process task j.

Proof of Lemma 1. It is straightforward to verify the first point. Within periods $[\tau, t)$, the interchanging policy $\tilde{\pi}$ has one less unit of i to process than π. (This is because π processes i in τ but $\tilde{\pi}$ does not, and within periods (τ, t), $\tilde{\pi}$ copies the actions of π.) This suggests that in period $t < b^i_\tau$, the policy $\tilde{\pi}$ can process task i since task i still has at least 1 unit to be processed.

To prove the second point, we consider three cases as follows.

Case 1: Suppose there exists a period $t \in (\tau, b^i_\tau)$ in which the original policy π first processes task j but not i. Since in period τ, task j has higher priority to task i, we have $y^j_\tau \geq y^i_\tau$ (i.e., the remaining process time of task j is no less than that of task i in period τ). Then in period $\tau+1$, $\tilde{y}^j_{\tau+1} = y^j_\tau - 1$ under the interchanging policy $\tilde{\pi}$ and $y^i_{\tau+1} = y^i_\tau - 1$ under the original policy π. This implies that $\tilde{y}^j_{\tau+1} \geq y^i_{\tau+1}$. We also know that in period $k = \tau+1, \ldots, t-1$, policy π must process task i whenever it processes task j (according to the definition of t). Then it is feasible for the interchanging policy $\tilde{\pi}$ to process task j whenever the original policy π processes task j.

Case 2: Suppose there does not exist a period $t \in (\tau, b^i_\tau)$ in which the original policy π first processes task j but not i, and task j's due date is no later than i's, i.e. $b^j_\tau \leq b^i_\tau$. The argument is identical to that of Case 1. Still, in period $\tau+1$ the remaining processing time $\tilde{y}^j_{\tau+1}$ of task j under the interchanging policy is no less than the remaining processing time $y^i_{\tau+1}$ of task i under the original policy π. We also know that in period $k = \tau+1, \ldots, b^i_\tau$, policy π must process task i whenever it processes task j, since $b^j_\tau \leq b^i_\tau < t = \infty$. Then it is feasible for the interchanging policy $\tilde{\pi}$ to process task j whenever the original policy π processes task j.

Case 3: Suppose there does not exist a period $t \in (\tau, b^i_\tau)$ in which the original policy π first processes task j but not i, and task j's due date is later than i's, i.e. $b^j_\tau > b^i_\tau$. In this case, since t does not exist and $b^j_\tau > b^i_\tau$, so within periods (τ, b^i_τ), the original policy π processes task i whenever it processes task j. Since in period τ, task j has higher priority to task i, we have $s^j_\tau \leq s^i_\tau$ (i.e., the slack time of task j is no greater than that of task i in period τ). Then in period $\tau+1$, $s^i_{\tau+1} = s^i_\tau$ and $s^j_{\tau+1} = s^j_\tau - 1$ under the original policy π (since π processes i but not j in period τ). This implies that $s^j_{\tau+1} < s^i_{\tau+1}$ and therefore $s^j_{b^i_\tau} < 0$. It then follows that under the interchanging policy $\tilde{\pi}$, the slack time of task j is non-positive in period b^i_τ (i.e., $\tilde{s}^j_{b^i_\tau} \leq 0$). This suggests that $\tilde{\pi}$ can possibly process j in every period within $[b^i_\tau, b^j_\tau)$. Then it is feasible for the interchanging policy $\tilde{\pi}$ to process task j whenever the original policy π processes task j. **Q.E.D.**

The following lemma shows that a feasible interchanging policy $\tilde{\pi}$ weakly dominates π along every sample path.

LEMMA 2. *The interchanging policy $\tilde{\pi}$ cannot incur higher total costs than the original policy π along any sample path.*

Proof of Lemma 2. Within this proof we fix a given sample path $\mathbf{f}^\tau_T = (f_{\tau+1}, \ldots, f_T)$. We consider two cases.

Case 1: Suppose first that there exists a period $t \in (\tau, b^i_\tau)$ in which the original policy π first processes task j but not i. In this case, the two policies $\tilde{\pi}$ and π always use the same amount of capacity, and thus these two policies result in the same (ex-post) total processing cost, along every possible sample path (cf. the expression of total processing cost in (1)). Finally, these two policies lead to the same total non-completion penalty (for all tasks) along every sample path, since policy $\tilde{\pi}$ can also finish both tasks i and j before their due dates. As a result, the two policies $\tilde{\pi}$ and π must result in the same total cost along the given sample path.

Case 2: The second case is more involved. Suppose now that there does not exist a period t, i.e., the original policy π processes task i whenever it processes task j, for period $k = \tau + 1, \ldots, \min\{b^i_\tau, b^j_\tau\} - 1$. Again, the two policies $\tilde{\pi}$ and π use the same amount of capacity in every period, and therefore result in the same total processing cost along the sample path.

It is possible that the two policies lead to different non-completion penalty on tasks i and j. We use r^i to denote the remaining processing time of task i at its due date (on the beginning of period b^i_τ), under the original policy π along the considered sample path. We note that under the interchanging policy $\tilde{\pi}$, task i's remaining processing time at its due date must be r^i+1. Similarly, if we use r^j to denote task j's remaining processing time at its due date under policy π, then task j's remaining processing time at its due date under policy $\tilde{\pi}$ must be r^j-1. Since task j's remaining processing time is no less than task i's in period τ, we must have $0 \le r^i < r^j$. Due to the convexity of non-completion penalty costs, it follows that the total non-completion penalty resulting from policy $\tilde{\pi}$ cannot be higher than that resulting from π, because

$$q(r^i)+q(r^j) \ge q(r^i+1)+q(r^j-1), \qquad \text{for all} \qquad 0 \le r^i < r^j.$$

Hence, the interchanging policy $\tilde{\pi}$ cannot result in a higher total cost than the original policy π, along the given sample path ($\tilde{\pi}$ incurs potentially less non-completion costs than π). **Q.E.D.**

4. An Effective Heuristic Policy and Numerical Experiments

Since computing the optimal policies using brute-force dynamic programming is intractable due to the curse of dimensionality (discussed earlier in §2), we exploit the structural properties of optimal policies to devise an effective SSLP-based policy to approximately solve this class of problems. The underlying idea of our proposed heuristic policy is conceptually simple – to compute a processing decision that strikes a good balance between the potential non-completion penalty costs (if processing less) and the processing costs (if processing more).

4.1. Description of the SSLP-based heuristic policy

To fully describe the heuristic policy, we introduce additional notation. Given the information set f_t, let r_t be the maximum number of tasks that can be processed in period t, i.e., $r_t = \min\{|I_t|, N_t\}$.

We describe a simple and effective way of determining the the number of active tasks (denoted by w_t) to be processed in each period t. First, we quantify the potential penalty cost for tasks not completed before due dates if we choose to process w_t in period t under the SSLP principle.

For any time period t, we generate an arbitrary sample-path f_T that contains all the (deterministic) task arrival and cost information until the end of planning horizon T. We rank all the active tasks in period t according to their slack times from shortest to longest (tie-breaking in favor of longer remaining processing time), and then choose to process the first w_t tasks from the ranked list. Conditioning on processing w_t number of active tasks in period t, we proceed to the next period $t+1$. We then carry out the same ranking of active tasks I_{t+1} (which depends on w_t), and choose to process the first r_{t+1} tasks (maximum possible) from the ranked list. We then repeat the procedure in period $t+1$ until the end of planning horizon T. Essentially, we process w_t number of active tasks in period t and then use full capacities to process active tasks from period $t+1$ to period T (under our ranking rule).

We then sum up all the non-completion penalty costs from t to T along this sampth path f_T, and denote it by $Q_t(w_t; f_T)$. This quantity is in fact the potential non-completion penalty cost impact caused by the decision w_t made in period t. It is not hard to check that $Q_t(w_t; f_T)$ is decreasing (in the non-strict sense) in w_t under our ranking rule, i.e., the more we process in period t, the less potential non-completion penalty cost we pay.

Second, we quantify the total processing costs of producing the first w_t tasks from the ranked list in period t as $S(w_t; f_T) = \sum_{i \in w_t} c_t^i$, which is increasing in the number of tasks w_t. Here we slightly abuse the notation w_t (in the summation) to also mean the set of tasks.

The production amount w_t^* in period t can then be computed as

$$w_t^* = \arg\min_{w_t \in \{0,\ldots,r_t\}} \mathbb{E}[Q(w_t) + S(w_t)], \tag{4}$$

where the expectation is taken over all possible sample paths f_T. One can approximate the expectation well by generating sufficiently large number of sample paths (typically having 1000 sample paths leads to an error tolerance $\pm 0.1\%$ in our examples). Due to lack of structure of $Q(\cdot)$, one may have to exhaustively search for w_t^*. Implementation-wise, if the non-penalty completion penalty is much more convex than the (linear) processing cost, one can search the optimal solution w_t^* from r_t downwards, and stop when the objective value in (4) does not improve. Upon computing the production amount w_t in each period t, we then exploit the optimality of SSLP principle to devise an effective heuristic policy presented in Algorithm 1.

