Joint Production and Maintenance Operations in Smart Custom-Manufacturing Systems

Abstract

Machines in custom manufacturing environments with IoT (Internet-of-Things) capability are predicted to be pervading enterprises. However, there is a need to develop new algorithms that reap the benefits of such technologies. We consider a system where jobs with stochastic workloads arrive to a machine in an arbitrary fashion and upon arrival, their workload is revealed (enabled by IoT). The tool on the machine gets used up based on the speed at which the jobs are processed. Knowing that tool-replacement consumes a significant amount of time, we want to develop online algorithms that maximize the capacity of the machine by determining: (a) the speed at which each job is processed; and (b) the epoch when the tool is replaced. We provide online approaches that leverage the ability to reveal workload in real-time and effectively balance future uncertainties. We derive asymptotic bounds for the online algorithm performance and show using numerical experimentation that a little revealed information could result in a tremendous improvement in performance. Our online algorithms also work under realistic conditions of non-stationary batch arrivals and correlated workloads. Our work opens up research directions for a variety of operational settings that may benefit from revealing stochastic quantities by mining information.

Keywords— Machine Scheduling, Next-Fit Binpacking, Online Algorithm, Asymptotic Optimality

1 Introduction

In the coming years it is envisioned that there would be a significant rise in the number of custom manufacturing facilities (see Sutherland et al. [2016]). These facilities are expected to have highly versatile machines and actuators that are interconnected enterprise-wide through Internet of Things (IoT) (Monostori et al. [2016]). IoT enables universal manufacturing resource availability and accessibility by integrating and connecting physical assets into an information network (Gao et al. [2015]). From a production and maintenance operations standpoint, there are several benefits of IoT for custom manufacturing such as reducing variability and uncertainty as well as jointly performing production and maintenance activities in a single framework which significantly improves productivity and efficiency of the system as a whole (Gao et al. [2015], Takenaka et al. [2016]). However, these necessitate the creation of new models and methods for decision-making and control.

As a small step in that direction, we consider a custom manufacturing machine which is a more generic version of a machine tool employed for flexible manufacturing operations (e.g., Uhlmann et al. [2016]) and could use a much broader set of subtractive and/or additive manufacturing operations. The machine is embedded with sensors and analytics to determine and communicate its internal state (e.g., tool condition), gather statuses of various IoT-enabled jobs across the enterprise, and is capable of autonomous operations, especially in terms of making and executing production planning and control decisions.

Traditionally, such custom manufacturing systems have been analyzed either under a completely stochastic framework (with random inter-arrival times and service times of jobs, and random breakdown and repairs of tools) (e.g., Conrad and McClamroch [1987]), or a completely deterministic framework where all information is available at the beginning of a day and a production plan is created (e.g., Niu et al. [2010]). With IoT for custom manufacturing, a middle ground is more appropriate but under a more generic and realistic setting. For example, jobs arrive arbitrarily into the system (not necessarily according to a renewal process) at random times but once they arrive, their processing requirements (or workloads) are revealed (i.e., known deterministically). That is because a job communicates its dimensions, specifications and requirements (i.e., as-is and to-be states) to the machine, and the machine develops a process plan through interactions with databases in a cloud. The plan would include a set of sub-tasks to be performed by the machine and each

sub-task is characterized by the amount of workload. The machine also interacts with the tool to get updates on the remaining life and assesses if another sub-task could be performed or would the tool have to be replaced (see Gao et al. [2015], Liu and Shi [2015]).

We keep our production decision fairly simplistic in that we wish to determine the speed at which the tool operates for each job. We note that the approach is quite general to be applicable to emerging custom manufacturing scenarios with IoT and smart manufacturing machines in that sense. It can be used in processes that involve complex chain of sub-processes, such as multiple steps of machining (e.g., drilling) operations. This scenario, for example, can reflect hole drilling operations specified on custom integral (near-net shape) components employed in aerospace, defense, energy and biomedical implant industry (e.g., integral aircraft or hull panels, oil pumps, hip joints, or artificial hearts). It could also be used in milling, turning and drilling where material is removed. We use the term "speed" as opposed to the more common term MRR (material removal rate) or the deposition rate. The faster the speed is, the faster the job is finished, albeit the tool degrades faster as well. We assume replacing a tool when it degrades (e.g., wears out) completely is time-consuming. Hence it is important to plan both speeds and replacements in advance so that tool replacements are scheduled between jobs, and not when a job is being executed, thereby efficiently operating the machine. For that, the objective considered in this paper is to maximize the effective processing capacity of the machine.

This paper is organized as follows. In Section 2 we describe the problem statement and related work. In Section 3, we introduce notations and modeling details of our problem. We explain analytic approaches to solve the problem and describe the asymptotic properties of the solutions in Section 4. We provide numerical results in Section 5 and state concluding remarks as well as future work in Section 6.

2 Problem Statement and Related Work

As described in Section 1, we consider a smart machine that can execute different custom jobs using a single process type (e.g., drilling of custom panels). A list L of $n(L)$ sub-tasks $L = (w_1, w_2, ..., w_{n(L)})$ are performed on the machine, where w_i is the quantity of work that needs to be performed for sub-task i. We assume that each w_i is a random variable and when the sub-task arrives, the random variable is realized. Let s_i be

the speed for processing sub-task i. The time it takes to complete sub-task i is $\frac{w_i}{s_i}$. Now, let $D(s)$ be the tool degradation rate when it processes a sub-task at speed s. After processing sub-task i, the tool degrades by amount $D(s_i)\frac{w_i}{s_i}$. The decision policy would determine the speed to process each sub-task and replacing the tool ($x_i = 1$) or not ($x_i = 0$) before processing sub-task i. Here we assume that replacing a tool takes τ time units.

The objective is to make a decision about tool replacement x_i and machine speed s_i for all i so that in the long run the total time spent per sub-task on average (this includes tool replacement time, if any, and processing time) is minimized. This would be a good indication of the capacity of the machine. It is important to note the implication of having a replacement time τ as it creates a trade off since high processing speeds would imply too many tool changes and low processing speeds would result in too few parts being processed. While we do not explicitly model job arrival processes, it is important to note that we place no restriction on it either. Jobs can arrive individually or in batches, the inter-arrival times can be correlated (i.e. non IID), and even time-varying. The same is the case with the workloads. This brings a tremendous amount of generality and can be used to conveniently decompose the decisions at each machine. Moreover, we relax the definition of job arrivals. In our study, a job "arrival" does not need to be at the machine. Once the workload of a job is revealed deterministically, we regard this job as an arrival. This is because it would be possible to extract full information about a job upstream to a machine with the help of IoT (see Gao et al. [2015]).

An important aspect of this study is to make online decisions based on revealed information. We consider the entire spectrum of revealed workload all the way from $n(L) = 0$ corresponding to no available workload information to $n(L) \to \infty$ corresponding to the offline case in which full information about workload is known before processing. Then we show that as more information is available, the online algorithms perform better.

Manufacturing control has been discussed for decades and there is a rich literature related to it. Many works such as Martinelli [2007], Hu et al. [1994], Srivatsan and Dallery [1998], Akella and Kumar [1986] and Sharifnia [1988] consider the workload flow controlling problem with continuous workload and objective of minimizing holding (i.e., inventory) cost. Akella and Kumar [1986] modeled a machine with two statuses

(up and down), and Sharifnia [1988] extend this to arbitrary number of machine statuses. Hu et al. [1994] considers the case where machine failure rate depends on processing rate. Discreteness of workload brings randomness in workload processing. Works such as Song [2006], Cheng et al. [2011], Pang [2015], Feng and Xiao [2002], Boukas et al. [1995] and Conrad and McClamroch [1987] consider problems with discrete workload and objective of minimizing processing cost. In these works, workload would not be revealed until the end of processing. The idea of revealing workload upon arrival is shown in online machine scheduling problems such as Hall et al. [1997] and Chekuri et al. [2001], in which job sequence is considered to minimize mean completion times (sojourn times) of jobs. However, speed controlling and machine maintenance are not considered in these models. Gans and Van Ryzin [1997] address a makespan minimization problem in which workload is revealed upon arrival, and machines are reconfigurable so that processing speed can be adjusted by changing different machine configurations. However, the setup times for reconfiguration is assumed to be negligible and tool degradation is not considered in Gans and Van Ryzin [1997]. Offline machine scheduling problems with minimizing makespan are also discussed, such as Sevastianov and Woeginger [1998] and Hall [1998]. In these problems, workload is known deterministically at the beginning. Job arrival and machine status are not considered in these works.

In summary, to the best of our knowledge, there aren't any articles in the open literature that consider the following four aspects in a single framework: **a)** Discrete part manufacturing; **b)** Machine with degrading tools and degradation is a function of processing speed; **c)** Jobs where workload is revealed upon "arrival"; and **d)** Objective of minimizing completion time.

Specifically, while we consider a manufacturing problem that has been discussed for decades, the aforementioned unified framework is novel. The jobs are discrete which is different from the settings in Hu et al. [1994] as well as Srivatsan and Dallery [1998]. We incorparate the phenomenon where high processing speed usually results in fast tool degradation, which is absent in Rabadi et al. [2006], Lee and Lin [2001], Lee and Leon [2001], Cai et al. [2009] andGans and Van Ryzin [1997]. We show that traditional approaches such as Cai et al. [2009], Boukas et al. [1995], Feng and Yan [2000] and Sana et al. [2004] are no longer applicable to our problem as the uncertainty of tool breakage and workload is eliminated in our model. The greedy algorithm used by Conrad and McClamroch [1987] are not as effective as the online algorithms that

we develop. Moreover, we show that knowing the distribution of workload beforehand and revealing workload upon arrival are crucial for decision making. We demonstrate the improvement by knowing workload distribution in Section 5 of this paper. A comparison of relevant literature is provided in Table 1 below.

In addition, we provide a new problem setting where job arrivals do not have to be at the machine to determine the workload. With the advent of IoT it would be possible to reveal workload well before the job physically arrives at a machine. In that way our model is adaptable to futuristic settings as compared to traditional control and scheduling models.

Table 1: Comparison of related literature

Paper	Objective	Job Arrival	Speed Control	Maintenance	Tool Degradation	Workload	Algorithms
Srivatsan and Dallery [1998], Akella and Kumar [1986] Sharifnia [1988], Hu et al. [1994], Martinelli [2007]	Holding Cost	Yes	Yes	Yes	Random	Continuous	Online
Feng and Yan [2000]	Holding Cost	Yes	Yes	Yes	Random	Random	Online
Feng and Xiao [2002], Cheng et al. [2011], Boukas et al. [1995] Song [2006], Pang [2015], Conrad and McClamroch [1987]	Processing Cost	Yes	Yes	Yes	Random	Random	Online
Schweitzer and Seidmann [1991]	Processing Cost	Yes	Yes	No	No	Deterministic	Offline
Rishel [1991]	Processing Cost	No	Yes	Yes	Random	No	Online
Hall et al. [1997], Chekuri et al. [2001]	Sojourn time	Yes	No	No	No	Revealed deterministically upon arrival	Offline&Online
Sevastianov and Woeginger [1998], Hall [1998]	Makespan	No	Yes	No	No	Deterministic	Offline
Rabadi et al. [2006]	Makespan	No	No	Yes	No	Deterministic	Offline
Cai et al. [2009]	Makespan	No	No	Yes	Random	Random	Online
Gans and Van Ryzin [1997]	Makespan	Yes	Yes	No	No	Revealed Deterministically upon arrival	Online
Our Paper	Makespan	Yes	Yes	Yes	Deterministic	Revealed deterministically upon "arrival"	Offline&Online

