Scheduling Benefit-generating Jobs to Capacitated Machines with a Consideration of Fairness

Ling-Chieh Kung[*] and Chi-Wei Liu

Department of Information Management, National Taiwan University, Taiwan

Abstract

We consider a problem of allocating jobs to identical machines. Each job has an amount of workload and a benefit collected upon completion. All machines have the same capacity. Under the no-splitting constraint of jobs and capacity constraint of machines, the problem is to maximize the benefit earned by the machine earning the least benefit. Our problem is thus a capacitated job allocation problem with a consideration of fairness. After showing that this problem is NP-hard, we propose an approximation algorithm modified from the longest processing time (LPT) rule. The algorithm is proved to have a worst-case performance guarantee 1/2 when job benefits are linear to workloads. Different approximation factors are also derived for convex and concave relationships. Finally, numerical studies illustrate the average performances of the algorithm and demonstrates that this algorithm works well when the jobs exhibit economy of scale but not so well when they exhibit diminishing marginal benefits.

Keywords: Scheduling, approximation algorithm, benefit-workload relationship, capacitated machine, fairness.

1 Introduction

Job allocation and scheduling has been widely discussed over decades and applied in many fields (Pinedo, 2012). When one assigns jobs to multiple factories, machines, or agents, jobs bring in workloads as well as benefits upon completion. While traditionally these benefits all go to the

[*]lckung@ntu.edu.tw; corresponding author.

central decision maker, in many cases an individual factory/machine/agent also care about the benefits generated by itself. The issue of allocation fairness then emerges.

An anonymous leading LED chip manufacturer in Taiwan, who owns around twenty factories in Taiwan and China, faces exactly this issue. The company accepts orders for producing different kinds of LED chips from its customers. Once per month, the company assigns these orders to its factories. Because factory managers are evaluated according to the amount of revenues generated by completing orders (as well as other performance indicators), they indeed would fight for easy-to-complete jobs of high margins. From this perspective, jobs are actually valuable *resources* that should be fairly assigned to factories. Unlike typical resource allocation problems, however, here a factory manager cannot accept too many jobs due to its capacity limitation. Job allocation considering fairness is thus of new challenges.

Some social enterprises nowadays also face a similar problem. With the idea of "giving them jobs, not money," a social enterprise (or sometimes a government) may hire jobless people to do part-time jobs to earn their livings. For example, "The Sock Mob Homeless Volunteer Network" in London finds that many homeless people are familiar with the street and may provide their own perspective to view a city. As a result, it hires and trains homeless people to be London city tour guides.[1] As another example, the magazine company "The Big Issue Taiwan" hires the homeless to sell magazines.[2] For these social enterprises, the objective is not limited to job completion and revenue maximization. Instead, a more important objective is to distribute benefits generated by job completion to help as many people as possible. Again, the first priority is not on *efficiency* but on *fairness*.

In this study, we consider the aforementioned job allocation problem with the fairness issue. For ease of exposition, we will call those working agents as machines. The most important feature of our job allocation problem is that the decision maker considers not only the overall profitability but also fairness among machines. In the environment, each job has an associated workload and an associated benefit upon completion. Each machine has a limited capacity and can afford only a certain amount of total workloads. As long as a machine has enough capacity to complete jobs assigned to it, it will prefer to be assigned more jobs so as to earn more benefits. We assume that all jobs cannot be split; otherwise, the problem can be solved trivially. While we consider fairness as the decision maker's most critical target, naturally it also should not

[1]For more information, please see `http://www.meetup.com/thesockmob/`.

[2]For more information, please see `http://www.bigissue.tw/`.

sacrifice profitability too much. Therefore, we set up the objective function in our problem as to maximize the benefit generated by the machine generating the least benefit. While adopting this objective takes care the performance of the poorest machine, it also intuitively encourages the whole system to complete as many high-benefit jobs as possible.

As our problem is strongly NP-hard, we focus on designing a polynomial-time approximation algorithm. We follow the scheduling literature and modify the famous longest processing time first (LPT) algorithm to propose a greedy algorithm for our problem. We first sort jobs in the descending order of benefits, and then in each iteration assign a job to a feasible machine currently having the minimum benefit. This simple algorithm, which is called capacitated highest benefit first (CHBF), is then proved to possess worst-case performance guarantee in various cases.

The approximation factor critically depends on the relationship between job benefits and workloads. When benefits are linear to workloads, CHBF is shown to be a $\frac{1}{2}$-approximation algorithm.[3] When the relationship is convex (i.e., the marginal benefit increases in workload), the factor is a function depending on the number of jobs, number of machines, and degree of convexity. When the relationship is concave (i.e., the marginal benefit decreases in workload), the factor becomes depending on the machine capacity and degree of concavity. Both factors under the convex and concave scenarios are shown to approach $\frac{1}{2}$ when the convex and concave functions approach a linear one.