4.2. Numerical experiments

An important question is how our proposed heuristic policy performs numerically. In this section, we have conducted extensive numerical experiments, and the numerical results show that our proposed policy performs consistently well. All algorithms are implemented in Matlab R2014a on an Intel Core i7-3770 3.40GHz PC.

Choice of benchmark. The prevalent strategies in the literature is to use earliest-due-date-first-based (EDF) policies (e.g., Stankovic et al. (1998), Doytchinov et al. (2001), Moyal (2013)), i.e., the decision maker processes the active tasks that have the earliest due dates. This type of policies are very easy to implement and ubiquitous in practice (e.g., Phan et al. (2011) in cloud computing

Algorithm 1 Procedures for computing the SSLP-based heuristic policy

Require: demand processes, cost processes, information sets

Ensure: *totalCost*

```
function SSLP-Based Policy
    for t ← 1 to T do
        Compute the production quantity w_t^* using (4).
        Rank all active tasks in I_t according to their slack times from shortest to longest
        (tie-breaking with longer remaining processing time);
        Process the first w_t^* tasks from the ranked list of active tasks.
    end for
    return totalCost
end function
```

applications). Since T is set long and N_t is kept time-invariant in our numerical experiments, we restrict our attention to stationary EDF-based policies (i.e., processing $\min\{|I_t|, \gamma N_t\}$ tasks in each period t where γ is a given threshold percentage). For each problem instance, we optimize on γ and choose the best stationary EDF-based policy as our benchmark to gauge the performance of our SSLP-based policy.

Performance measure. Since computing the exact optimal solution is intractable, we shall compare the total cost of our proposed policy $\pi^{\mathbf{SSLP}}$, with that of the benchmark policy $\pi^{\mathbf{EDF}}$. Then the improvement ratio is defined as follows,

$$\text{Improvement ratio}\eta = \left(1 - \frac{\mathscr{C}(\pi^{\mathbf{SSLP}})}{\mathscr{C}(\pi^{\mathbf{EDF}})}\right) \times 100\%,$$

where $\mathscr{C}(\pi)$ is defined as the total cost incurred by an arbitrary feasible policy π. That is, the improvement ratio is the percentage cost improvement of our SSLP-based policy over the benchmark EDF-based policy.

Design of experiments. The numerical experiments are designed as follows. Since our paper is focused on the multi-task due date scheduling problem with random processing costs, we keep the demand process independent and identically distributed (i.i.d.), and choose to test the following four stochastic cost processes (including one independent and three correlated processes) as follows:

(a) Independent and identically distributed (i.i.d.) cost process (IID);

(b) Markov-modulated cost model with three states (MMC).

(c) Autoregressive cost model AR(1) with decreasing trend (ARD);

(d) Autoregressive cost model AR(1) with increasing trend (ARI);

We set the planning horizon $T = 100$ periods and the production capacity $N_t = 16$ for each period t. The demand process is initialized as follows: the number of incoming active tasks follows an i.i.d. Poisson distribution with rate $\lambda \in [6, 8]$ in each period, which represents a mild overloaded regime; and for each incoming active task, the remaining processing time is an i.i.d. discrete uniform distribution on $[1, 4]$ and the remaining time is its remaining processing time plus another i.i.d. discrete uniform distribution on $[1, 4]$. We consider three types of non-completion penalty cost functions (convex in remaining processing time y), e.g., linear cost function $q(y) = 30y$, quadratic cost function $q(y) = 30y^2$ and exponential cost function $q(y) = (6)5^y$.

Next we specify our four processing cost models. We set the base cost c^i for each task i dependent on the type $\kappa(i)$ of task i, i.e., $c^i = c^{\kappa(i)}$ for each task i. For simplicity, we consider two types of tasks: $\kappa(i) = 1$ if task i is discounted, $\kappa(i) = 2$ if task i is regular, and we set $c^1 = 15$ and $c^2 = 20$. Each incoming task i has equal probability to be one of the two types defined above.

For the IID model (a), the cost in period t is $c_t^{\kappa(i)} = c^{\kappa(i)} + \epsilon_t$ for each task i, where the random perturbation term ϵ_t follows an i.i.d. Normal distribution with mean 0 and standard deviation 2. For the MMC model (b), the cost is governed by the state of the economy: low-cost economy (labeled as state 1), and high-cost economy (labeled as state 2). If the state of the economy in period t is j ($j = 1, 2$), then the cost in period t is $c_t^{\kappa(i),j} = j(c^{\kappa(i)} + \epsilon_t)$ for task i where the random perturbation term ϵ_t follows an i.i.d. Normal distribution with mean 0 and standard deviation 2. We assume that the state of the economy follows an exogenous Markov chain with transition probabilities

$$p_{11} = p_{22} = 0.8, \quad p_{12} = p_{21} = 0.2.$$

For the ARD model (c), the cost in period t is $c_t^{\kappa(i)} = \alpha c_{t-1}^{\kappa(i)} + \epsilon_t$ for each task i, where the drift term $\alpha = 0.99$ and the random perturbation term ϵ_t follows an i.i.d. Normal distribution with mean 0 and standard deviation 0.5, and initial value $c^{\kappa(i)}$. For the ARI model (d), the cost structure is identical to (c) except that the drift term $\alpha = 1.01$.

Numerical Results. For brevity in presentation, we only present the numerical results for the quadratic and exponential penalty costs, since the linear case is comparable to the quadratic case. Figures 1 – 2, 3 – 4, 5 – 6 and 7 – 8 show the average costs of EDF- and SSLP-based policies for the IID model (a), the MMC model (b), the ARD model (c) and the ARI model (d), respectively. In general, we observe that the improvement ratio η increases as the system gets more overloaded, i.e., more tasks are subject to non-completion penalty costs. Also, the improvement ratio η increases as the non-completion penalty cost becomes more convex.