3 Model and Notations

Assume workload of job i is upper bounded by w_u. Notice that in this paper we characterize tool degradation by a physics-based deterministic model and we introduce remaining tool level as the measure of its remaining processing capacity. For example, in drilling and other material removal processes, the remaining level of a tool is the amount of wear, measured in terms of length of the wear h it can sustain before it needs to be replaced. We assume all the *new* tools are of the same level h^* and their wear is governed by the same degradation function $D(s)$. For convenience, we provide a list for notations, which is shown in Table 2. When job i is about to be processed, it observes the remaining tool level h_i at that time. After the tool processes sub-task i at speed s for time $\widehat{t}$, its remaining level is $h_i - D(s)\widehat{t}$. In our paper, we mainly use Taylor's formula to characterize tool degradation (see Stephenson and Agapiou [2016]), since it perfectly describes the relation of tool degradation and processing speed for jobs. Taylor's formula is given by $sT^n = C^*$, where

T is the tool life (i.e., the duration of the time the tool can be used in the specified processing condition), $n \in (0,1)$ and C^* are constants which are experimentally determined. For convenience, we let $\nu = \frac{1}{n}$ (hence $\nu > 1$), and Taylor's formula can be written as

$$s^\nu T = C, \quad \nu > 1 \tag{1}$$

where $C = (C^*)^\nu$. When processing with speed s, a tool of remaining level h^* is of lifetime T_s, thus the degradation rate is given by

$$D(s) = \frac{h^*}{T_s} = \frac{s^\nu}{C} h^*, \quad \nu > 1. \tag{2}$$

Assume changing of speed and replacing the tool are not allowed during processing (Xu and Cao [2015]), thus for job i of workload w_i, it takes $\frac{w_i}{s_i}$ to finish processing. Therefore h_i can be updated by $h_{i+1} = h_i - D(s_i)\frac{w_i}{s_i}$. Since tool level cannot be negative, we have for each job i: $h_i - D(s_i)\frac{w_i}{s_i} > 0$. When a tool degrades to a status in which $h_i = 0$, it has to be replaced by a new tool and the new tool will be used to process job i, then we have $h_i = h^*$. Note that we also allow replacing a tool even if has not reached the final status. Since $x_i = \{0,1\}$ is a binary variable denoting replacing decision before processing job i, we can write $h_i \triangleq h^* x_i + (1 - x_i)\left(h_{i-1} - D(s_{i-1})\frac{w_{i-1}}{s_{i-1}}\right)$ where $x_i = 1$ indicates replacing the tool before job i and $x_i = 0$ for not.

Note that the workload of a job is only revealed when the IoT system gathers sufficient information about it. We first describe an offline problem in which we assume that all the information about jobs in the future is obtained before processing, the objective is to find the optimal decision $\mathbf{s} = \{s_1, s_2, ..., s_{n(L)}\}$ and $\mathbf{x} = \{x_1, x_2, ..., x_{n(L)}\}$ to minimize makespan. The **Offline Problem** (**OF**) can be described as follows:

Table 2: Notations Used

Parameters
w_i: Revealed workload of job i
$w_b = \left((\nu-1)\tau C^{\frac{1}{\nu-1}}\right)^{\frac{\nu-1}{\nu}}$: Optimal tool level
w_u: Upper bound for the workload w_i
w_c: Tool/buffer capacity, the maximal workload a tool/buffer can serve/hold
W_i: Random variable denoting workload of future job i before revealing
h_i: Remaining tool level before processing job i
h^*: Tool level for a new tool
T_s: Tool life time for a new tool with processing speed s
τ: Replacement time for one tool
$\nu = \frac{1}{n}$, C: constants experimentally determined for Taylor's formula
$D(s)$: Degradation rate for processing speed s
η_j: Processed/packed workload by tool/buffer j
Variables
s_i: Processing speed for job i
x_i: Decision variable for tool replacement before processing job i
Problems and Approaches
OF: Offline Problem
OFR: Offline Problem with Rearranging
BC: Best Case Problem
BC*: Modified Best Case Problem
FS: Fixed Speed Approach
FB: Fixed Buffer Approach
SOP: Stochastic Optimal Policy
SA: Simple Approach

Problem 1 (**Offline Problem**)**.**

$$T_{OF}(L) = \min_{\boldsymbol{x},\boldsymbol{s}} \frac{1}{n(L)} \sum_{i=1}^{n(L)} \left(\frac{w_i}{s_i} + \tau x_i \right) \tag{3}$$

$$s.t. \quad h_{i+1} \triangleq h^* x_{i+1} + (1 - x_{i+1}) \left(h_i - D(s_i) \frac{w_i}{s_i} \right), \tag{4}$$

$$h_i - D(s_i) \frac{w_i}{s_i} \geq 0, \tag{5}$$

$$s_i > 0, \tag{6}$$

$$x_i = \{0, 1\} \ for \ i = 1, 2, ..., n(L). \tag{7}$$

The objective function (3) is to minimize the average cycle time of each job that consists of processing time and tool replacement time. The first constraint (4) is the update function, showing that before processing job i, the remaining tool level is either h^* if replacement happened or the remaining level $h_{i-1} - D(s_{i-1})\frac{w_{i-1}}{s_{i-1}}$ if replacement did not happen. Constraint (5) indicates that tool level cannot be negative. Our decision variable speed s_i should be positive, and replacing decision variable x_i, which is binary, are shown in constraints (6)

and (7).

If we also consider the order of service, we have the **Offline Problem with Rearranging** as **Problem 2**, where $R(L)$ is the set of lists in which each list is made by rearranging the order of list L. Any list in $R(L)$ is given by $\{w_{(1)}, w_{(2)}, ..., w_{(n(L))}\}$, which is a permutation of list $\{w_1, w_2, ..., w_{n(L)}\}$.

Problem 2 (**Offline Problem with Rearranging (OFR)**).

$$
\begin{aligned}
& T_{OFR}(L) \triangleq \min_{L^0 \in R(L)} \frac{1}{n(L^0)} \sum_{i=1}^{n(L^0)} \left(\frac{w_{(i)}}{s_i} + \tau x_i \right) \\
& s.t.\ h_{i+1} \triangleq h^* x_{i+1} + (1 - x_{i+1}) \left(h_i - D(s_i) \frac{w_{(i)}}{s_i} \right), \\
& h_i - D(s_i) \frac{w_{(i)}}{s_i} \geq 0, \\
& s_i > 0,\ x_i = \{0, 1\},\ for\ i = 1, ..., n(L^0).
\end{aligned}
\tag{8}
$$

Notice we can also write **OFR** as

$$
T_{OFR}(L) \triangleq \min_{L^0 \in R(L)} T_{OF}(L^0) \tag{9}
$$

The permutation set $R(L)$ has $n(L)!$ elements. We want to choose the best order from $R(L)$, say L^*, whose optimal value of L^* is less than that of any other lists in set $R(L)$. When job list L has finite number of sub-tasks, then we have $L^* = arg \min_{L^0 \in R(L)} T_{OF}(L^0)$ such that $T_{OFR}(L) = T_{OF}(L^*)$. If we add a constraint on the **Offline Problem with Rearranging** that processing speed is a constant, the modified problem is to find the minimal number of tool replacements, which is a bin packing problem and known to be NP-hard (see Korte et al. [2012]).

We select makespan as objective function because it is a good indication for machine capacity. It allows for general arrival processes (i.e, correlated, time varying or in batches). Besides, arrival does not need to be at the machine. As long as a job has revealed its workload, we regard it as an arrival. Moreover, if there is a cost rate for machine in operation, say K, and cost for tool replacement, say R, then the makespan minimization problem can be written as a cost minimization problem. Notice that since we do not allow tool breakage during processing, every job will be processed successfully. The **Cost Minimization Problem**

(CM) can be written as follows:

$$\begin{aligned}
\textbf{Cost Minimization}: \quad & \min \sum_{i=1}^{n(L)} \left(K\frac{w_i}{s_i} + Rx_i \right). \\
s.t. \qquad & h_{i+1} \triangleq h^* x_{i+1} + (1 - x_{i+1}) \left(h_i - D(s_i)\frac{w_i}{s_i} \right), \\
& h_i - D(s_i)\frac{w_i}{s_i} \geq 0, \\
& s_i > 0, \ x_i = \{0, 1\} \ for \ i = 1, 2, ..., n(L).
\end{aligned}$$

The setting of this problem relaxes the one investigated in Conrad and McClamroch [1987] by considering that workload is revealed on upon arrival. We will show in **Section 5** of the paper that if we use the greedy approach in Conrad and McClamroch [1987], it would result a much weaker solution than ours. Notice **CM** is different from **OF** only in the objective function. If $\tau = \frac{R}{K}$, then these two problems are identical.

4 Online Algorithms

Since the workload of a job is not revealed until sufficient information is gathered, to solve **OF** we must look through all $n(L)$ jobs. A scenario we observe is that once new jobs are revealed we can update our information about jobs dynamically. As more information becomes available, better decisions can be made. Thus our approaches will focus on how to deal with the information updating process. When future arrivals have not been revealed, we make decisions using online algorithms. We use a behavior-imitating way to construct the online algorithms and we want the online algorithms to behave similarly in some aspects as the optimal policy. We first characterize some useful properties that our algorithms can imitate. Also we provide a lower bound of **OF** that can be used to characterize the performance of approaches that we propose. After that we provide an approach where we process jobs with a fixed speed, finally we introduce an approach which solves **OF** part by part by choosing the size of sub-problems dynamically and show that rearranging is unnecessary for online algorithms.

4.1 Properties of the Offline Problem

The main idea of constructing the online algorithms is to find some unique properties of **OF** that online algorithms can imitate. So in this subsection we mainly discuss those properties of **OF**. We first have Theorem 1 in the following to characterize the optimal solution to **OF**. Note that in Theorem 1 we assume $\frac{D(s)}{s}$ is convex and increasing in s. This is true when using Taylor's formula (Equation (1)) to describe the degradation process, in which case degradation rate is given by $\frac{s^\nu}{C}h^*$, as shown in Equation (2). Since $\nu > 1$ in Taylor's formula, we have $\frac{D(s)}{s} = \frac{s^{\nu-1}}{C}h^*$ to be convex and increasing when $s > 0$. However Theorem 1 is not confined to using Taylor's formula to define tool degradation rate function $D(s)$.

Theorem 1. *On solving **OF**, if $D(s)$ is the degradation rate such that $\frac{D(s)}{s}$ is convex and increasing when $s > 0$ with $\frac{D(s)}{s}|_{s=0} > 0$, then the optimal processing speeds for jobs processed by the same tool are identical, i.e. $s_i = s_j$ if job i and j are processed by the same tool.*

Proof. We know that the optimal solution is a series of x_i's which are either 1 or 0. Without loss of generality, assume that $h_1 = h^*$ thus $x_1 = 1$. Say $x_2 = x_3 = \cdots = x_{m-1} = 0$ and $x_m = 1$. Thus there are m jobs served by this tool, and total time working on these m jobs is $\sum_{i=1}^{m}\left(\frac{w_i}{s_i}\right) + \tau$. Since the same tool is used and no replacement happens during processing, tool level cannot be negative in the end. Suppose ϵ of tool level remains when finishing these m jobs. From the constraint (5) and (6), we have $\sum_{i=1}^{m}\frac{w_i}{s_i}D(s_i) = h^* - \epsilon$. We want to prove that processing time is minimal when $s_1 = s_2 = ... = s_m$.