Beside our theoretical investigation, we also conduct numerical studies. These numerical studies serve for multiple purposes. First, it demonstrates that CHBF's average-case performance is much better than its worst-case performance guarantee in all situations. Moreover, it delivers insightful managerial implications of CHBF. Numerically we observe that CHBF's average-case performance is better under the convex benefit-workload relationship than under the linear one, which is then better than the concave one. This implies that a decision maker facing our job allocation problem may consider whether to apply CHBF according to the characteristics of its jobs/products. If its production environment is of economy of scale, e.g., due to cost saving through standardized mass production, CHBF would be a good choice. On the contrary, if the marginal benefit of producing products is decreasing, which is typical when the

[3]Throughout this paper, we follow the convention of Williamson and Shmoys (2011) and say that an α-approximation algorithm (or an algorithm of factor $\alpha \in (0,1)$) for a maximization problem always attains an objective value z such that $z \geq \alpha z^*$, where z^* is the objective value of an optimal solution.

market-clearing price decreases as more products are put onto the market, CHBF may be less effective.

In the next section, we will review some relevant literature about job scheduling and allocation, approximation algorithms, and allocation fairness. In Section 3, we will formulate our job allocation problem into an integer program and demonstrate its NP-hardness. The algorithm CHBF will then be introduced in Section 4. More importantly, the worst-case performance guarantees of CHBF will be proved in the same section. We conduct numerical studies in Section 5. Section 6 concludes.

2 Literature Review

All kinds of job allocation and scheduling problems have been widely studied in the literature. A unique feature of our job allocation problem is that each job has two attributes, workload and benefit, which may have various kinds of relationships. A special case of our problem is for machines to be uncapacitated. In this case, job workloads do not matter. If we consider job benefits as processing times, the problem reduces to maximizing minimum completion time on parallel identical machines, which is denoted as $P||C_{\min}$ according to the three-field notation in the scheduling literature (Pinedo, 2012; Walter et al., 2017). Applications of $P||C_{\min}$ include replacement part sequencing (Friesen and Deuermeyer, 1981), regional allocations of investments (Haouari and Jemmali, 2008), among others.

Some researchers tackle $P||C_{\min}$ by proposing exact algorithms by improving the generic branch-and-bound algorithms (Mokotoff, 2004; Haouari and Jemmali, 2008; Walter et al., 2017). More research are devoted to approximation algorithms or approximation schemes. As the "dual" version of $P||C_{\min}$, makespan minimization is one of the earliest problems for which approximation algorithms are developed (Williamson and Shmoys, 2011). It has been shown by Graham (1966, 1969) that the longest processing time first (LPT) algorithm has an asymptotic approximation factor $\frac{4}{3}$. Deuermeyer et al. (1982) show that the same algorithm can be adopted for $P||C_{\min}$ with a factor $\frac{3}{4}$. Csirik et al. (1992) improves the performance guarantee to $\frac{3m-1}{4m-2}$, where m is the number of machines. Walter (2013) propose a similar approximation algorithm called restricted LPT (RLPT) and prove that it has an approximation fact $\frac{1}{m}$. The author also shows that LPT always outperforms RLPT. Alon et al. (1998) prove that under some mild conditions a PTAS can be constructed. To the best of our knowledge, this study is the first to extend $P||C_{\min}$ to allocate benefit-generating jobs to capacitated machines. This generalization provides uniqueness to this study.

Sometimes researchers look for *a posteriori* bounds for NP-hard problems. For makespan minimization, Blocher and Sevastyanov (2015) improves the *a posteriori* Coffman-Sethi bound of LPT by considering the maximum number of jobs scheduled on a machine. Massabò et al. (2016) derive a tight *a posteriori* bound of LPT for the two-machine makespan minimization problem. This bound depends on the index of the last job assigned to the critical machine. Following their ideas, the bounds provided in our study are proved by considering the first job that cannot be directly assigned due to the capacity constraints. Nevertheless, for all scenarios we provide *a priori* bounds, not only *a posteriori* ones.