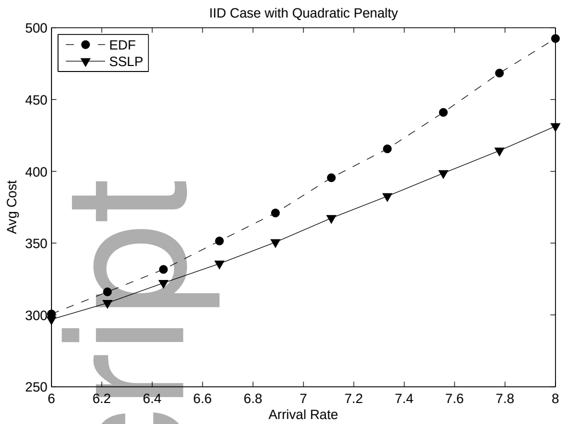


Figure 1 IID with Quadratic $q(y) = 30y^2$

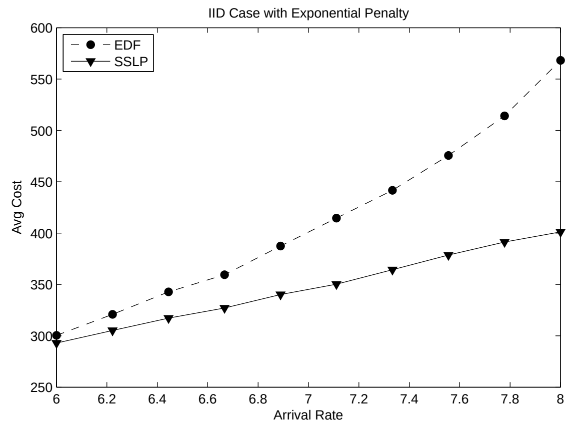


Figure 2 IID with Exponential $q(y) = (6)5^y$

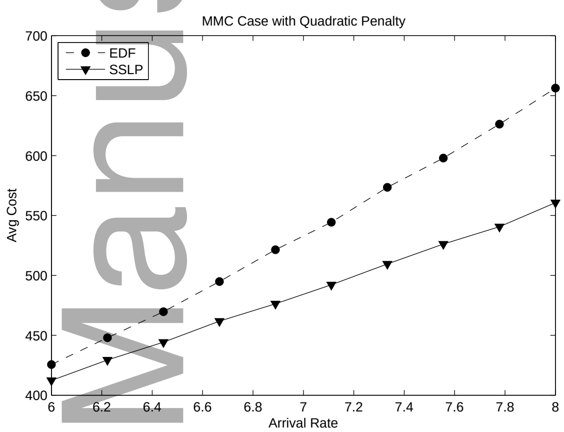


Figure 3 MMC with Quadratic $q(y) = 30y^2$

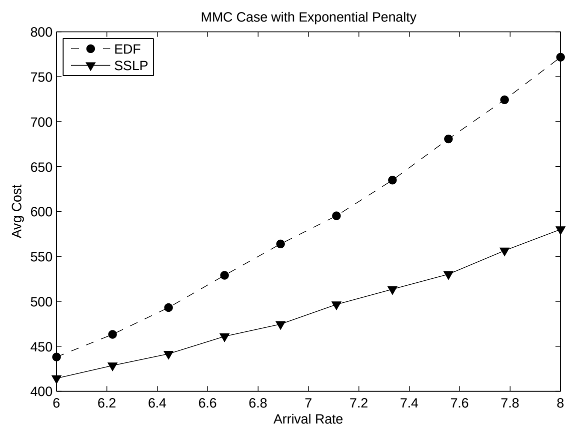


Figure 4 MMC with Exponential $q(y) = (6)5^y$

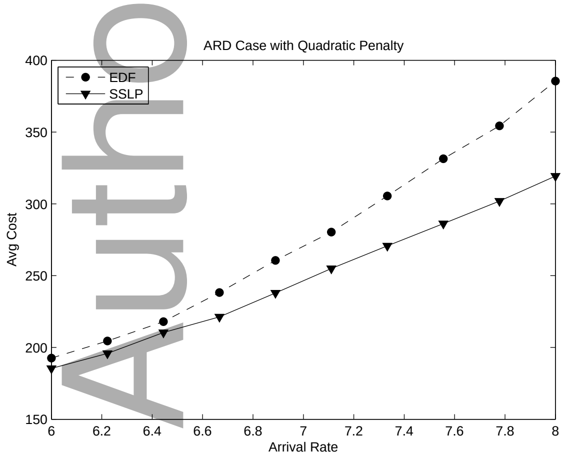


Figure 5 ARD with Quadratic $q(y) = 30y^2$

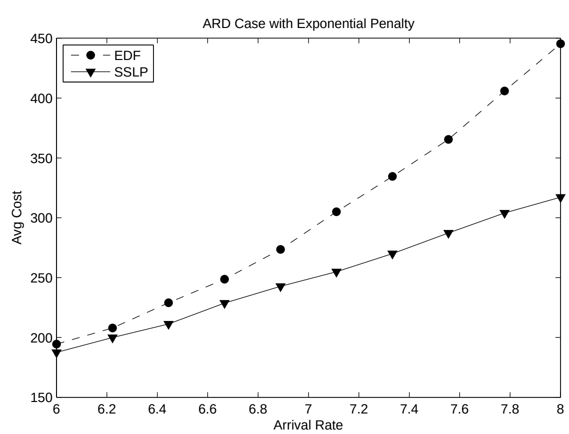


Figure 6 ARD with Exponential $q(y) = (6)5^y$

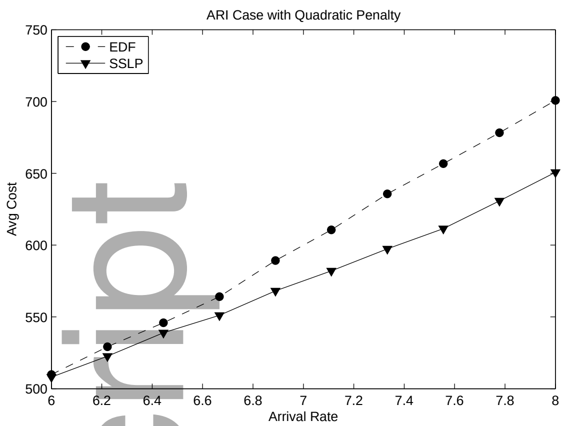


Figure 7 ARI with Quadratic $q(y) = 30y^2$

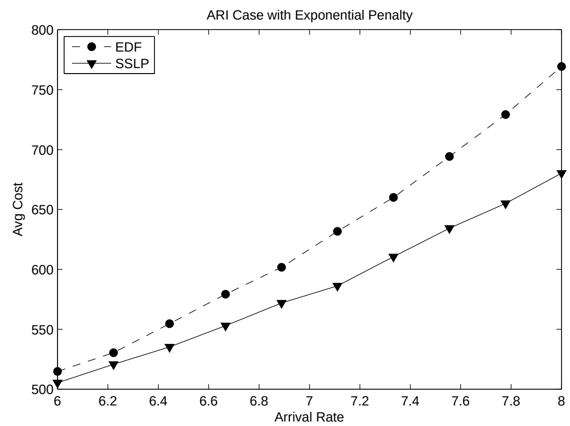


Figure 8 ARI with Exponential $q(y) = (6)5^y$

For the IID and MMC and ARD models, in the mild overloaded regime, the improvement ratio η can go up to 20% and 35% for quadratic and exponential penalty cost functions, respectively.

For the ARI model, the improvement ratio η is observed to be much lower, up to 10% and 15% for quadratic and exponential penalty cost functions, respectively. The key reason for lower improvement ratios lies in the processing cost component. As seen from our theoretical analysis, the benefit of SSLP in the processing cost component is due to the fact that the firm prefers to have many small unfinished tasks that could be simultaneously processed in some future period when the processing cost becomes lower. As a result, if the processing cost has a clear increasing trend, this benefit of saving processing costs becomes very small. Most of the improvement comes from the non-completion penalty cost component. We also remark that, in the case of ARI, both the EDF- and SSLP-based heuristic policies are expected to work very well in practical implementations (since any near-optimal policies should process as many tasks as early as possible when the processing cost is increasing).

It is worth mentioning that the major advantage of our SSLP-based heuristic policy is that, based on the demand and cost information, all the decisions could be made in an online manner (which avoids recursive computation), which makes the algorithm especially appealing in practice.

5. Extensions

In §5.1, we establish the SSLP principle in a more general setting with heterogeneous non-completion penalty. In §5.2, we consider an alternative setting where the processing rate constraint is relaxed, and show that EDF-based policies are optimal under Assumption 1.

5.1. Heterogeneous non-completion penalty cost

We have assumed that all tasks have the same non-completion penalty function. Now we consider a more general model where each task has an individual penalty cost function. For each task $j \in J_t^2$ that is not completed before its due date, the manager incurs a nonnegative penalty cost in terms of its remaining processing time, denoted by $q^j(y_t^j - u_t^j)$, with $q^j(0) = 0$. The total non-completion penalty cost incurred in period t is given by $\sum_{j \in J_t^2} q^j(y_t^j - u_t^j)$.

We first note that the definition of the interchanging policy $\tilde{\pi}$ as well as Lemma 1 naturally extend to the generalized setting considered in this subsection. The following lemma will be useful in the establishment of the SSLP principle.

Lemma 3. *Suppose that $i \preccurlyeq j$ at task state $\mathbf{z}_\tau$, a policy π processes i but not j at system state $(\mathbf{z}_\tau, f_\tau)$, and that along a given sample path, π can finish both tasks i and j before their due dates. Then, along this sample path there exists a period $t \in (\tau, b_\tau^i)$ in which policy π processes task j but not i.*

Proof of Lemma 3. Within this proof, we consider a fixed sample path $(f_{\tau+1}, \ldots, f_T)$. Since $i \preccurlyeq j$ at task state $\mathbf{z}_\tau$, and policy π processes i but not j at system state $(\mathbf{z}_\tau, f_\tau)$, task j must have strictly shorter slack time and longer remaining processing time in period $\tau + 1$, i.e., $s_{\tau+1}^j < s_{\tau+1}^i$ and $y_{\tau+1}^j > y_{\tau+1}^i$ under policy π.

Suppose that Lemma 3 does not hold, i.e., in period $k = \tau+1, \ldots, b_\tau^i - 1$, whenever the policy π processes j, it must also process i. If the due date of task i is not earlier than that of task j, then the policy π cannot finish task j before its due date, since in period $\tau + 1$ task j has strictly longer remaining processing time than i.

If, on the other hand, the due date of task i is earlier than that of task j, then before task i's due date, whenever the policy π processes j, it must also process i. Since in period $\tau + 1$ task j has strictly shorter slack time than i, we conclude that at task i's due date b_τ^i the slack time of task j must be negative. As a result, policy π cannot finish task j before its due date. We have proved the lemma by contradiction. **Q.E.D.**

If the original policy π can finish both tasks i and j along a given sample path, Lemma 3 shows the existence of a period $t \in (\tau, b_\tau^i)$ in which policy π processes task j but not i. In this case, we have shown in the proof of Theorem 1 that the two policies $\tilde{\pi}$ and π must result in the same total cost on this sample path.