First consider the following sub-problem with only the first m jobs of L. Notice its constraints are specified in terms of the update function of Equation (4) and no replacement happens during processing of these m jobs. Hence

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m}\left(\frac{w_i}{s_i}\right) + \tau \\
s.t. \quad & h_2 - h^* + \frac{w_1}{s_1}D(s_1) = 0, \\
& \vdots \\
& h_m - h_{m-1} + \frac{w_{m-1}}{s_{m-1}}D(s_{m-1}) = 0, \ \textit{and} \\
& \epsilon - h_m + \frac{w_m}{s_m}D(s_m) = 0.
\end{aligned}$$

Write the constraints of above problem as $\mathbf{g}(s) = (g_1(s), g_2(s), ..., g_m(s))^T = 0$, where $g_i(s)$ is the LHS of the ith constraint above. The Lagrange function of above problem can be written as

$$\begin{aligned}\mathcal{L}(s,\lambda) &= \sum_{i=1}^{m}\left(\frac{w_i}{s_i}\right) + \tau - \lambda_1(h_2 - h_1 + \frac{w_1}{s_1}D(s_1)) - ... - \lambda_{m-1}(h_m - h_{m-1} + \frac{w_{m-1}}{s_{m-1}}D(s_{m-1})) \\ &\quad -\lambda_m(\epsilon - h_m + \frac{w_m}{s_m}D(s_m)).\end{aligned}$$

From the first order condition of the Lagrange function we have $-\frac{w_i}{s_i^2} - \lambda_i \frac{\partial}{\partial s_i}\left(\frac{w_i}{s_i}D(s_i)\right) = 0$ for all $i \in \{1, ..., m\}$. It is easy to show that the point $s^* = \{s_1^*, s_2^*, ..., s_m^*\}$ which satisfies $s_1^* = s_2^* = ... = s_m^*$ is a stationary point. Thus we know that $Z(s^*) = \{w_2, -w_1, 0, ..., 0\}$ is a null space of $\nabla \mathbf{g}(s^*) = 0$. From the first order condition we have $\lambda_i = \frac{-w_i}{\frac{\partial}{\partial s_i}(\frac{D(s_i)}{s_i})s_i^2} \mid_{s_i = s_i^*}$. Note that $\lambda^* = (\lambda_1^*, ..., \lambda_m^*)$, then we know $\nabla_{ss}\mathcal{L}(s^*, \lambda^*)$ is a diagonal matrix with the ith diagonal element $\left(\frac{2w_i}{s_i^3} + \frac{w_i \frac{\partial}{\partial s_i^2}(\frac{D(s_i)}{s_i})}{\frac{\partial}{\partial s_i}(\frac{D(s_i)}{s_i})s_i^2}\right)\Big|_{s_i = s_i^*}$. Since $\frac{D(s)}{s}$ is increasing, we have $Z(s^*)^T \nabla_{ss}\mathcal{L}(s^*, \lambda^*) Z(s^*)$ is positive definite. Since we know s_i's being identical is a solution of above Lagrange function, from convexity this solution is optimal (see Griva et al. [2009]). □

Corollary 1.1. *One solution to **OF** is such that when the tool is about to be replaced, its remaining level is zero. That is, when $x_{i+1} = 1$, we have that $h_i - D(s_i)\frac{w_i}{s_i} = 0$.*

Proof. Suppose m jobs are processed by tool j and ϵ tool level is left when tool j is about to be replaced. Then we have that $\sum_{i=1}^{m} \frac{w_i}{s_i} D(s_i) = h^* - \epsilon$, and time between replacements $\sum_{i=1}^{m}\left(\frac{w_i}{s_i}\right) + \tau$, which is the portion corresponding to this tool in (3). From **Theorem 1** we see that in the optimal solution, tool j processes its jobs with a single speed, say s_j^*. If $\widehat{w} = \sum_{i=1}^{m} w_i$, we have the time between two replacements equal to $\frac{\widehat{w}}{s_j^*} + \tau$, with the constraint $\widehat{w}\frac{D(s_j^*)}{s_j^*} = h^* - \epsilon$. Since $\frac{D(s)}{s}$ is increasing with respect to s, then the solution of $\widehat{w}\frac{D(s_0)}{s_0} = h^*$, say s_0, is greater than s_j^*. Then, since $\frac{\widehat{w}}{s_0} < \frac{\widehat{w}}{s_j^*}$, we can see $\epsilon = 0$ will result in minimal time between replacements. □

So far we have seen that in the optimal solution of **OF**, jobs served by the same tool have identical processing speed. We want our online algorithms to imitate this property. However, to see how good it would be to fix the processing speed, we need a benchmark to compare our algorithms against. For that, we solve another system which serves as the lower bound of **OF**. If we find an algorithm that gets a result

larger than but close to this lower bound, we can assure that this result is close to what **OF** gets. Since we know $T_{OFR} \leq T_{OF}$, then we can use the optimal solution of **OFR** as the lower bound of **OF**. We introduce the following problem to get the solution of **OFR** analytically.

Problem 3 (**Best Case Problem (BC)**)**.**

$$\begin{aligned} & T_{BC}(L) \triangleq \min \frac{1}{n(L)} \sum_{i=1}^{n(L)} \left(\frac{w_i}{s_i} + \tau x_i \right) \\ s.t. \quad & \sum_{i=1}^{n(L)} w_i \frac{D(s_i)}{s_i} \leq \sum_{i=1}^{n(L)} x_i h^* \\ & s_i > 0, \ x_i = \{0, 1\}, for \ i = 1, ..., n(L). \end{aligned} \tag{10}$$

Notice the **BC** differs from **OF** only in the constraints. **BC** provides the number of tools needed for serving a given total workload, without considering feasibility.

Lemma 2. *For the same job list L,* ***BC*** *always gets optimal value no larger than* ***OFR****.*

Proof. We only need to show for any order $L^0 \in R(L)$, $T_{BC}(L^0) \leq T_{OF}(L^0)$. Since the objective functions of both **BC** and **OF** are the same, we only need to prove the feasible region of the **OF** is contained in that of **BC**. For any feasible solution of **OF**, $\mathbf{x} = (x_1, x_2, ...)$ is made of a series of ones and zeros. Suppose x_{l_m} and $x_{l_{m+1}}$ are two adjacent 1 in the solution, then $\sum_{i=l_m}^{l_{m+1}-1} w_i \frac{D(s_i)}{s_i} \leq h^*$. Thus $\sum_{i=1}^{n(L)} w_i \frac{D(s_i)}{s_i} \leq h^* \sum_{i=1}^{n(L)} x_i$, we can get inequality (10). Then its feasible region is contained in that of **OF** for any list L. From Equation (9) we get $T_{BC}(L) \leq T_{OFR}(L)$. □

When the degradation rate is given by Equation (2), the next lemma gives a bound for $T_{BC}(L)$.

Lemma 3. *For* ***BC*** *with finite $n(L)$, when $D(s) = h^* \frac{s^\nu}{C}$, if $w_b = \left((\nu - 1) \tau C^{\frac{1}{\nu - 1}} \right)^{\frac{\nu - 1}{\nu}}$ and $w_i \leq w_b$, given that $\widehat{w} = \sum_{i=1}^{n(L)} w_i$, then its optimal solution is given by $s = \left(\frac{Cy^*}{\widehat{w}} \right)^{\frac{1}{\nu - 1}}$, $y^* = \sum_{i=1}^{n(L)} x_i^* = \lfloor \frac{\widehat{w}}{w_b} \rfloor$ or $\lceil \frac{\widehat{w}}{w_b} \rceil$. We have either $\frac{\tau^{\frac{1}{\nu}} (\nu - 1)^{\frac{1}{\nu} - 1} \nu \widehat{w}}{C^{\frac{1}{\nu}} n(L)} \leq T_{BC}(L) \leq \frac{\lfloor \frac{\widehat{w}}{w_b} \rfloor \tau \nu}{n(L)}$ or $\frac{\tau^{\frac{1}{\nu}} (\nu - 1)^{\frac{1}{\nu} - 1} \nu \widehat{w}}{C^{\frac{1}{\nu}} n(L)} \leq T_{BC}(L) \leq \frac{\lceil \frac{\widehat{w}}{w_b} \rceil \tau \nu}{n(L)}$. Also, if $\frac{\widehat{w}}{w_b}$ is an integer, then $y^* = \frac{\widehat{w}}{w_b}$.*

Proof. Let $y = \sum_{i=1}^{n(L)} x_i$. We first show that the optimal solution always guarantees that all s_i are identical,

say $s_i = s_1^*$, and $\sum_{i=1}^{n(L)} w_i \frac{s_1^{*\nu-1}}{C} = y$. The Lagrange function of **BC** is

$$\mathcal{L}(s, x, \lambda, \mu) = \sum_{i=1}^{n(L)} \left(\frac{w_i}{s_i} + \tau x_i \right) - \lambda \sum_{i=1}^{n(L)} \left(w_i \frac{s_i^{\nu-1}}{C} - x_i \right).$$

By first and second order conditions we have that $s_i = s_1^*$ (for $\forall i$) is the optimal solution. From the constraint $\frac{\widehat{w} s^{\nu-1}}{C} = y$, we have that $s = (\frac{Cy}{\widehat{w}})^{\frac{1}{\nu-1}}$. Then **BC** can by written as:

$$\min_{y \in \mathbb{Z}_+} \qquad \frac{\widehat{w}}{(\frac{Cy}{\widehat{w}})^{\frac{1}{\nu-1}}} + y$$

Then the results in the theorem follow. □

Notice that **Lemma 2** and **Lemma 3** hold for any lists L. From **Lemma 3** we have if $\frac{\widehat{w}}{w_b}$ is an integer, then exactly $\frac{\widehat{w}}{w_b}$ tools are used for processing $\widehat{w}$ amount of workload, with each tool processing w_b amount of workload. In this case w_b gives the optimal workload a tool processes in **OFR** based only on tool parameters C, ν and τ. If we only have one job whose workload is w_b, both **OFR** and **BC** have the same optimal value. If we are able to rearrange a job list so that each tool processes exact w_b amount of workload, then **BC** and **OFR** gets the same optimal makespan for this list. But on what condition can we guarantee the existence of such a list?

Notice that if we add a constraint that specifies a maximal amount of workload a tool can process, say w_c, then this constrained **OFR** can be described as follows: we want to allocate jobs to tools, with each tool getting w_c workload so that the total makespan is minimized. This idea is quite similar to the bin packing problem (see Karmarkar [1982]). In the bin packing context, we have several pieces to be packed into a few identical bins. In the one dimensional bin packing problem, we have several pieces with different lengths and bins of identical capacity/length. We want to pack pieces with minimal number of bins. In our paper, we can regard each tool as a bin with a specific capacity, where the capacity is given by the maximal workload it can process. We also regard jobs as pieces to be packed by bins, where the length of each piece is given by the workload. We will discuss how to find the optimal w_c later on, but once we fix w_c, the problem now is to allocate jobs of L to tools so as to use the least number of tools. As we can see, if all the tools are

fully "packed" by jobs, which means each tool processes exact w_c amount of workload, we have the minimal number of tool replacement. Suppose $k_{opt}(L)$ is the minimal number of tools we use to process jobs in list L with rearranging under a fixed speed s^* where s^* is the solution of $w_c\frac{D(s^*)}{s^*} = h^*$. We say perfect packing happens when $w(L) = k_{opt}w_c$. We have the following theorem to show that **OFR** is asymptotically close to **BC** in a special case.