3 Model

We consider a problem of assigning n jobs in set $J = \{1, 2, ..., n\}$ to m machines in set $I = \{1, 2, ..., m\}$. The capacity of machine i is $K > 0$ for all $i = 1, ..., m$. For job j, there is a workload $c_j > 0$ and a benefit $b_j > 0$. With the assumption that a job cannot be split and assigned to multiple machines, we try to assign jobs to machines to maximize the minimum total benefit among the machines while the capacity constraints are satisfied. More precisely, let

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i \\ 0 & \text{otherwise} \end{cases}, i \in I, j \in J,$$

be our decision variables, our job allocation problem can be formulated as

$$\begin{aligned} \max \quad & \min_{i \in I} \left\{ \sum_{j \in J} b_j x_{ij} \right\} \\ \text{s.t.} \quad & \sum_{j \in J} c_j x_{ij} \leq K \quad \forall i \in I \\ & \sum_{i \in I} x_{ij} \leq 1 \quad \forall j \in J \\ & x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J. \end{aligned} \tag{1}$$

The first constraint ensures that the total workload of jobs assigned to machine i will not exceed its capacity K, the second constraint ensures that a job can only be assigned once, and the last constraint guarantees integrality. The objective function is to maximize the benefit generated by the machine generating the lowest benefit. Table 1 lists the notations used in this model.

Parameter	
I	The set of machines
J	The set of jobs
n	The number of jobs (in set J)
m	The number of machines (in set I)
K	The capacity of machine i ($K > 0$)
c_j	The finite workload of job j ($c_j > 0$)
b_j	The finite benefit of job j ($b_j > 0$)
Decision variable	
$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i \\ 0 & \text{otherwise} \end{cases}$	$, i \in I, j \in J.$

Table 1: List of notations

Two special cases of our problem have been shown to be NP-hard. First, when all the machines are uncapacitated, our problem becomes to $P||C_{\min}$, for which many researchers design approximation algorithms for (Alon et al., 1998; Csirik et al., 1992; Deuermeyer et al., 1982; Walter, 2013). Second, when there is only one machine, our problem is exactly a knapsack problem. Using either reduction shows that our problem is NP-hard.

Given a pair of jobs j_1 and j_2, we say that job j_1 is *larger* (respectively, *smaller*) than job j_2 if $c_{j_1} > c_{j_2}$. Without loss of generality, we may assume that $c_1 \geq c_2 \geq \cdots \geq c_n$. If the workload of the smallest job c_n is below 1, we may without loss of generality normalize it to 1 and adjust job workloads and machine capacity proportionally. After the normalization, if machine capacity is less than 2, one will be able to solve the problem in polynomial time (as a machine can only be assigned one job). Therefore, we exclude this uninteresting case and further assume that machine capacity is at least 2. Finally, it is clear that jobs whose workloads are above the machine capacity can be removed without affecting an optimal solution. We formally state these assumptions below.

Assumption 1. $K \geq c_1 \geq c_2 \geq \cdots \geq c_n \geq 1$ *and* $K \geq 2$.

The difficulty of our problem naturally depends on the relationship between job benefits b_j and workloads c_j. Intuitively, the problem is more complicated if benefits and workloads are not regulated by any relationship. Fortunately, in practice typically a larger job brings in a higher benefit. Therefore, we will restrict our analysis in the next section to this typical case, i.e., $c_{j_1} > c_{j_2}$ if and only if $b_{j_1} > b_{j_2}$ for all j_1, $j_2 \in J$. We further consider three special relationships between b_j and c_j: In the *linear* relationship, $b_j = hc_j$ for some $h > 0$; in the *convex* relationship, $b_j = hc_j^t$ for some $h > 0$ and $t > 1$; in the *concave* relationship, $b_j = hc_j^t$ for some $h > 0$ and $t < 1$. These assumptions are formalized below.

Assumption 2. *$b_j = hc_j^t$ for some $h > 0$ and $t > 0$ for all $j \in J$.*

4 Worst-case performance analysis

We propose a natural extension of the classic LPT algorithm to develop our algorithm. We first sort all jobs in a non-increasing order according to their benefits. Then we follow the order and in each iteration try to assign a job to a machine which has the lowest cumulative benefit. If a job cannot be assigned to a machine because the residual capacity on that machine is not enough, we try the machine with the second lowest cumulative benefit. If a job cannot be assigned to any machine, it is discarded. We repeat this process until the last job has been tried to be assigned. All ties are broken arbitrarily. For ease of exposition, we denote this algorithm as the capacitated highest benefit first (CHBF) algorithm. Throughout this section, we denote z^* as the objective value of an optimal solution and z' as that of the CHBF solution.

Below we will prove the worst-case performance guarantees of CHBF for $t = 1$, $t > 1$, and $t < 1$. All the three approximation ratios are no greater than $\frac{1}{2}$. Before we start the proofs, We first make a note. During the CHBF allocation process, if all jobs can be assigned to its first-priority machine without violating the capacity constraint, the resulting schedule will be the same as that obtained by the LPT rule as if the machines are uncapacitated. It then follows from Deuermeyer et al. (1982) that the minimum benefit of this schedule is at least $\frac{3}{4}$ of that of an optimal schedule, and the approximation ratios are indeed valid. Therefore, below we will only focus on the case that at least one job cannot be assigned to a machine that has the minimum benefit.