Theorem 2 **(The SSLP Principle)**. *Suppose that task j has priority to i (cf. Definition 1) at task state $\mathbf{z}_\tau$, and let $\tilde{\pi}$ be the interchanging policy generated by a policy π. The interchanging policy $\tilde{\pi}$ weakly dominates the original policy π, i.e.,*

$$V_\tau^\pi(\mathbf{z}_\tau, f_\tau) \geq V_\tau^{\tilde{\pi}}(\mathbf{z}_\tau, f_\tau), \qquad \forall f_\tau, \tag{5}$$

if at least one of the following two conditions holds:

1. *the original policy π can finish both tasks i and j before their due dates with probability one;*
2. *task j's incremental non-completion penalty is no less than task i's, i.e,*

$$q^j(y) - q^j(y-1) \geq q^i(y') - q^i(y'-1), \qquad 1 \leq y \leq y^j_\tau, \qquad 1 \leq y' \leq y. \tag{6}$$

If the first condition holds, e.g., when there is enough time to complete both tasks, it follows from Theorem 1 and Lemma 3 that the two policies $\tilde{\pi}$ and π incur the same total cost with probability one. If the second condition holds, Theorem 2 shows that the SSLP principle helps reduce the non-completion penalty cost.

Proof of Theorem 2. If the first condition holds, it follows from Lemma 3 that there exists a period $t \in (\tau, b^i_\tau)$ in which the original policy π first processes task j but not i with probability one. In this case, it follows from the proof of Theorem 1 that the two policies $\tilde{\pi}$ and π must result in the same expected total cost.

We now consider a sample path $(f_{\tau+1}, \ldots, f_T)$ along which the original policy π processes task i whenever it processes task j, for period $k = \tau+1, \ldots, b^i_\tau - 1$. Again, the two policies $\tilde{\pi}$ and π use the same amount of capacity in every period, and therefore result in the same total processing cost. It is possible that the two policies lead to different non-completion penalty. We use r^i to denote the remaining processing time of task i at its due date (on the beginning of period b^i_τ), under the original policy π along the considered sample path. We note that under the interchanging policy $\tilde{\pi}$, task i's remaining processing time at its due date must be $r^i + 1$. Similarly, if we use r^j to denote task j's remaining processing time at its due date under policy π, then task j's remaining processing time at its due date under policy $\tilde{\pi}$ must be $r^j - 1$. Since task j's remaining processing time is no less than task i's in period τ, we must have $0 \leq r^i \leq r^j - 1$. It therefore follows from the second condition in (6) that the total non-completion penalty resulting from policy $\tilde{\pi}$ cannot be higher than that resulting from π, since

$$q^i(r^i) + q^j(r^j) \geq q^i(r^i+1) + q^j(r^j - 1).$$

Hence, the policy $\tilde{\pi}$ cannot result in a higher total cost than the original policy π, along the given sample path. **Q.E.D.**

5.2. The case without capacity constraint

In this subsection, we consider the case where no processing rate constraint is imposed on each individual task. However, the total processing capacity used by all tasks has to be bounded by N_t

in each period t. At a state $\mathbf{z}_t$, the feasible action space is

$$U_t^o(\mathbf{z}_t, f_t) \triangleq \left\{ \mathbf{u}_t : u_t^i \in \{0, \dots, y_t^i\} \text{ for all } i \in I_t \text{ and } \sum_{i \in I_t} u_t^i \leq N_t \right\}. \tag{7}$$

Definition 3 (An EDF-based Interchanging Policy). Let π be a heuristic policy that can finish all tasks before their due dates along every possible sample path. In every period, an EDF-based interchanging policy $\hat{\pi}$ uses the same amount of capacity as the original policy π, and gives priority to those active tasks with earliest due dates. Under policy $\hat{\pi}$, at every system state $(\mathbf{z}_t, f_t)$ and for every active task i, we have $u_t^i > 0$ only if for every task j with a due date earlier than i, its the remaining processing time y_t^j is zero. □

Before proceeding, we introduce additional notation that will be useful in the statement and proof of Theorem 3. Under one full sample path $\mathbf{f}_T = (f_0, \dots, f_T)$, we let $G^\pi(\mathbf{z}_0, \mathbf{f}_T)$ denote the total (realized) cost under policy π, i.e.,

$$G^\pi(\mathbf{z}_0, \mathbf{f}_T) = \sum_{k=0}^{T} C_t(\mathbf{z}_k, f_k, \mathbf{u}_k) \mid \mathbf{f}_T.$$

In the following Theorem 3, we will prove the feasibility of an EDF-based interchanging policy $\hat{\pi}$ defined above, i.e., it is always feasible for the policy $\hat{\pi}$ to process the same amount of orders as the original policy π. The crux of our argument rests on showing that the policy $\hat{\pi}$ can also finish all orders before their due dates, along every sample path. This result directly leads to the main result in Theorem 3.

Theorem 3. *We consider a policy π that can finish all tasks before their due dates along every sample path. Let $\hat{\pi}$ be an EDF-based interchanging policy generated from the policy π according to Definition 3. The policy $\hat{\pi}$ is feasible, and achieves the same realized total cost as the original policy π along every sample path, i.e.,*

$$G^\pi(\mathbf{z}_0, \mathbf{f}_T) = G^{\hat{\pi}}(\mathbf{z}_0, \mathbf{f}_T), \qquad \forall\, \mathbf{z}_0, \quad \forall\, \mathbf{f}_T.$$

Proof of Theorem 3. Within this proof, we fix an arbitrary sample path (i.e., a sequence of realizations $\mathbf{f}_T$). We will show by induction that the interchanging policy $\hat{\pi}$ can also finish all tasks before their due dates along this sample path. We note that this result directly implies that it is feasible for the interchanging policy $\hat{\pi}$ to utilize the same amount of capacity as the original policy π in every period. The desired result then follows from the fact that the two policies incur the same processing cost in every period, and both policies yield zero non-completion penalty.

It is straightforward to check (from Definition 3) that the above claim holds for $t = 0$. Now suppose that the claim holds for $t = 0, 1, \ldots, k-1$, we now argue that it holds for period k. We note that since the policy $\hat{\pi}$ finishes all tasks before their due dates in periods $0, 1, \ldots, k-1$, it must be feasible for policy $\hat{\pi}$ to use the same amount of capacity as policy π in period k.

Suppose that the desired result does not hold for period k, i.e., the interchanging policy $\hat{\pi}$ fails to finish all tasks before their due dates at the end of period k, i.e., there exists some task j with due date $k+1$ that is not finished by policy $\hat{\pi}$ at the end of period k. Let $w_k^{\pi}(\mathbf{z}_k, f_k) = \sum_i u_k^i$ denote the total capacity utilized in period k by policy π. According to Definition 3, in period k the EDF policy $\hat{\pi}$ must process $w_k^{\pi}(\mathbf{z}_k, f_k)$ unit of unfinished tasks with due date $k+1$. We therefore conclude that in period $k-1$, the total processing time of all unfinished tasks with due date $k+1$ is strictly less under the original policy π, i.e.,

$$\sum_{i \in I_{k-1}:\, b^i = k+1} y_{k-1}^i < \sum_{i \in \hat{I}_{k-1}:\, \hat{b}^i = k+1} \hat{y}_{k-1}^i, \tag{8}$$

where the hat symbol is used to denote quantities resulting from the interchanging policy $\hat{\pi}$, e.g., $\hat{I}_{k-1}$ is the set of unfinished tasks at period $k-1$, under the policy $\hat{\pi}$.

We note that both policies π and $\hat{\pi}$ complete all tasks with due dates no later than k, before their due dates. Since both policies always use the same amount of capacity and face the same set of arrived tasks, the inequality in (8) contradicts with Definition 3, since the interchanging policy $\hat{\pi}$ does not process any orders with due dates later than $k+1$ before it finishes tasks with due dates no later than k, and gives highest priority to tasks with due date $k+1$ once it completes all tasks with due dates no later than k, in every period $t \in \{0, \ldots, k-1\}$. **Q.E.D.**

Theorem 3 considers the case where the completion of all tasks before their due dates is always feasible. The theorem essentially shows the existence of an EDF policy that is at least as good as any policy that always finishes all tasks before their due dates. If each task's non-completion penalty is higher than its highest possible processing cost (for example, when the following Assumption 1 holds), then it is optimal to finish all tasks before their due dates (provided that it is feasible to do so), and therefore an optimal EDF policy must exist (see the formal statement in the following corollary).

Assumption 1. *Suppose that for every task i, its incremental non-completion penalty is no less than its highest per-unit processing cost, i.e.,*

$$q^i(y) - q^i(y-1) \geq c^i + \varepsilon(f_t), \qquad y = 1, 2, \ldots, \qquad \forall\, f_t.$$

We note that if Assumption 1 is violated, then the operator/manager may prefer to withhold capacity and pay non-completion penalty. This assumption may not always hold in practical situations when the jobs cannot be done in certain periods (due to factors such as unavailability of workers or unfavorable weather conditions) , i.e., the processing cost is infinite in those periods.