Theorem 4. *If workload w_i is drawn independently from a uniform distribution within $[0, w_u]$ and $w_u \leq w_b$, then as $n(L) \to \infty$, $\dfrac{\boldsymbol{E}[T_{OFR}(L)]}{\boldsymbol{E}[T_{BC}(L)]} \to 1$.*

Proof. Suppose we have a finite job list L. We fix $w_c = w_b$ and process all the jobs at speed s^*, where s^* is the solution of $w_b\frac{D(s^*)}{s^*} = h^*$. Once the speed is fixed, we only need to allocate jobs of L into tools whose capacities are w_b. In the end we find the minimal number of tools we used is $k_{opt}(L)$, in which $k_0(L)$ tools are fully packed. Let $k_1(L) = k_{opt}(L) - k_0(L)$. We know $k_0(L) = 0$ if no tool is fully packed. From **Lemma 2** we have $\dfrac{\mathbf{E}[k_0(L)]w_b}{s^*} + \mathbf{E}[k_0(L)]\tau \leq n(L)\mathbf{E}[T_{BC}(L)] \leq n(L)\mathbf{E}[T_{OFR}(L)] \leq \dfrac{\mathbf{E}[k_0(L)]w_b}{s^*} + \mathbf{E}[k_0(L)]\tau + \mathbf{E}[k_1(L)]\dfrac{w_b}{s^*} + \mathbf{E}[k_1(L)]\tau$. The last equality holds because rearranging guarantees the optimal decision among all the possible permutations. Thus

$$\frac{\mathbf{E}[T_{OFR}(L)]}{\mathbf{E}[T_{BC}(L)]} \leq 1 + \frac{\mathbf{E}[k_1(L)]\dfrac{w_b}{s^*} + \mathbf{E}[k_1(L)]\tau}{\dfrac{\mathbf{E}[k_0(L)]w_b}{s^*} + \mathbf{E}[k_0(L)]\tau} = 1 + \frac{\mathbf{E}[k_1(L)]}{\mathbf{E}[k_0(L)]}$$

We now prove that $\lim_{n(L)\to\infty} \dfrac{\mathbf{E}[k_1(L)]}{\mathbf{E}[k_0(L)]} = 0$ by contradiction. From Karmarkar [1982] we know that once w_i is uniformly distributed within $[0, w_u]$, we have that $\lim_{n(L)\to\infty} \dfrac{\mathbf{E}[w(L)]}{\mathbf{E}[k_{opt}(L)w_b]} = 1$. Suppose that $\lim_{n(L)\to\infty} \dfrac{\mathbf{E}[k_1(L)]}{\mathbf{E}[k_0(L)]} = \epsilon$, then

$$\frac{\mathbf{E}[w(L)]}{\mathbf{E}[k_{opt}(L)w_b]} = \frac{w_b\mathbf{E}[k_0(L)] + \mathbf{E}[w(L) - k_0(L)w_b]}{\mathbf{E}[k_{opt}(L)w_b]} \leq \frac{w_b\mathbf{E}[k_0(L)] + \mathbf{E}[k_1(L)w_b]}{\mathbf{E}[k_{opt}(L)w_b]} = \frac{w_b\mathbf{E}[k_0(L)] + \mathbf{E}[k_1(L)w_b]}{w_b\mathbf{E}[k_0(L) + k_1(L)]}.$$

The second inequality is an equality only when $k_1(L) = 0$, otherwise $\dfrac{\mathbf{E}[w(L)]}{\mathbf{E}[k_{opt}(L)w_b]} < \dfrac{w_b\mathbf{E}[k_0(L)] + \mathbf{E}[k_1(L)w_b]}{w_b\mathbf{E}[k_0(L) + k_1(L)]}$, take the limit of both sides we have that $1 < 1$, getting a contradiction. Hence proved. □

So far we have acquired the asymptotic optimal value of **OFR** which can serve as the lower bound of **OF**. Also we have a benchmark that we can compare our online algorithms against. Now we discuss different

ways to construct online algorithms by imitating the same speed property (**Theorem 1**) of **OFR**.

4.2 Fixed Speed Approach

When we have only one revealed workload in system, we have little information to help us make decisions. Using up one tool to process this job is not a good idea since replacement also takes time. We shall see the performance of this policy later in Section 5. We still want to keep some remaining tool level to process jobs in future. In this section we will discuss an online approach which relies on one revealed workload and keeps a reasonable amount of tool life for jobs in future.

As we observe from **OFR**, jobs are allocated to tools in a way which is similar to the bin packing problem. In the bin packing context, we have a policy called Next-Fit (Coffman et al. [1980]). In one dimensional space, assume pieces each of a specific *length* come in a sequence, say $i = 1, 2, 3, ...$ We have infinite number of identical bins whose capacity is c and pieces are packed into bins one by one. We define the *level* of a bin by the sum length of all the pieces in that bin. The Next-Fit tells us that if the length of piece i plus the current bin level is greater than c, we finish our current bin and pack piece i into a new bin. If we index our bin by $j = 1, 2, ...$, the Next-Fit can be described as: If $length_i + level_j < c$, choose bin $j + 1$ for piece i, otherwise choose bin $j + 1$.

We find that it may be a good idea to serve jobs in a Next-Fit way. In our problem, if we fix a processing speed for every job, say s^*, from constraint of $w\frac{D(s^*)}{s^*} = h^*$ we know that each tool can process at most $\frac{h^* s^*}{D(s^*)}$ amount of workload. We let $w_c = \frac{h^* s^*}{D(s^*)}$ be the tool capacity, which is the maximal workload a tool can process. Similarly, we regard each tool as a "bin" which we describe above by defining η_j as the workload tool j has already processed. The remaining tool level then becomes $w_c - \eta_j$ because each tool can process at most w_c amount of workload. Therefore, if we process jobs in a Next-Fit way, when $w_i + \eta_j \leq w_c$ for job i and tool j, we process job i with tool j at speed s^*, otherwise we process job i with tool $j + 1$ at speed s^*.

So the problem is how to find s^*. The speed should not be slow, otherwise the processing may be sluggish hence impacting the objective. On the other hand, from constraint $w_c\frac{D(s^*)}{s^*} = h^*$ and the monotonicity of $\frac{D(s)}{s}$, fast speed always results in a small w_c. If we assume workload of each job is bounded by w_u, then w_c should not be less than w_u, otherwise we would not be able to process jobs with workload w_u at speed s^*.

Suppose w_c^* is the optimal tool capacity we fix to determine s^*, we call this the **Fixed Speed Approach** (**FS**). Denote $T_{FS}(L)$ as the processing time for list L under **FS**, and **FS** can be described by **Algorithm 1**.

Algorithm 1 Fixed Speed Approach

Speed s^* is determined by $w_c\frac{D(s^*)}{s^*} = h^*$.
for i=1:$n(L)$ **do**
  if $h_i - D(s^*)\frac{w_i}{s^*} > 0$ **then** Keep the current tool and process job i at speed s^*;
  else Replace the tool and then process job i at speed s^* using the new tool;
  end if
  if i==$n(L)$ **then** Replace the tool;
  end if
end for

Theorem 5. *If processing speed is fixed at* $s^* = \left(\frac{C}{w_c^*}\right)^{\frac{1}{\nu-1}}$ *and* $w_c^* \geq w_u$ *where* w_c^* *is the optimal tool capacity, then* $\frac{\boldsymbol{E}[T_{BC}(L)]}{\boldsymbol{E}[T_{FS}(L)]} \in [\frac{(1-\beta)\nu}{(\nu-1)(1-\beta)+1}, 1]$, *where* $1-\beta = \frac{\boldsymbol{E}[\eta_i]}{w_c^*}$, *as* $n(L) \to \infty$.

Proof. We first show the result by fixing $w_c = w_b$, i.e. $w_c = \left(\tau(\nu-1)C^{\frac{1}{\nu-1}}\right)^{\frac{\nu-1}{\nu}}$, though we know there should be a better choice for w_c. In this case, the speed is fixed at $s^0 = \left(\frac{C}{w_b}\right)^{\frac{1}{\nu-1}}$. Suppose under **FS**, k tools have been used to process jobs in list L and tool $(k+1)$ is just about to be put into use for processing a new job outside of L and this job cannot be processed by tool k under speed s^0. At this point, $n(L)T_{FS}^0(L) = \sum_{i=1}^k \frac{\eta_i}{s^*} + k\tau$. Workload served by these k tools is given by $w(k) = \sum_{i=1}^k \eta_i$. When $w(k)$ workload is done under **BC**, from **Theorem 3** we know that

$$\frac{\tau^{\frac{1}{\nu}}(\nu-1)^{\frac{1}{\nu}-1}\nu\widehat{w}}{C^{\frac{1}{\nu}}n(L)} \leq T_{BC}(L) \leq \frac{(\frac{\widehat{w}}{w_b}+1)\tau\nu}{n(L)}.$$

From Coffman et al. [1980] we know that $\{\eta_i\}$ is a continuous Markov Chain, so we know that when $k \to \infty$ we have $\lim_{k\to\infty}\frac{\mathbf{E}[w(k)]}{k} = \mathbf{E}[\eta_i] = (1-\beta)w_b$. Suppose w_0 is the first workload served by the last tool, then

$$\begin{aligned}\lim_{n(L)\to\infty}\frac{\mathbf{E}[T_{BC}(L)]}{\mathbf{E}[T_{FS}^0(L)]} &\geq \lim_{n(L)\to\infty}\frac{\tau^{\frac{1}{\nu}}(\nu-1)^{\frac{1}{\nu}-1}\nu\mathbf{E}[w(k)]}{C^{\frac{1}{\nu}}\mathbf{E}[\sum_{i=1}^{k-1}\frac{\eta_i}{s^*}+\frac{w_b}{s^*}+k\tau]} \\ &\geq \lim_{k\to\infty}\frac{\tau^{\frac{1}{\nu}}(\nu-1)^{\frac{1}{\nu}-1}\nu\mathbf{E}[\sum_{i=1}^{k-1}\eta_i+w_0]}{C^{\frac{1}{\nu}}\mathbf{E}[\sum_{i=1}^{k-1}\frac{\eta_i}{s^*}+\frac{w_b}{s^*}+k\tau]} = \frac{(1-\beta)\nu}{(1-\beta)(\nu-1)+1}.\end{aligned}$$

Now, to show $\lim_{n(L)\to\infty} \frac{\mathbf{E}[T_{BC}(L)]}{\mathbf{E}[T^0_{FS}(L)]} \leq \frac{(1-\beta)\nu}{(1-\beta)(\nu-1)+1}$, we have

$$\lim_{n(L)\to\infty} \frac{\mathbf{E}[T_{BC}(L)]}{\mathbf{E}[T^0_{FS}(L)]} \quad \leq \quad \lim_{n(L)\to\infty} \frac{\mathbf{E}[\frac{w(k)}{w_b}+1]\tau\nu}{\mathbf{E}[\sum_{i=1}^{k-1}\frac{\eta_i}{s^*}+\frac{w_0}{s^*}+k\tau]} = \lim_{k\to\infty} \frac{(\mathbf{E}[\sum_{i=1}^{k-1}\frac{\eta_i}{w_b}+1)\tau\nu]}{\mathbf{E}[\sum_{i=1}^{k-1}\frac{\eta_i}{s^*}+\frac{w_0}{s^*}+k\tau]} = \frac{(1-\beta)\nu}{(1-\beta)(\nu-1)+1}.$$

We know we can definitely find other better options for w_c, thus proving the theorem. □

Corollary 5.1. *If $w_u = w_b$ and $w_c = w_b$, which means w_i is drawn from $U(0, w_b)$ and tool capacity is also w_b, we have that* $\lim_{n(L)\to\infty} \frac{\boldsymbol{E}[T_{BC}(L)]}{\boldsymbol{E}[T_{FS}(L)]} = \frac{3\nu}{3\nu+1}$.

Proof. From Coffman et al. [1980] we know that $\mathbf{E}[\eta] = \frac{3}{4}w_u$, and by **Theorem 5** we have the result. □

Notice in **FS**, w_c is not necessarily equal to w_b. That is to say, w_b is not the optimal option to fix speed by solving $w_c\frac{D(s)}{s} = h^*$. Here we did not give the actual optimal w_c. Although the cdf of bin level is given by Karmarkar [1982], the expectation of bin level is difficult to characterize analytically. Numerical tests are provided in the Section 5.

If we do not reveal the workload upon arrival and $w_u = w_b$, the optimal policy is to process each job by exhausting one tool. We call this policy the **Stochastic Optimal Policy (SOP)**. The following corollary provides the performance of this policy.

Corollary 5.2. *If $w_u = w_b$ and no workload is revealed upon arrival, the optimal speed is determined by $w_b\frac{D(s)}{s} = h^*$. Then we have* $\lim_{n(L)\to\infty} \frac{\boldsymbol{E}[T_{BC}(L)]}{\boldsymbol{E}[T_{SOP}]} = \frac{\nu}{\nu+1}$.