COROLLARY 1. *Suppose that Assumption 1 holds and that it is feasible to finish all tasks along every sample path. There exists an optimal EDF policy that minimizes the expected total cost.*

We note that similar results on the optimality of EDF policies are well known in periodic task scheduling systems with a single processor (see, e.g., Liu and Layland (1973)) or multiple identical processors (see, e.g., Goossens et al. (2003) under additional assumptions on processor capacity). While processing cost are usually assumed to be constant in the aforementioned literature, in our setting the processing cost is allowed to be time-variant and stochastic. We would like to emphasize here that for the case with capacity constraint we have shown (in Example 1) that giving priority to tasks with earlier due dates could be suboptimal, even if it is feasible to complete all tasks before their due dates.

When the completion of all tasks is not feasible, on the other hand, it is well known that EDF policies may perform poorly (see Locke (1986)). In this "overload" setting (without capacity constraint) there does not exist an optimal "on-line" policy which makes scheduling decision based only on the states of active tasks that have arrived (see G. Buttazzo (1995)). We further believe that in this overload setting, there does not exist an optimal rule that determines the priority among active tasks without taking into account future system dynamics (like the SSLP principle established for the case with capacity constraint). As a result, in order to efficiently schedule existing active tasks the manager has to look ahead into future (random) task arrivals and processing costs, and the design of effective heuristic policies depends on the pattern of task arrival and processing cost processes.

6. Conclusion and Future Directions

We have established the optimality of Shorter Slack time and Longer remaining Processing time (SSLP) principle for capacitated MTO systems under varying processing costs and convex non-completion penalty costs: priority should be given to a task with shorter slack time and longer remaining processing time. Our main result suggests the optimality of the Latest Due date First (LDF) principle among jobs with the same slack time. This is sharply contrary to the convention and the common sense that the EDF policy is optimal. The main methodology employed in this paper is using interchanging arguments to show that the updated policies (with desired structures) dominate the original ones, which is an effective tool to analyze systems with huge state spaces (that render the direct computation using dynamic programming intractable). We have also proposed an effective cost-balancing heuristic policy for this class of problems, based on the derived structural results. Our computational results show that our proposed policies significantly outperform the myopic EDF-based benchmark policy.

We believe that this class of optimal production planning models in capacitated MTO systems under varying costs represents a fertile area for future research. To conclude the paper, we would like to point out several plausible future research directions. (a) The SSLP-based priority rule is still a partial characterization of optimal polices (see Example 2). (b) The current model only considers processing and non-completion penalty costs. Other costs, such as the idle costs of processors and holding costs of completed tasks, could be considered as well. (c) One may also consider the options to reject or accept the incoming tasks (where managers can control the production load by acceptance and rejection of incoming tasks). This class of problems would require new models and methodological innovations.

Acknowledgments

We sincerely thank the department editor Michael Pinedo, the anonymous senior editor, and the anonymous referees for their constructive comments and suggestions, which helped significantly improve both the content and the exposition of this paper. The research is partially conducted when Yunjian Xu was visiting the University of Michigan. The research of Yunjian Xu is partially supported by the MIT-SUTD International Design Center (IDC) Grant IDG21400103. The research of Cong Shi is partially supported by NSF grants CMMI-1362619 and CMMI-1451078.

References

Afèche, P., O. Baron, Y. Kerner. 2013. Pricing time-sensitive services based on realized performance. *Manufacturing & Service Operations Management* **15**(3) 492–506.

Baker, K. R., G. D. Scudder. 1990. Sequencing with earliness and tardiness penalties: A review. *Operations Research* **38**(1) 22–36.

Benjaafar, S., W. L. Cooper, S. Mardan. 2011. Production-inventory systems with imperfect advance demand information and updating. *Naval Research Logistics* **58**(2) 88–106.

Buchbinder, N., T. Kimbrel, R. Levi, K. Makarychev, M. Sviridenko. 2013. Online make-to-order joint replenishment model: Primal-dual competitive algorithms. *Operations Research* **61**(4) 1014–1029.

Bülbül, K., P. Kaminsky, C. Yano. 2007. Preemption in single machine earliness/tardiness scheduling. *Journal of Scheduling* **10**(4-5) 271–292.

Carr, S., I. Duenyas. 2000. Optimal admission control and sequencing in a make-to-stock/make-to-order production system. *Operations Research* **48**(5) 709–720.

Chen, C. C., Y. Yih. 1996. Indentifying attributes for knowledge-based development in dynamic scheduling environments. *International Journal of Production Research* **34**(6) 1739–1755.

Chen, Z. L. 2010. Integrated production and outbound distribution scheduling: review and extensions. *Operations Research* **58**(1) 130–148.

Cheng, T. C. E., M. C. Gupta. 1989. Survey of scheduling research involving due date determination decisions. *European Journal of Operational Research* **38**(2) 156–166.

Davis, R. I., A. Burns. 2011. A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys* **43**(4) 35:1–35:44.

Dellaert, N. P., M. T. Melo. 1995. Heuristic procedures for a stochastic lot-sizing problem in make-to-order manufacturing. *Annals of Operations Research* **59**(1) 227–258.

Dertouzos, M. 1974. Control robotics: The procedural control of physical processes. *Proceedings of IFIP Congress*. 807–813.

Dobson, G., C. A. Yano. 2002. Product offering, pricing, and make-to-stock/make-to-order decisions with shared capacity. *Production and Operations Management* **11**(3) 293–312.

Doytchinov, B., J. Lehoczky, S. Shreve. 2001. Real-time queues in heavy traffic with earliest-deadline-first queue discipline. *Annals of Applied Probability* **11**(2) 332–378.

Draper, N. R., H. Smith. 1998. *Applied regression analysis*. No. 1 in Wiley series in probability and statistics: Texts and references section, Wiley, New York.

Duenyas, I., M. P. Van Oyen. 1996. Heuristic scheduling of parallel heterogeneous queues with set-ups. *Management Science* **42**(6) 814–829.

Farzan, A., M. Ghodsi. 2002. New results for lazy bureaucrat scheduling problem. *Proceedings of the 7th CSI Computer Conference*. CSICC '02, 66–71.

Federgruen, A., G. Mosheiov. 1997. Single machine scheduling problems with general breakdowns, earliness and tardiness costs. *Operations Research* **45**(1) 66–71.

G. Buttazzo, F. Sensini, M. Spuri. 1995. Value vs. deadline scheduling in overload conditions. *Proceedings of the 1995 IEEE Real-Time Systems Symposium*. 90–99.

Gallego, G., Ö. Özer. 2001. Integrating replenishment decisions with advance demand information. *Management Science* **47**(10) 1344–1360.

Goossens, J., S. Funk, S. Baruah. 2003. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-time Systems* **25**(2) 187–205.

Gupta, D., L. Wang. 2007. Capacity management for contract manufacturing. *Operations Research* **55**(2) 367–377.

He, Q.-M., E. M. Jewkes, J. Buzacott. 2002. The value of information used in inventory control of a make-to-order inventory-production system. *IIE Transactions* **34**(11) 999–1013.

Heath, D. C., P. L. Jackson. 1994. Modeling the evolution of demand forecasts with application to safety stock analysis in production/distribution system. *IIE Transactions* **26**(3) 17–30.

Iravani, S. M. R., T. Liu, D. Simchi-Levi. 2012. Optimal production and admission policies in make-to-stock/make-to-order manufacturing systems. *Production and Operations Management* **21**(2) 224–235.

Kaminsky, P., O. Kaya. 2008. Scheduling and due-date quotation in a make-to-order supply chain. *Naval Research Logistics* **55**(5) 444–458.

Keskinocak, P., S. Tayur. 2004. Due date management policies. D. Simchi-Levi, S. D. Wu, Z.-J. Shen, eds., *Handbook of Quantitative Supply Chain Analysis*, *International Series in Operations Research & Management Science*, vol. 74. Springer US, 485–554.

Leung, J. Y. T., Z. L. Chen. 2013. Integrated production and distribution with fixed delivery departure dates. *Operations Research Letters* **41**(3) 290–293.

Leung, J. Y. T., H. Li, M. Pinedo. 2006. Scheduling orders for multiple product types with due date related objectives. *European Journal of Operational Research* **168**(2) 370–389.

Liu, C. L., J. W. Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM* **20**(1) 46–61.

Locke, C. D. 1986. Best-effort decision-making for real-time scheduling. Phd thesis, Carnegie Mellon University.

Moyal, P. 2013. On queues with impatience: stability, and the optimality of earliest deadline first. *Queueing Systems* **75**(2-4) 211–242.

Panwar, S. S., D. Towsley, J. K. Wolf. 1988. Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service. *Journal of the ACM* **35**(4) 832–844.