Intuitively we can guess that **FS** performs better than **SOP** since **FS** has more information (**FS** reveals one workload and **SOP** reveals none). We will compare the performance of **FS** and **SOP** numerically in Section 5. **FS** is simple as it does not require much computation or storage space. It does not rely on workload distribution either, since w_c can be preassigned with only information of w_u (notice that $w_c \geq w_u$). However, as stated before, we have to fix w_c such that it is no smaller than w_u, the upper bound of workload of each job. In the case when w_u is large, w_c must be larger than w_b. Intuitively when w_c is much larger than w_b, **FS** would not be promising. We will show numerical results in Section 5. Another drawback of this method is that tools sometimes are discarded though they have not been used up. They are replaced just because the new job has larger workload than the tool can handle under a fixed speed. This implies that

FS can be definitely improved when there are more than one job in system, so we introduce an algorithm which makes full use of every tool in Section 4.3.

4.3 Fixed Buffer Approach

In the case where more than one workload is revealed, we may consider another way to better imitate the optimal solution to **OF**. Notice that in **FS** we fix a speed for all jobs, however the solution of **OF** only tells us to fix the speed for the same tool. In other words, once we know how much workload a tool processes, we can then fix the speed accordingly. This is the idea for **Fixed Buffer Approach (FB)** that we will provide in this subsection.

4.3.1 Bounded Workload

We first show the performance of **FB** in the case when w_u is bounded. The idea of **FB** in the bounded workload case is very similar to **FS** that we discuss in Section 4.2. To see this, we suppose there is a sequence of virtual buffers in front of our machine. Jobs entering the system are not processed instantly one by one, but put into the first buffer. We start processing jobs in this buffer when it is unable to accept more jobs, and put new arrivals into the next buffer, the same as Next-Fit described in Section 4.2. Buffers are indexed by $j = 1, 2, ...$ We abuse our notation by defining the capacity of a buffer to be w_c, which is the maximal amount of workload a buffer could hold. Note this is different from the maximal workload that a tool can process that we defined in Section 4.1 and 4.2. Here we mainly use w_c to denote a threshold because want **FB** to be applied in a more generic case (Section 4.3.2) in which workload may be unbounded. Again, we use η_j to denote the *level* of buffer j, which means the amount of workload that buffer j already received. So the buffer packing procedure can be described as following: we pack job i into buffer $j+1$ if $w_i + \eta_j > w_c$, and process jobs in buffer j at speed s_j where s_j is the solution to $\eta_j \frac{D(s_j)}{s_j} = h^*$. The machine serves the buffers one by one, with one tool serving only one buffer. We call it **Fixed Buffer Approach** (**FB**). As we can see, **FB** can make full use of every tool. No tool would be replaced with non-zero level. **FB** is described in **Algorithm 2**.

Notice in Algorithm 2, **FB** can also deal with the case when workload is unbounded, we leave the

Algorithm 2 Fixed Buffer Approach

for i=1:$n(L)$ **do**
  if $\eta + w_i \leq w_c$ **then** Put job i into the buffer and $\eta \leftarrow \eta + w_i$;
  else if $\eta + w_i \geq w_c$ && $w_i < w_c$ **then** Process jobs in buffer with speed s determined by $\eta\frac{D(s)}{s} = h^*$; $\eta \leftarrow w_i$;
  else Process jobs in buffer with speed s determined by $\eta\frac{D(s)}{s} = h^*$; Replace; Process job w_i with speed s_i determined by $w_i\frac{D(s_i)}{s_i} = h^*$; Replace; $\eta \leftarrow 0$;
  end if
  if i==$n(L)$ **then** Process jobs in buffer with speed s determined by $\eta\frac{D(s)}{s} = h^*$; Replace the tool;
  end if
end for

discussion for unbounded workload case to Section 4.3.2. Here we mainly focus on discussion for bounded workload cases.

Now the problem is how to fix the buffer's capacity. Notice that if $w_u \leq w_b$, then $w_c = w_b$ may be a good option since perfect packing is achievable in this case. When $w_u > w_b$, we may try fixing $w_c = w_u$. Again, here we assume that workload is uniformly distributed within $[0, w_u]$. We choose $w_c = \max\{w_b, w_u\}$. In this case we denote processing time of job list L under **FB** as $T^0_{FB}(L)$. Notice that this choice of w_c is not optimal. However, by showing the performance of this we will find the lower bound that **FB** can achieve. Notice that the solution of $T_{BC}(L)$ may not be achievable by **OFR** when $w_u > w_b$. Definitely we can use **BC** as a benchmark but it may perform far from **OFR**. To find the optimal value of **OFR**, we provide the following lemma.

Lemma 6. *Let $D(s) = \frac{s^\nu}{C}h^*$, given a list of jobs L where there is a subset of L in which each workload is less than or equal to w_b, say L_1, if L_1 can be perfectly put into buffers each of capacity w_b, then the optimal solution to **OFR** contains two parts: jobs in L_1 are perfectly packed into buffers thus $T_{OFR}(L_1) = T_{BC}(L_1)$, and jobs in $L_0 = L \setminus L_1$ each is served by one tool thus $T_{OFR}(L_0) = T_{OF}(L_0)$.*

Proof. We prove our theorem by contradiction. Suppose $L^* \in R(L)$ is the optimal order for **OFR**, which means the optimal value of L^* by solving **OF** is less than the optimal value of any other lists in $R(L)$. First we prove that in the optimal solution to the $T_{OF}(L^*)$, it is impossible that one tool serves more than one jobs whose workload is greater than w_b. Suppose that $w_1 \geq w_2 \geq w_b$ and w_1 and w_2 are served by the same

tool. Then the processing time plus replacement time is

$$\begin{aligned}\frac{(w_1+w_2)^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}+\tau &\geq \frac{w_1^{\frac{\nu}{\nu-1}}+\frac{\nu}{\nu-1}w_1^{\frac{1}{\nu-1}}w_2}{C^{\frac{1}{\nu-1}}}+\tau\\ &\geq \frac{w_1^{\frac{\nu}{\nu-1}}+\frac{\nu}{\nu-1}w_2^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}+\tau\\ &\geq \frac{w_1^{\frac{\nu}{\nu-1}}+w_2^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}+\frac{1}{\nu-1}\frac{w_b^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}+\tau=\frac{w_1^{\frac{\nu}{\nu-1}}+w_2^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}+2\tau.\end{aligned}$$

Since the last expression of the above inequalities is the processing time when w_1 and w_2 done by two tools separately, then we cannot have the case where two large jobs are served by one tool.

Now suppose that in list L^*, when getting the optimal solution of **OF**, there are elements in L_0, say w_{0j}, which are served with some of the elements in L_1 by same tools. Since we only consider the workload served by tools, the tools that serve L_0 are serving $\sum_{j=1}^{n(L_0)}(w_{0j}+\gamma_j)$ in total, where γ_j denotes the workload from L_1 which is served with w_{0j} by the same tool. Let $\widehat{w}=w(L)-\sum_{j=1}^{n(L_0)}(w_{0j}+\gamma_j)$. Then the time spent on L^* under **OF** is

$$n(L^*)T_{OF}(L^*) \geq \sum_{j=1}^{n(L_0)}\frac{(w_{0j}+\gamma_j)^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}+n(L_0)\tau+\frac{\widehat{w}^{\frac{\nu}{\nu-1}}}{(C\frac{\widehat{w}}{w_b})^{\frac{1}{\nu-1}}}+\tau\frac{\widehat{w}}{w_b}.$$

The inequality holds since given any list L, $T_{OF}(L)\geq T_{BC}(L)$. Let $\gamma=\sum_{i=1}^{n(L_0)}\gamma_i$. Suppose $T_{B1}(L^*)$ is the processing time by using the way described in **Lemma 6**, i.e.,

$$n(L^*)T_{B1}(L^*)=\frac{\widehat{w}+\gamma}{C^{\frac{1}{\nu}}}\nu(\nu-1)^{\frac{1-\nu}{\nu}}\tau^{\frac{1}{\nu}}+\sum_{j=1}^{n(L_0)}\frac{w_{0j}^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}+n(L_0)\tau.$$

From the fact that $\frac{\widehat{w}}{C^{\frac{1}{\nu}}}\nu(\nu-1)^{\frac{1-\nu}{\nu}}\tau^{\frac{1}{\nu}}=\frac{\widehat{w}^{\frac{\nu}{\nu-1}}}{(C\frac{\widehat{w}}{w_b})^{\frac{1}{\nu-1}}}+\tau\frac{\widehat{w}}{w_b}$, we have

$$n(L^*)\left(T_{OF}(L^*)-T_{B1}(L^*)\right)=\sum_{j=1}^{n(L_0)}\frac{(w_{0j}+\gamma_j)^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}-\frac{\gamma}{C^{\frac{1}{\nu}}}\nu(\nu-1)^{\frac{1-\nu}{\nu}}\tau^{\frac{1}{\nu}}-\sum_{j=1}^{n(L_0)}\frac{w_{0j}^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}.$$

From Taylor's expansion and $w_j > w_b$ we have

$$(w_{0j}+\gamma_j)^{\frac{\nu}{\nu-1}} \quad \geq \quad w_{0j}^{\frac{\nu}{\nu-1}} + \frac{\nu}{\nu-1} w_{0j}^{\frac{1}{\nu-1}} \gamma_j > w_{0j}^{\frac{\nu}{\nu-1}} + \frac{\nu}{\nu-1} \left((\nu-1)\tau C^{\frac{1}{\nu-1}}\right)^{\frac{1}{\nu}} \gamma_j.$$

Thus we show that $\sum_{j=1}^{k} \frac{(w_0 j+\gamma_j)^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}} - \frac{\gamma}{C^{\frac{1}{\nu}}} \nu(\nu-1)^{\frac{1-\nu}{\nu}} \tau^{\frac{1}{\nu}} - \sum_{j=1}^{k} \frac{w_{0j}^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}} > 0$. Hence we can see that $T_{OF}(L^*) - T_{B1}(L^*) > 0$, which means that L^* is not the best order. □

Now suppose we have a list L in which workload of each job is uniformly distributed within $[0, w_u]$ where $w_u > w_b$. Extract the elements whose workload are greater than w_b from L to form a new list L_0. From the remaining elements we extract those that can be perfectly packed by tools with capacity w_b and form them as list L_1. The other elements are gathered to be list L_2. Notice we do not consider the order of these lists since we only discuss the rearranging method here.

Given any list L, we have that $n(L)T_{OFR}(L) \leq n(L_1 \cup L_2)T_{OFR}(L_1 \cup L_2) + n(L_0)T_{OFR}(L_0)$ since rearranging always guarantees the optimal order of the entire list L. Also notice that $T_{OFR}(L_1) = T_{BC}(L_1)$, from **Lemma 6** we have $n(L)T_{OFR}(L) \geq n(L_1 \cup L_0)T_{OFR}(L_1 \cup L_0) = n(L_1)T_{OFR}(L_1) + n(L_0)T_{OFR}(L_0)$. Notice that $L_1 \cup L_2$ contains all the elements whose workload is less than or equal to w_b, and L_0 contains all elements whose workload are greater than w_b. We want to get the expected number of jobs in these sets when given $n(L)$. We have the following lemma, for which we define W as the workload of a job arriving in the future:

Lemma 7. ***P***$(W \leq a | W > w_b)$ *is the cdf of a uniform distribution in* $[w_b, w_u]$ *if* W *is uniformly distributed in* $[0, w_u]$.

Proof. Consider the conditional cdf when $a > w_b$

$$\mathbf{P}(W \leq a | W > w_b) = \frac{\mathbf{P}(w_b < W < a)}{\mathbf{P}(W > w_b)} = \frac{a - w_b}{w_u - w_b}.$$

If $a \leq w_b$ we have that $\mathbf{P}(W < a | W > w_b) = 0$. □

Similarly, we can show that $\mathbf{P}(W \leq a | W < w_b)$ is also a cdf for a uniform distribution in $[0, w_b]$. Let $\theta = \frac{w_u}{w_b}$ then $\mathbf{E}[n(L_1 \cup L_2)|n(L)] = \frac{1}{\theta} n(L)$ and $\mathbf{E}[n(L_0)|n(L)] = \frac{\theta-1}{\theta} n(L)$. To characterize $\mathbf{E}[T_{OFR}(L_1 \cup L_2)]$,

from **Lemma 7** we know that the elements in $L_1 \cup L_2$ have uniform distribution with upper bound w_b. From **Theorem 4** as $n(L_1 \cup L_2) \to \infty$, $\mathbf{E}[T_{OFR}(L_1 \cup L_2)] \to \mathbf{E}[T_{BC}(L_1 \cup L_2)]$. Also from **Lemma 3** we know that $\lim_{n(L_1 \cup L_2) \to \infty} \mathbf{E}[T_{BC}(L_1 \cup L_2)] \leq \lim_{n(L_1 \cup L_2) \to \infty} \dfrac{(\frac{\mathbf{E}[w(L_1 \cup L_2)]}{w_b} + 1)\tau\nu}{n(L_1 \cup L_2)} = \dfrac{\tau\nu}{2}$, and

$$\lim_{n(L_1 \cup L_2) \to \infty} \mathbf{E}[T_{BC}(L_1 \cup L_2)] \geq \lim_{n(L_1 \cup L_2) \to \infty} \frac{\tau^{\frac{1}{\nu}} (\nu - 1)^{\frac{1}{\nu} - 1} \nu \mathbf{E}[w(L_1 \cup L_2)]}{C^{\frac{1}{\nu}} n(L_1 \cup L_2)} = \frac{\tau\nu}{2}.$$

Thus $\lim_{n(L_1 \cup L_2) \to \infty} \mathbf{E}[T_{BC}(L_1 \cup L_2)] = \dfrac{\tau\nu}{2}$.

Now we wish to show that

$$\lim_{n(L) \to \infty} \frac{\mathbf{E}[n(L_1) T_{BC}(L_1) | n(L)]}{n(L)} = \lim_{n(L) \to \infty} \frac{\mathbf{E}[n(L_1 \cup L_2) T_{BC}(L_1 \cup L_2) | n(L)]}{n(L)}.$$

Since $\dfrac{n(L_1)}{n(L)} T_{BC}(L_1) \leq \dfrac{n(L_1 \cup L_2)}{n(L)} T_{BC}(L_1 \cup L_2)$, and its RHS has the property that $\dfrac{n(L_1 \cup L_2)}{n(L)} T_{BC}(L_1 \cup L_2) \leq \dfrac{n(L_1)}{n(L)} T_{BC}(L_1) + \dfrac{n(L_2)}{n(L)} T_{BC}(L_2)$, we want to show that $\lim_{n(L) \to \infty} \dfrac{\mathbf{E}[n(L_2) | n(L)]}{n(L)} = 0$. Otherwise assume that $\lim_{n(L) \to \infty} \dfrac{\mathbf{E}[n(L_2) | n(L)]}{n(L)} = \delta > 0$ holds, then

$$\lim_{n(L_1 \cup L_2) \to \infty} \frac{\mathbf{E}[w(L_1 \cup L_2) | n(L)]}{\mathbf{E}[k_{opt}(L_1 \cup L_2) | n(L)]} = \lim_{n(L) \to \infty} \frac{\mathbf{E}[w(L_1 \cup L_2) | n(L)]}{\mathbf{E}[k_{opt}(L_1) + k_{opt}(L_2) | n(L)]} < w_b.$$

Thus $\lim_{n(L) \to \infty} \dfrac{\mathbf{E}[n(L_2) T_{BC}(L_2) | n(L)]}{n(L)} = 0$.

Then we have that

$$\begin{aligned} \lim_{n(L) \to \infty} \mathbf{E}[T_{OFR}(L)] &= \lim_{n(L_1 \cup L_2) \to \infty} \frac{1}{\theta} \mathbf{E}[T_{BC}(L_1 \cup L_2)] + \lim_{n(L_0) \to \infty} \frac{\theta - 1}{\theta} \mathbf{E}[T_{OF}(L_0)] \\ &= \frac{\tau\nu}{2\theta} + \lim_{n(L_0) \to \infty} \frac{\theta - 1}{\theta} \mathbf{E}[T_{OF}(L_0)]. \end{aligned}$$

Next we wish to evaluate $\lim_{n(L_0) \to \infty} \mathbf{E}[T_{OF}(L_0)]$. Let $m = \frac{\nu}{\nu - 1}$ and W be uniformly distributed in $[0, w_u]$, then

$$\mathbf{E}[W^m | W > w_b] = \int_{w_b}^{w_u} \frac{1}{w_u - w_b} x^m dx = \frac{1}{(m+1)(w_u - w_b)} \left(w_u^{m+1} - w_b^{m+1} \right) = \frac{1}{m+1} \frac{1}{\theta - 1} w_b^m (\theta^{m+1} - 1).$$

Then we have $\lim_{n(L_0)\to\infty} \mathbf{E}[T_{OF}(L_0)] = \frac{1}{m+1}\frac{1}{\theta-1}\frac{w_b^m}{C^{\frac{1}{\nu-1}}}(\theta^{m+1}-1)+\tau$. Thus as $n(L)\to\infty$,

$$\mathbf{E}[T_{OFR}(L)] \to \left(\frac{\nu}{2\theta}+\frac{\theta-1}{\theta}+\frac{\nu-1}{\theta(m+1)}(\theta^{m+1}-1)\right)\tau.$$

Lemma 8. *Let $f_L(x)$ be the probability density function of level of each tool, if capacity of each tool is w_u and the workload of each job is uniformly distributed in $[0, w_u]$, then $f_L(x) = \frac{3x^2}{w_u^3}$ where $x \in [0, w_u]$.*

Proof. Proof is given by Coffman et al. [1980]. □

From **Lemma 8** we have that

$$\mathbf{E}[\eta^m] = \int_0^{w_u} \frac{3x^{m+2}}{w_u^3}dx = \frac{3}{3+m}w_u^m.$$

Thus $\lim_{n(L)\to\infty} \mathbf{E}[T_{FB}^0(L)] = \frac{2}{3}(\frac{3}{3+m}\frac{\theta^m w_b^m}{C^{\frac{1}{\nu-1}}}+\tau)$.

Then as $n(L)\to\infty$, $\dfrac{\mathbf{E}[T_{OFR}(L)]}{\mathbf{E}[T_{FB}^0(L)]} \to \dfrac{\nu-2+2\theta+\frac{2(\nu-1)^2}{2\nu-1}(\theta^{m+1}-1)}{\frac{4(\nu-1)^2}{4\nu-3}\theta^{m+1}+\frac{4}{3}\theta}$.

Remark 9. *If workload of each job is uniformly distributed within $[0, w_u]$ and $\theta = \frac{w_u}{w_b} \geq 1$, suppose $w_c^* \geq w_u$ is the optimal tool capacity to be fixed in **FB**, then as $n(L)\to\infty$,*

$$\frac{\boldsymbol{E}[T_{OFR}(L)]}{\boldsymbol{E}[T_{FB}(L)]} \in [\frac{\nu-2+2\theta+\frac{2(\nu-1)^2}{2\nu-1}(\theta^{m+1}-1)}{\frac{4(\nu-1)^2}{4\nu-3}\theta^{m+1}+\frac{4}{3}\theta}, 1].$$

Notice we do not use $T_{BC}(L)$ here as frame of reference anymore since it may not be achievable when $w_u > w_b$.

Remark 10. *We use uniform distribution because in Section 4.2 we showed that the tool level or the buffer level η is determined by $\frac{w_u}{w_b}$. When tool replacement time is large, w_b is large also. Uniform distribution in this case provides us with an extreme case where w_u can be at least as large as w_b. Results for some other distributions are shown in Section 5.*

As we can see, if we fix buffer capacity as w_c when $w_u \leq w_c$, both **FB FS** process at most w_c workload. The only difference is that if both **FB** and **FS** process the same amount of workload, say $\widehat{\eta} \leq w_c$, using the

same tool, **FB** will process at speed $\widehat{s}$, which is the solution of $\widehat{\eta}\frac{D(\widehat{s})}{\widehat{s}} = h^*$, whereas **FS** processes it at speed s^*, where s^* satisfies $w_c\frac{D(s^*)}{s^*} = h^*$. Notice that $\widehat{\eta} \leq w_c$ and the fact that $\frac{D(s)}{s}$ is an increasing function of s, we have that $\widehat{s} \geq s^*$. In this case we can see that $T_{FB}(L) \leq T_{FS}(L)$ for any list L. Then we have the following theorem.

Theorem 11. *For **FS** we fix a speed s^* as the solution of $w_c\frac{D(s^*)}{s^*} = h^*$, for **FB** we fix an buffer capacity also as w_c and use s^* to process jobs in the last buffer, then $T_{FS}(L) \geq T_{FB}(L) \geq T_{OF}(L) \geq T_{OFR}(L) \geq T_{BC}(L)$ for any L.*

Proof. It is easy to observe that $T_{FS}(L) \geq T_{FB}(L)$ and $T_{OF}(L) \geq T_{OFR}(L)$, and $T_{BC}(L)$ would be the smallest as it is the best case. We only need to prove that $T_{FB}(L) \geq T_{OF}(L)$. From **FB** we know its processing speed is determined by $\eta\frac{D(s)}{s} = h^*$, thus remaining tool level cannot be negative. Then it is easy to see that any feasible solutions of **FB** are in the feasible region of **OF**. Thus $T_{FB}(L) \geq T_{OF}(L)$. □

In previous sections we do not consider the waiting process or the queue in the systems since stability does not hamper the performance of our approaches. We now characterize the capacity of the system under different approaches. For **FS** with capacity w_c, the service rate for list L is $\frac{w(L)}{n(L)T_{FS}(L)}$. Notice that if $k+1$ tools are used for processing L and η_i is the workload served by tool i, then the service rate can be given by $\frac{w(L)}{n(L)T_{FS}(L)} = \frac{\sum_{i=1}^{k}\eta_i+\eta_{k+1}}{\sum_{i=1}^{k+1}\frac{\eta_i}{s}+(k+1)\tau} \leq \frac{(k+1)w_c}{\frac{(k+1)w_c}{2s}+(k+1)\tau}$, where $s = \left(\frac{C}{w_c}\right)^{\frac{1}{\nu-1}}$. By dominated convergence theorem and strong law of large numbers, we have that as $n(L) \to \infty$, $\mathbf{E}[\frac{w(L)}{n(L)T_{FS}(L)}] \to \frac{(1-\beta)w_c}{\frac{(1-\beta)w_c^{\frac{\nu}{\nu-1}}}{C^{\frac{1}{\nu-1}}}+\tau} = \frac{(1-\beta)w_cC^{\frac{1}{\nu-1}}}{(1-\beta)w_c^{\frac{\nu}{\nu-1}}+\tau C^{\frac{1}{\nu-1}}}$, where $1-\beta = \mathbf{E}[\eta]$. For the optimal capacity w_c^* for **FS** we have $\mathbf{E}[\frac{w(L)}{n(L)T_{FS}(L)}] \in [\frac{(1-\beta)w_bC^{\frac{1}{\nu-1}}}{(1-\beta)w_b^{\frac{\nu}{\nu-1}}+\tau C^{\frac{1}{\nu-1}}}, \frac{w_b}{\nu\tau}]$ knowing that **BC** provides the maximal service rate and w_c^* provides higher service rate than w_b does for **FS**. Thus an arrival rate no greater than $\frac{(1-\beta)w_cC^{\frac{1}{\nu-1}}}{(1-\beta)w_c^{\frac{\nu}{\nu-1}}+\tau C^{\frac{1}{\nu-1}}}$ is the necessary condition for stability of **FS**. Likewise, for capacity of **FB**, we have as $n(L) \to \infty$, $\mathbf{E}[\frac{w(L)}{n(L)T_{FB}(L)}] \to \frac{\mathbf{E}[\eta]}{\frac{\mathbf{E}[\eta^{\frac{\nu}{\nu-1}}]}{C^{\frac{1}{\nu-1}}}+\tau} = \frac{\mathbf{E}[\eta]C^{\frac{1}{\nu-1}}}{\mathbf{E}[\eta^{\frac{\nu}{\nu-1}}]+\tau C^{\frac{1}{\nu-1}}}$. Specifically, when $W_i \sim U(0, w_u)$, $w_u > w_b$ and based on the optimal capacity, we have the arrival rate up to $\frac{\frac{3}{4}w_uC^{\frac{1}{\nu-1}}}{\frac{3\nu-3}{3\nu-2}w_u^{\frac{\nu}{\nu-1}}+\tau C^{\frac{1}{\nu-1}}}$ as the necessary condition of the stability for **FB**.