Phan, L. T., Z. Zhang, Q. Zheng, B. T. Loo, I. Lee. 2011. An empirical analysis of scheduling techniques for real-time cloud-based data processing. *Proceedings of the 2011 IEEE International Conference on Service-Oriented Computing and Applications*. SOCA '11, IEEE Computer Society, Washington, DC, USA, 1–8.

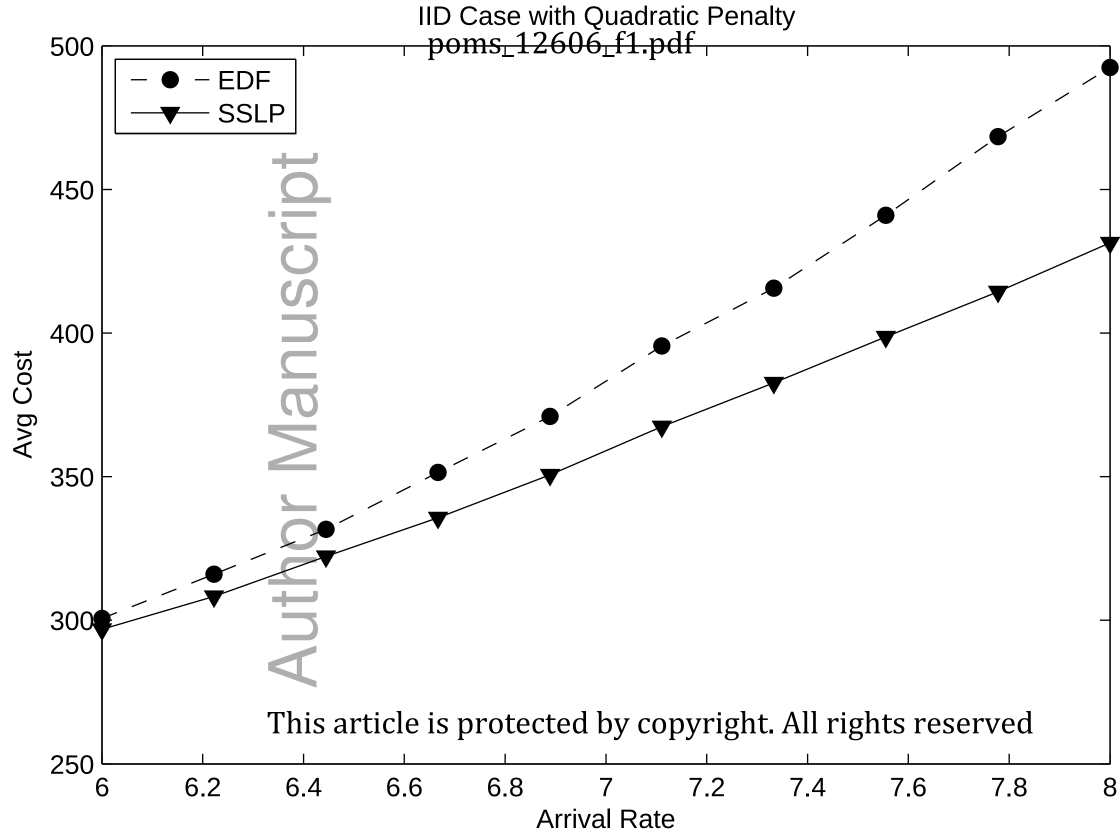
Pinedo, M. 1983. Stochastic scheduling with release dates and due dates. *Operations Research* **31**(3) 559–572.

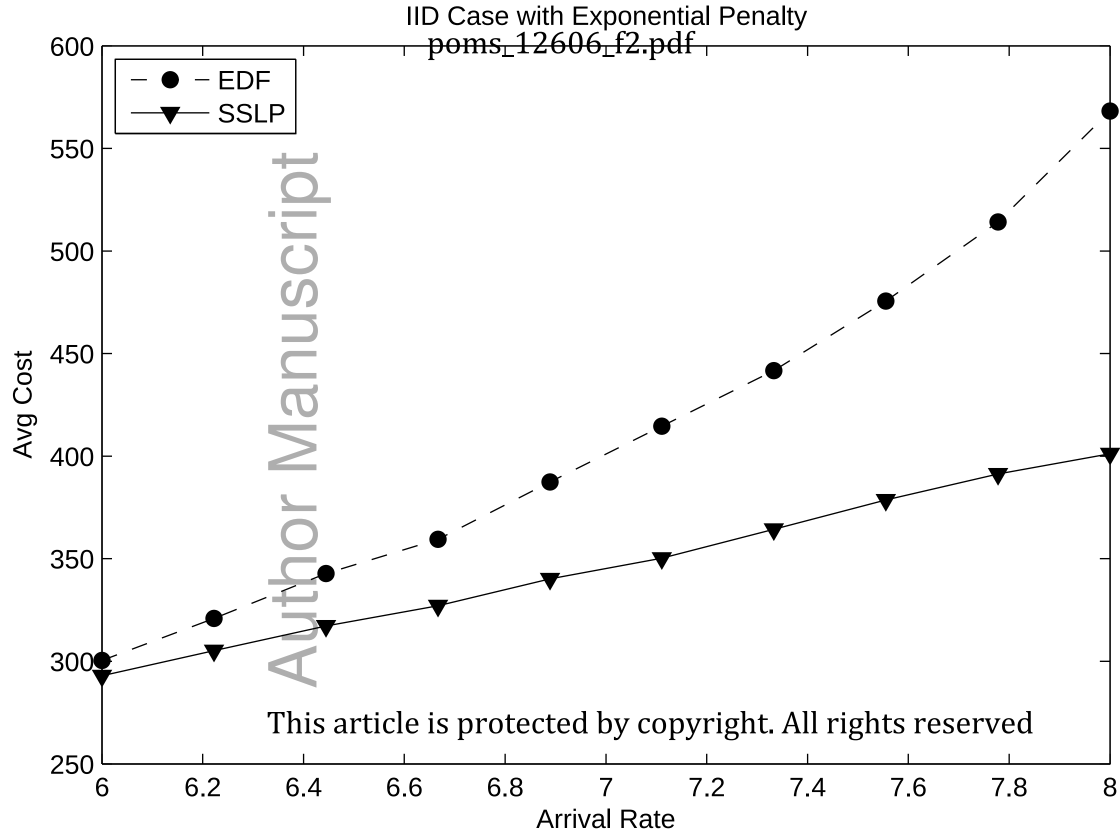
Pundoor, G., Z. L. Chen. 2005. Scheduling a production-distribution system to optimize the tradeoff between delivery tardiness and distribution cost. *Naval Research Logistics* **52**(6) 571–589.

Stankovic, J. A., K. Ramamritham, M. Spuri. 1998. *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA.

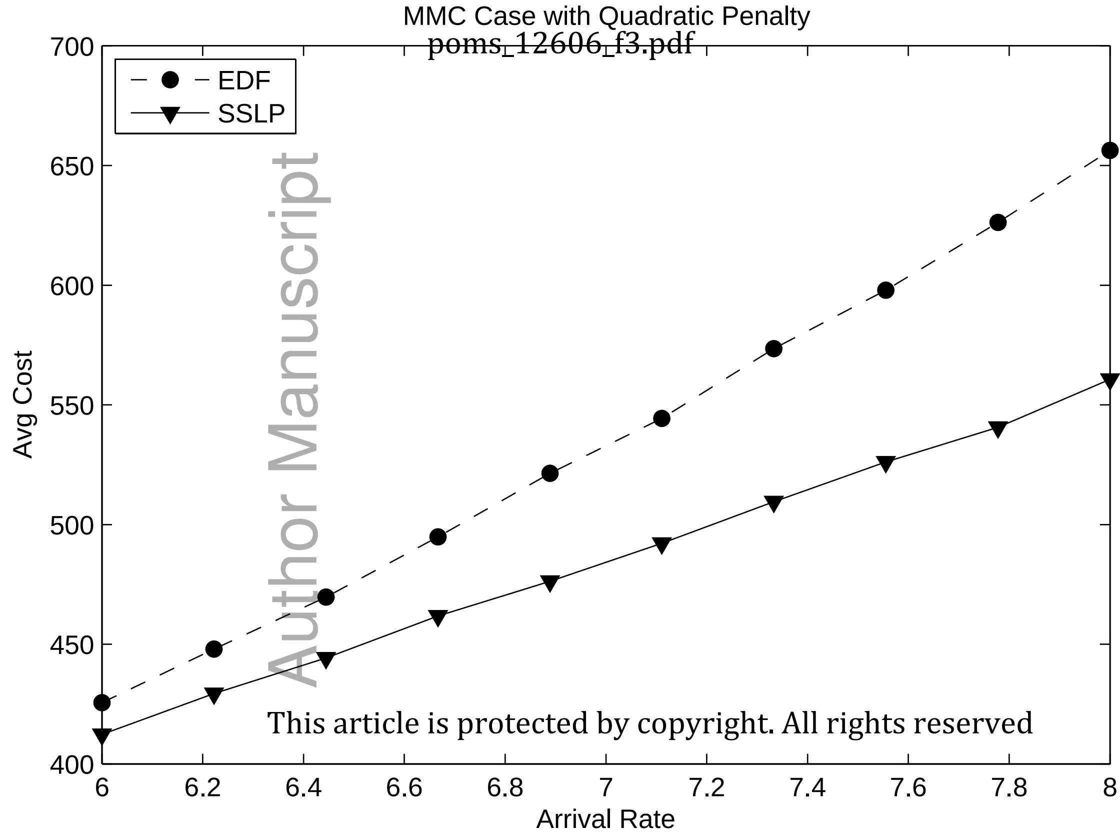
Youssef, K. H., C. Van Delft, Y. Dallery. 2004. Efficient scheduling rules in a combined make-to-stock and make-to-order manufacturing system. *Annals of Operations Research* **126**(1-4) 103–134.