4.3.2 Unbounded Workload

Here we discuss the case when workload is unbounded. In this case, we may receive jobs with workload greater than w_c so that a single buffer cannot hold it. However, Taylor's Formula in Equation (1) tells us to use a relative low speed to process a large job. **FB** can adjust the processing speed accordingly without putting the large job in a "virtual buffer", as stated in Algorithm 2. Jobs with small workload are still put into buffers one by one. Once a buffer is full, **FB** processes jobs in this buffer using the optimal speed, and new jobs are put into the next buffer. Once a job with workload $w_i > w_c$ arrives, we know this job cannot be put into either the current buffer or a new buffer whose capacity is w_c. In this case **FB** would firstly process jobs in the current buffer using the optimal speed, then process this large job using one tool by speed s_i, where s_i is the solution to $w_i \frac{D(s_i)}{s_i} = h^*$.

In the unbounded workload case, the optimal value of **OFR** can no longer be approximated by **BC**, thus we can only use **BC** as the benchmark. However **BC** may not be a good benchmark because a real workload greater than w_b can never be packed in a buffer with capacity $w_c = w_b$ (See Section 4.1 and Theorem 4 for detail). We thus provide a modified **BC** to approximate the optimal makespan of $T_{OFR}(L)$. For simplicity in notation, we denote the modified problem **BC***. Similar to Lemma 6, we let L_1 be the subset of L containing jobs whose workload is smaller than w_b and $L_0 = L \backslash L_1$. Let $k^* = \lfloor \frac{w(L_1)}{w_b} \rfloor$, then from Lemma 3 we have $k^* \tau\nu \leq n(L_1) T_{BC}(L_1) < (k^*+1)\tau\nu$. We let $T_{BC*}(L) = k^*\tau\nu + T_{OFR}(L_0)$, and by Lemma 6 we have $T_{BC^*}(L) \leq T_{OFR}(L) < (k^*+1)\tau\nu + T_{OFR}(L_0)$. Then $\lim_{n(L)\to\infty} \frac{\mathbf{E}[T_{BC*}(L)]}{\mathbf{E}[T_{OFR}(L)]} = 1$ if we assume workload is uniformly distributed in $(0, w_u)$, This implies that we can also use **BC*** to approximate **OFR** for the results in Section 4.3.1. Note that without the uniform distribution assumption, we still have $T_{BC^*}(L) \leq T_{OFR}(L)$. We then use **BC*** as the benchmark to compare **FB** against. The numerical study of this case is provided in Section 5.

5 Numerical Results

In this section we perform numerical experiments to develop insights in terms of performance for various model parameters as well as compare methods. Since in Section 4.2 and 4.3 we only give the optimal bound

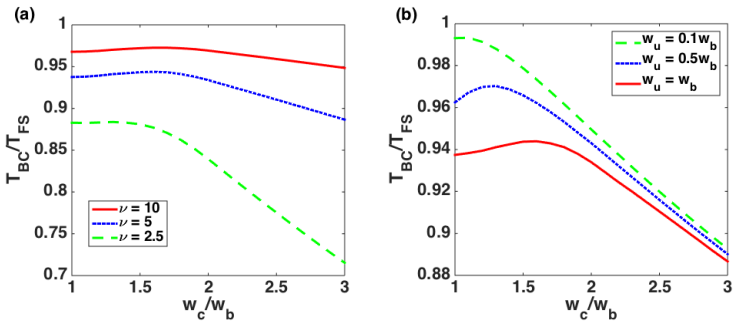

Figure 1: Performance of Fixed Speed Approach when (a) $w_u = w_b$, (b) $\nu = 5$

for both **FS** and **FB**, now we will study their performance using simulation. Notice that in **Theorem 5**, the expression $\lim_{n(L)\to\infty} \dfrac{\mathbf{E}[T_{BC}(L)]}{\mathbf{E}[T_{FS}(L)]}$ depends only on ν and β, where ν is determined by Taylor's Formula (1) which is also shown in Equation (2) and β is determined by $\frac{w_u}{w_b}$, knowing that w_b is a constant given in **Lemma 3** and w_u is the upper bound of each workload. Similarly, the expression $\lim_{n(L)\to\infty} \dfrac{\mathbf{E}[T_{BC}(L)]}{\mathbf{E}[T_{FB}(L)]}$ in **Theorem 6** depends only on ν and θ, where θ is determined by w_u. So we start by testing the performance of **FS** and **FB** for different ν and w_u. For numerical results, we choose $C = 100$ and $\tau = 100$. Once C and τ are fixed, w_b depends only on ν as shown in **Lemma 3**. All the numerical tests are done in MATLAB and $T_{OF}(L)$ is solved by Bonmin solver (see Bonami [2007]).

We begin by considering the influence of ν on **FS**. From Stephenson and Agapiou [2016] we can get some typical ν for several common tool materials, which is provided in the Table 3. We let $n(L) = 10^5$, a finite but large number which helps show the asymptotic results, selecting w_b as defined in **Lemma 3**. We then get Figure 1(a) for different ν's by letting $w_u = w_b$, and each line is the average result over 100 independent lists with each list having length $n(L) = 10^5$. As shown in Figure 1(a), large ν improves the performance of **FS**. When $\nu = 10$, **FS** is close to **BC** no matter how we choose the tool capacity w_c. Next we show the influence of w_u by fixing $n = 0.2$, i.e., $\nu = 5$. Let workload be drawn from $U(0, w_u)$. We test **FS** by letting $n(L) = 10^5$ for each of the 100 lists and we get Figure 1(b) for different w_u's. From Figure 1(b) we can see small $\frac{w_u}{w_b}$ also improves the performance of **FS**. From a practical standpoint, when replacement time τ is large, then the corresponding w_b is large thus $\frac{w_u}{w_b}$ is small, making **FS** work extremely well. Figure 1(b) also shows that w_b is not always the best option for fixing w_c. When $w_u = 0.5w_b$ and $w_u = w_b$, the maximal points of both lines correspond to best w_c's which are larger than w_b.

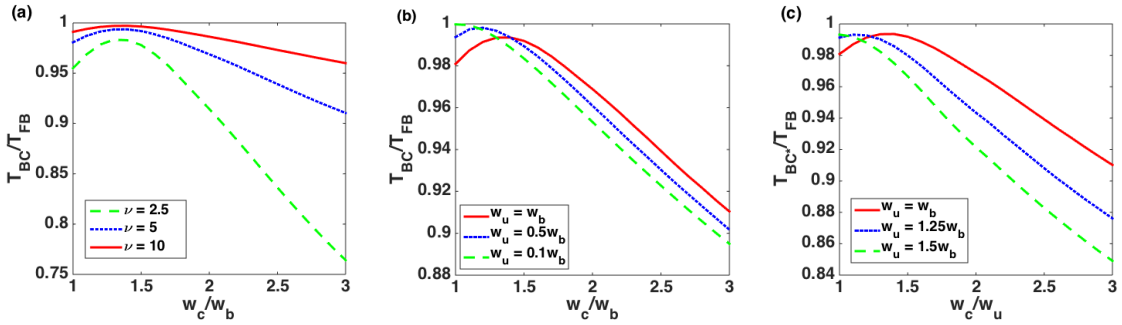

Figure 2: Performance of Fixed Buffer Approach when (a) $w_u = w_b$, (b) $w_u \leq w_b$, $\nu = 5$, (c) $w_u \geq w_b$, $\nu = 5$

Now we show the performance of **FB** for different ν's and w_u's. Notice that when $w_u > w_b$ we cannot use **BC** to approximate the optimal value of **OFR**, as stated in **Lemma 6**. Thus we use **BC*** as the frame of reference (details are in Section 4.3.2). We draw $n(L) = 10^5$ jobs independently for each of 100 job lists to approximate the expected optimal value of **FB** and **OFR**. Similarly, we find from Figure 2(a) where we let $w_u = w_b$ that large ν improves the performance of **FB**. When $\nu = 10$, we can see the best performance of **FB** is as good as **OFR**. Next we show the performance of **FB** for different w_u's. We fix $\nu = 5$ and let workload be drawn from $U(0, w_u)$. The performance of **FB** is shown in Figure 2(b) and Figure 2(c) for $w_u \leq w_b$ and $w_u \geq w_b$ respectively. Similarly, we observe that small $\frac{w_u}{w_b}$ improves the performance of **FB**. Notice that when given a large replacement time τ, the corresponding w_b is large, causing $\frac{w_u}{w_b}$ to be small. So the case where $w_u \geq w_b$ is an extreme case, but it turns out that **FB** works well even w_u is large. Besides, as shown in Figure 2(d) where we also test **FB** on modified beta distribution with different parameters, the optimal performance of **FB** is not sensitive to the assumption of probability distributions.

In Section 4.3.2 we show **FB** can be applied to the case when workload is unbounded, here we show in Figure 3 the performance of **FB** under different workload distributions. For all the cases in Figure 3, we assume $\nu = 5$. In Figure 3(a) we assume workload is according to modified Beta distributions. **BC** is used as the benchmark in this case since **OFR** cannot be approximated by **BC**. As we can see, the performance of **FB** under Beta distributions is similar to it has for uniform distribution, which shows when workload is bounded, the performance of **FB** is not sensitive to assumptions for workload distribution. Figure 3(b) and (c) show the performance for **FB** when workload is unbounded. **BC*** is used as the benchmark. Similarly, we find if we fix tool capacity for **FB** to be around $1.5w_b$, the gap between **BC*** and **FB** is always small

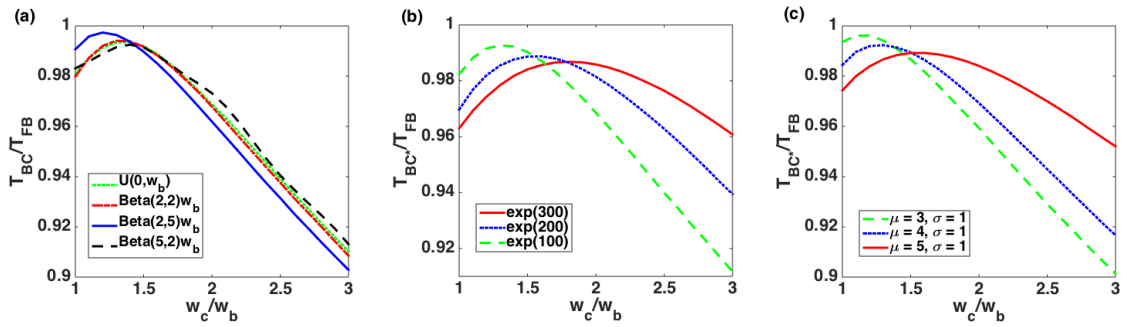

Figure 3: Performance of Modified Fixed Buffer Approach when workload is (a) Beta Distributed, $\nu = 5$ (b) Exponential Distributed, $\nu = 5$, (c) Log-Normal Distributed, $\nu = 5$.

($\frac{T_{BC*}}{T_{FB}} \in (0.98, 1)$), which shows the robustness of **FB**.

We then provide a comparison of methods to show the power of revealing workload using IoT. We select the parameters so that each method performs its own best. We again let $w_u = w_b$, and fix $w_{c_1} = 1.6w_b$ when $\nu = 5$ and $w_{c_1} = w_b$ when $\nu = 2$ for **FS** since they are numerically near optimal. Similarly we fix $w_{c_2} = 1.4w_b$ when $\nu = 5$ and $w_{c_2} = 1.3w_b$ when $\nu = 2$ for **FB**. We test **FS**, **FB** and **OF** by drawing workload $n(L) = 45$ times independently from $U(0, w_u)$ where we let $w_u = w_b$. We choose $n(L) = 45$ and each tool serves $\frac{3}{2}$ jobs on average as shown in Section 4.3, thus every $n(L)$ jobs correspond to nearly 30 tools. We run this experiment for 100 times and the average result over these 100 experiments are shown in Table 4.