Zhao, X., K. E. Stecke, A. Prasad. 2012. Lead time and price quotation mode selection: Uniform or differentiated? *Production and Operations Management* **21**(1) 177–193.


IID Case with Quadratic Penalty
poms_12606_f1.pdf
EDF
SSLP
Avg Cost
Arrival Rate
500
450
400
350
300
250
6
6.2
6.4
6.6
6.8
7
7.2
7.4
7.6
7.8
8
Author Manuscript

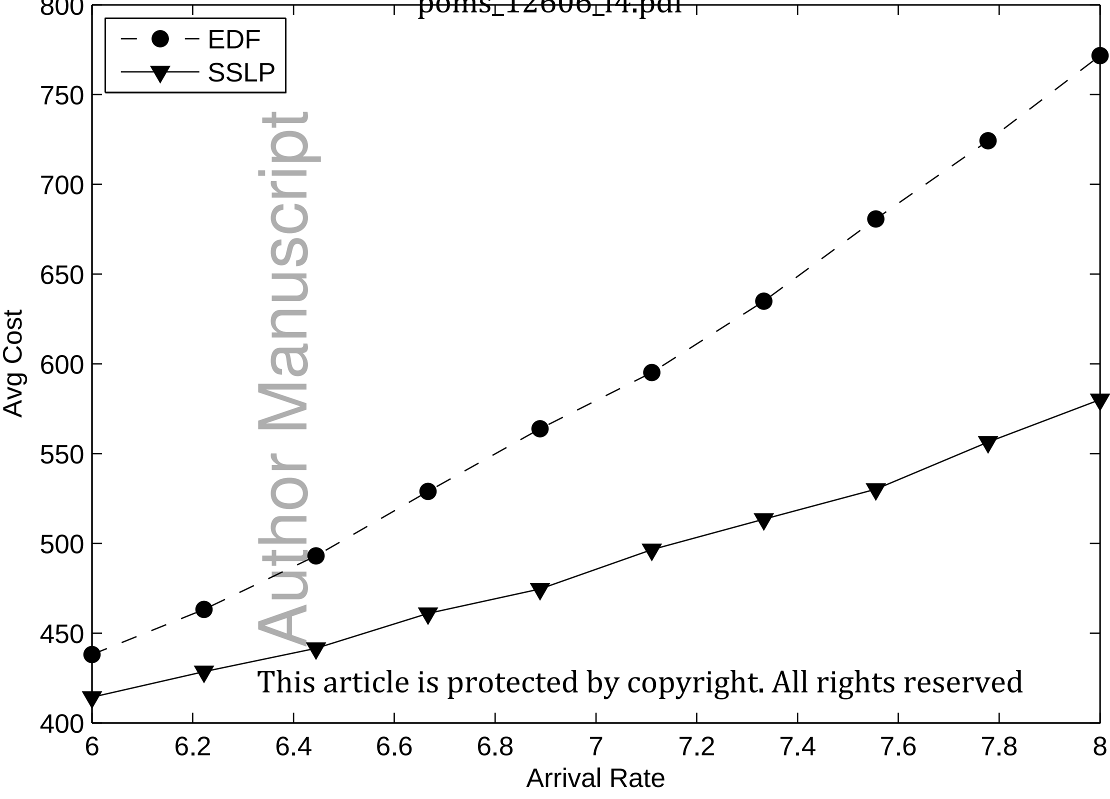


IID Case with Exponential Penalty
EDF
SSLP
Avg Cost
Arrival Rate
600
550
500
450
400
350
300
250
6
6.2
6.4
6.6
6.8
7
7.2
7.4
7.6
7.8
8

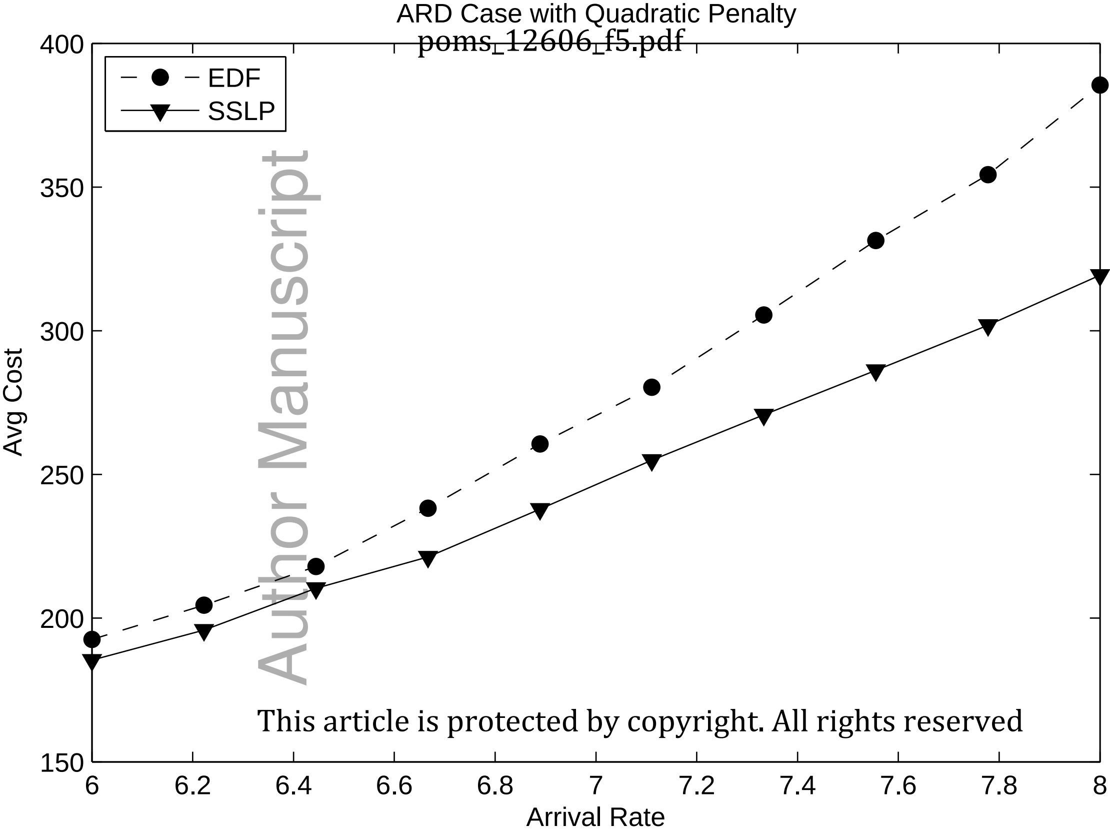

MMC Case with Quadratic Penalty
EDF
SSLP
Avg Cost
Arrival Rate
700
650
600
550
500
450
400
6
6.2
6.4
6.6
6.8
7
7.2
7.4
7.6
7.8
8