Knowing the workload also results in a tremendous improvement in the performance of online algorithms. In the case where no workload is revealed and if $w_u = w_b$, then the optimal policy is: the tool processes every job with a fixed speed which is determined by $w_b \frac{D(s)}{s} = h^*$. This is also the control policy provided in Conrad and McClamroch [1987]. The performance comparison is provided by the **Stochastic Optimal Policy (SOP)** in Table 4 below. The actual performance of this optimal policy is given by $\lim_{n(L)\to\infty} \frac{\mathbf{E}[T_{BC}(L)]}{\mathbf{E}[T_{SOP}(L)]} = \frac{\nu}{\nu+1}$. The **Fixed Speed Approach** requires one revealed workload, and we see from Table 4 below a big improvement of performance it makes (94.38% of optimal offline) compared to the **Stochastic Optimal Policy** (83.33% of optimal offline). Ability to acquire workload distribution beforehand is also important. It helps improve the performance of online algorithms. If workload is revealed however the distribution of future workload is unknown, the optimal control policy for this case would be myopic as well: we choose the maximal speed (given by $w\frac{D(s)}{s} = h^*$) to process the current job w. We call it **Simple Approach (SA)** (a simplified data-driven method) which is shown in Table 4. When using **Simple Approach** we assume

Material	n	$\nu = 1/n$
High Speed Steels	0.1-0.17	5.88-10.0
Uncoated WC Tools	0.2-0.25	4.0-5.0
TiN or TiN Coated WC Tools	0.3	3.3
Al_2O_3 Coated WC Tools	0.4	2.5
Solid Ceramics Tools	0.4-0.6	1.67-2.5

Table 3: Taylor's Parameter ν for different materials.

$\frac{T_{BC}}{T_{Method}}$ Methods	when $\nu = 5$	when $\nu = 2$
Stochastic Optimal Policy (No revealed workload)	0.8286	0.6600
Simple Approach (No distribution knowledge)	0.8933	0.7347
Fixed Speed (One revealed workload)	0.9402	0.8509
Fixed Buffer (More than one revealed workload)	0.9929	0.9727
Offline (All revealed information)	0.9957	0.9842

Table 4: Performance of Different Methods for Different Values of ν

workloads are drawn from a single uniform distribution; however our online algorithms can be modified accordingly if workload distributions are changed.

6 Concluding Remarks and Future Work

In this paper we consider a custom-manufacturing setting with IoT. A benefit of using IoT is that it is possible to accurately ascertain workloads and tool degradation before processing a job as well as to know workload distribution beforehand. However until information is available in custom-manufacturing settings there is uncertainty. We use a behavior-imitating method to construct our online algorithms so that algorithms behave similar to offline solutions thus have performance guaranteed. We achieve a middle ground in our algorithm using a bin-packing setting where jobs are "packed" into each tool so that the tool is used to process those jobs and then discarded. We call these algorithms **Fixed Speed Approach** (i.e., **FS**) and **Fixed Buffer Approach** (i.e., **FB**) where information for certain small number of jobs is needed and as a result, they perform exceedingly well. In addition, the setting considers fairly arbitrary arrival processes and hence can easily be extended to multi-station systems.

The contribution of this paper is as follows. We provide a model under IoT manufacturing which considers in a single frame work (1) discrete part manufacturing, (2) machine with degrading tools and degradation is a function of processing speed, (3) jobs where workload is revealed upon "arrival" and (4) objective of minimizing completion time. In addition, we provide the idea of behavior-imitation which can be adapted to other online algorithms as well. There are also several insights that can be gleaned from our problem. We show that job re-sequencing is unnecessary for general settings where the workload is far less than the tool replacement time. This implies when replacement takes a long time, online algorithms perform very well. Also because the loss caused by prohibiting re-sequencing is compensated by the ability of fast processing,

the online algorithms that we provide work reasonably well in the case when future jobs' information is unavailable. Moreover, we provide a list of optimal resource (tools) and time needed for processing a job list, which greatly enhance production planning. We show the optimal workload a tool serves depends only on the tool properties. Further, the performance of online algorithm in our setting relies heavily on the tool material. This helps practitioners to choose tool material scientifically.

There are several extensions to this research that can be considered in the future. Single stations with multiple tools per machine and multiple machines can be considered with degrading tools. The key idea would be to extend the policies that have been shown to be powerful in this paper. Subsequently, it would also be possible to extend to multiple workstations with various part flows. It needs to be seen if a decomposed approach would indeed result in efficient operations under such a setting. The effects of uncertainty especially in terms of random failures and random tool wear could also be considered in the future. While these problems do get much more complex than what is considered in this paper, we believe that the theories would nicely extend to more complex scenarios and they can be useful in designing and controlling large scale IoT-based smart-manufacturing systems with customized production.

Acknowledgment

The authors are grateful to the editorial board and anonymous reviewers. Their comments and suggestions have significantly improved the content and presentation of this work.

References

Ramakrishna Akella and P Kumar. Optimal control of production rate in a failure prone manufacturing system. *IEEE Transactions on Automatic control*, 31(2):116–126, 1986.

Pierre Bonami. Bonmin users' manual, 2007. URL `https://www.coin-or.org/Bonmin/`.

El-Kébir Boukas, Qingz Zhang, and G Yin. Robust production and maintenance planning in stochastic manufacturing systems. *IEEE Transactions on Automatic Control*, 40(6):1098–1102, 1995.

Xiaoqiang Cai, Xianyi Wu, and Xian Zhou. Stochastic scheduling subject to preemptive-repeat breakdowns with incomplete information. *Operations Research*, 57(5):1236–1249, 2009.

Chandra Chekuri, Rajeev Motwani, Balas Natarajan, and Clifford Stein. Approximation techniques for average completion time scheduling. *SIAM Journal on Computing*, 31(1):146–166, 2001.

TCE Cheng, Chunyan Gao, and Houcai Shen. Production and inventory rationing in a make-to-stock system with a failure-prone machine and lost sales. *IEEE transactions on automatic control*, 56(5):1176–1180, 2011.

E.G. Coffman, K. So, M. Hofri, and A.C. Yao. A stochastic model of bin-packing. *Information and Control*, 44(2):105–115, February 1980.

C Conrad and N McClamroch. The drilling problem: A stochastic modeling and control example in manufacturing. *IEEE Transactions on automatic control*, 32(11):947–958, 1987.

Youyi Feng and Baichun Xiao. Optimal threshold control in discrete failure-prone manufacturing systems. *IEEE Transactions on Automatic Control*, 47(7):1167–1174, 2002.

Youyi Feng and Houmin Yan. Optimal production control in a discrete manufacturing system with unreliable machines and random demands. *IEEE Transactions on Automatic Control*, 45(12):2280–2296, 2000.

Noah Gans and Garrett Van Ryzin. Optimal control of a multiclass, flexible queueing system. *Operations Research*, 45(5):677–693, 1997.

R. Gao, L. Wang, R. Teti, D. Dornfeld, S. Kumara, M. Mori, and M. Helug. Cloud-enabled prognosis for manufacturing. *CIRP Annals - Manufacturing Technology*, 64(2):749772, 2015.

Igor Griva, Stephen G. Nash, and Ariela Sofer. *Linear and Nonlinear Programming: Second Edition*. Society for Industrial and Applied Mathematics, January 2009.

Leslie A Hall. Approximability of flow shop scheduling. *Mathematical Programming*, 82(1-2):175–190, 1998.

Leslie A Hall, Andreas S Schulz, David B Shmoys, and Joel Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of operations research*, 22(3):513–544, 1997.

Jian-Qiang Hu, P Vakili, and Guo-Xiang Yu. Optimality of hedging point policies in the production control of failure prone manufacturing systems. *IEEE Transactions on Automatic Control*, 39(9):1875–1880, 1994.

Narendra Karmarkar. Probabilistic analysis of some bin-packing problems. In *23rd Annual Symposium on Foundations of Computer Science, IEEE SFCS'08.*, pages 107–111, 1982.

Bernhard Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial optimization*, volume 2. Springer, 2012.

C-Y Lee and V Jorge Leon. Machine scheduling with a rate-modifying activity. *European Journal of Operational Research*, 128(1): 119–128, 2001.

Chung-Yee Lee and Chen-Sin Lin. Single-machine scheduling with maintenance and repair rate-modifying activities. *European Journal of Operational Research*, 135(3):493–513, 2001.

K Liu and J Shi. Internet of things (iot)-enabled system informatics for service decision making: Achievements, trends, challenges, and opportunities. *IEEE Intelligent Systems*, 30(6):18–21, 2015.

Francesco Martinelli. Optimality of a two-threshold feedback control for a manufacturing system with a production dependent failure rate. *IEEE Transactions on Automatic Control*, 52(10):1937–1942, 2007.

L. Monostori, B. Kdr, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology*, 65(2):621641, 2016.

Gang Niu, Bo-Suk Yang, and Michael Pecht. Development of an optimized condition-based maintenance system by data fusion and reliability-centered maintenance. *Reliability Engineering and System Safety*, 95(7):786–796, 2010.

Zhan Pang. Optimal control of a single-product assemble-to-order system with multiple demand classes and backordering. *IEEE Transactions on Automatic Control*, 60(2):480–484, 2015.

Ghaith Rabadi, Reinaldo J Moraga, and Ameer Al-Salem. Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17(1):85–97, 2006.

Raymond Rishel. Controlled wear process: Modeling optimal control. *IEEE Transactions on Automatic Control*, 36(9):1100–1102, 1991.

S Sana, Suresh Kumar Goyal, and KS Chaudhuri. A production–inventory model for a deteriorating item with trended demand and shortages. *European Journal of Operational Research*, 157(2):357–371, 2004.

Paul J Schweitzer and Abraham Seidmann. Optimizing processing rates for flexible manufacturing systems. *Management Science*, 37 (4):454–466, 1991.

Sergey V Sevastianov and Gerhard J Woeginger. Makespan minimization in open shops: A polynomial time approximation scheme. *Mathematical Programming*, 82(1):191–198, 1998.

A Sharifnia. Production control of a manufacturing system with multiple machine states. *IEEE Transactions on Automatic Control*, 33(7):620–625, 1988.

Dong-Ping Song. Optimal production and backordering policy in failure-prone manufacturing systems. *IEEE transactions on automatic control*, 51(5):906–911, 2006.

N Srivatsan and Yves Dallery. Partial characterization of optimal hedging point policies in unreliable two-part-type manufacturing systems. *Operations Research*, 46(1):36–45, 1998.

David A. Stephenson and John S. Agapiou. *Metal Cutting Theory and Practice, Third Edition*. CRC Press, April 2016.

John W. Sutherland, Justin S. Richter, Margot J. Hutchins, David Dornfeld, Rachel Dzombak, Jennifer Mangold, Stefanie Robinson, Michael Z. Hauschild, Alexandra Bonou, Paul Schnsleben, and Felix Friemann. Design for additive manufacturing: Trends, opportunities, considerations, and constraints. *CIRP Annals - Manufacturing Technology*, 65(2):689–712, 2016.

Takeshi Takenaka, Yoshinobu Yamamoto, Ken Fukuda, Ayaka Kimura, and Kanji Ueda. Enhancing products and services using smart appliance networks. *CIRP Annals - Manufacturing Technology*, 65(1):397400, 2016.

E. Uhlmann, B. Mullany, D. Biermann, K.P. Rajurkar, T. Hausotte, and E. Brinksmeier. Process chains for high-precision components with micro-scale features. *CIRP Annals - Manufacturing Technology*, 65(2):549–572, 2016.

W. Xu and L. Cao. Optimal tool replacement with product quality deterioration and random tool failure. *International Journal of Production Research*, 53(6), 2015.