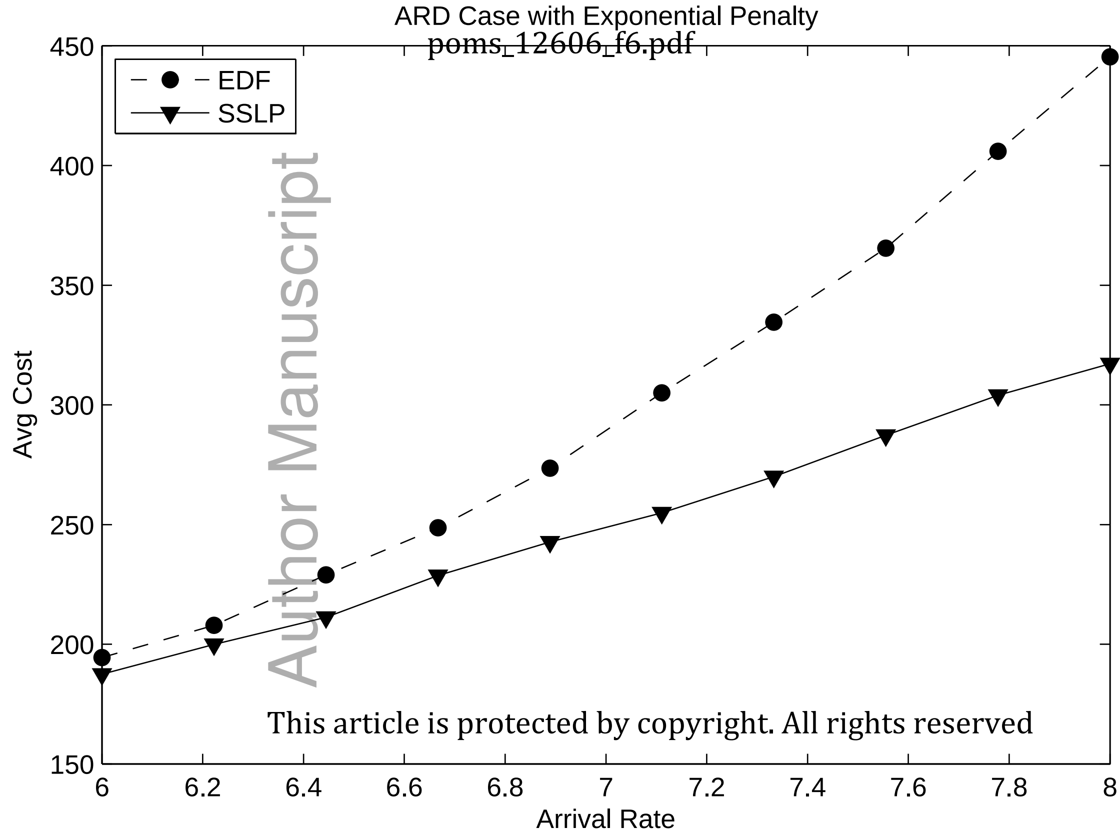
MMC Case with Exponential Penalty

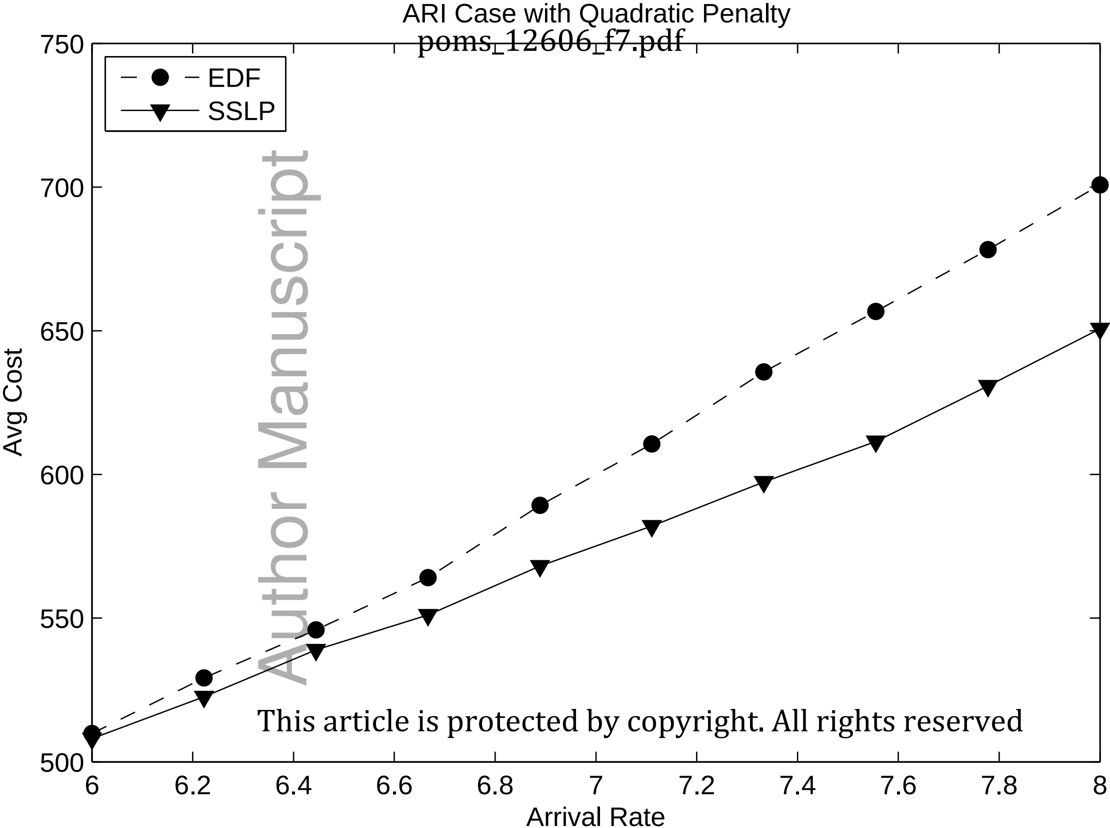
poms_12606_f4.pdf

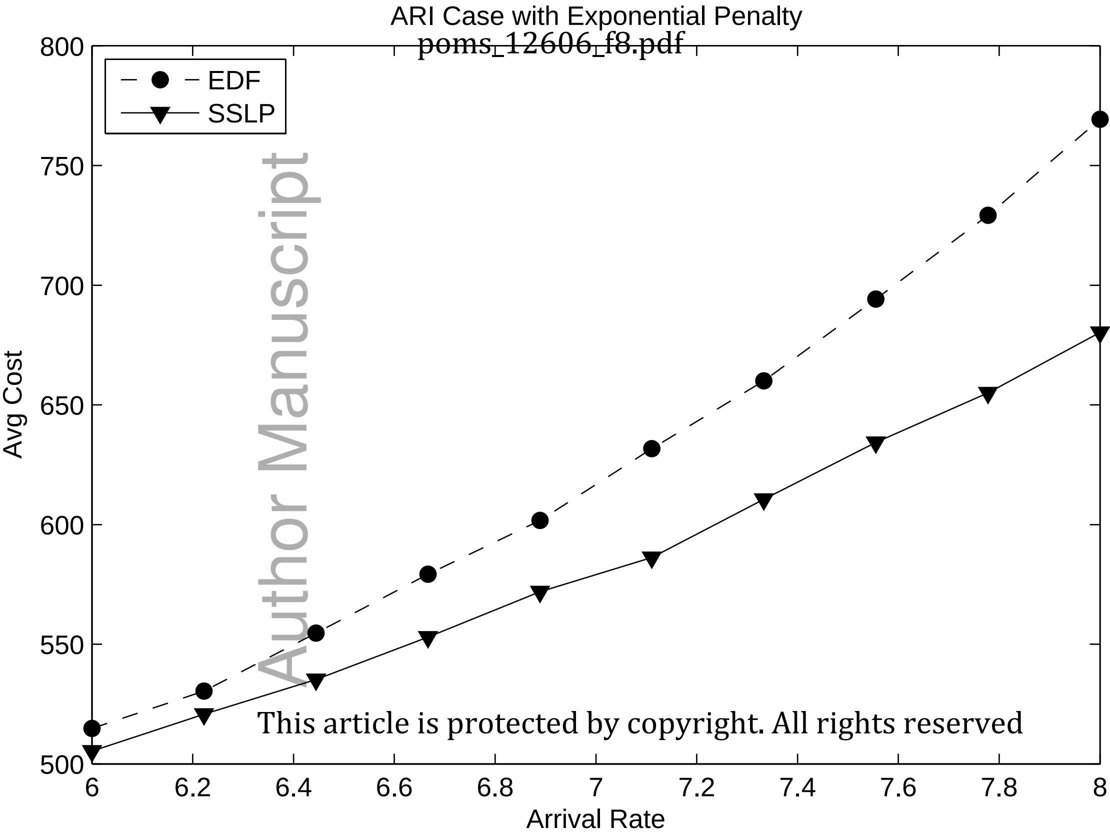


poms_12606_f5.pdf




ARD Case with Exponential Penalty
EDF
SSLP
Avg Cost
Arrival Rate
450
400
350
300
250
200
150
6
6.2
6.4
6.6
6.8
7
7.2
7.4
7.6
7.8
8


ARI Case with Quadratic Penalty
750
700
650
600
550
500
Avg Cost
6
6.2
6.4
6.6
6.8
7
7.2
7.4
7.6
7.8
8
Arrival Rate
EDF
SSLP


ARI Case with Exponential Penalty
EDF
SSLP
Avg Cost
Arrival Rate
800
750
700
650
600
550
500
6
6.2
6.4
6.6
6.8
7
7.2
7.4
7.6
7.8
8