We start our analysis of CHBF's worst-case performance guarantee when $t = 1$, i.e., the benefit-workload relationship is linear. In Theorem 1, we show that CHBF is guaranteed to generate a solution that is at least one half as good as an optimal solution.

Theorem 1. *If $b_j = hc_j$ for all $j \in J$ for some $h > 0$, we have $z' \geq \frac{1}{2}z^*$.*

Proof. First, an obvious upper bound of z^* is hK, as no machine may earn more than hK as its benefit. If we may prove that $z' \geq \frac{1}{2}hK$, we will have $z' \geq \frac{1}{2}hK \geq \frac{1}{2}z^*$, and the proof is complete. Suppose that $z' < \frac{1}{2}hK$, and job j is the first job not assigned to its first-priority machine, say, machine i. Because $z' < \frac{1}{2}hK$, the cumulative benefit of machine i must also be lower than $\frac{1}{2}hK$ at the time of assigning job j. This implies that $b_j < \frac{1}{2}hK$, otherwise, by CHBF job j should have been assigned before any job assigned to machine i. Therefore, $c_j < \frac{1}{2}K$. However, if $z_i < \frac{1}{2}hK$, the total workloads of jobs that have been assigned to machine i must be less than $\frac{1}{2}K$. This means the residual capacity of machine i is greater than $\frac{1}{2}K$, and there is a room to assign job j. This violates our assumption that job l cannot be assigned to machine i. With this contradiction, the proof is complete. □

The following example shows that the bound $\frac{1}{2}$ is tight. Let ϵ be a small positive number. Suppose that we have $2m$ jobs with workload $\frac{1}{2}K$ and 1 job of workload $\frac{1}{2}K + \epsilon$. Let $h = 1$. CHBF will result in $m - 1$ machines being allocated 2 jobs but one machine only 1 job. The machine with only 1 job will earn benefit $z' = \frac{1}{2}K + \epsilon$. However, the optimal solution is to allocate each machine 2 jobs by ignoring the largest job. This results in $z^* = K$. As the result, $\frac{z'}{z^*} = \frac{\frac{1}{2}K+\epsilon}{K}$, which approaches $\frac{1}{2}$ as K approach infinity.

We next consider the case when job benefits are convex to workloads. We first derive an *a posteriori* worst-case performance guarantee, which depends on the number of jobs assigned on the minimum-benefit machine of the first job that cannot be assigned to its first priority by CHBF.

Definition 1. *Suppose that job j is the first job that cannot be assigned to its first-priority machine by CHBF. We define p as the number of jobs that have been assigned to that minimum-benefit machine.*

The main result is stated below.

Theorem 2. *If $b_j = hc_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have $z' \geq \frac{p}{(p+1)^t}z^*$, where p is defined in Definition 1.*

Proof. First of all, note that hK^t is an upper bound of z^*, which happens when each machine contains one job of workload K. Below we prove $z' \geq \frac{p}{(p+1)^t}hK^t$, which is again done by considering the moment when we cannot assign job j to the minimum benefit machine i for the first time.

We prove by contradiction by assuming that that $z' < \frac{p}{(p+1)^t} hK^t$. This implies that at that moment machine i's benefit is also lower than $\frac{p}{(p+1)^t} hK^t$. Because job j cannot be assigned to machine i, which has already been assigned p jobs by CHBF, we have $b_j \leq [\frac{p}{(p+1)^t} hK^t]/p = \frac{hK^t}{(p+1)^t}$, where the equality holds if and only if all the p jobs are equally large. This implies that $c_j < \frac{K}{p+1}$, which then implies that the consumed capacity on machine i is at least $K - \frac{K}{p+1} = \frac{pK}{p+1}$, otherwise machine i would have enough capacity to be assigned job j. Due to the characteristic of convexity, the least beneficial way to consume that amount of capacity is for the p jobs to be equally large. In this case, machine i's benefit is

$$ph\left(\frac{\frac{pK}{p+1}}{p}\right)^t = \frac{p}{(p+1)^t} hK^t,$$

which contradicts to our assumption. Therefore, we must have $z' \geq \frac{p}{(p+1)^t} hK^t \geq \frac{p}{(p+1)^t} z^*$. $\square$

Through an investigation on the lower bound of $\frac{p}{(p+1)^t}$, we may further obtain an *a priori* approximation ratio.

Theorem 3. *If $b_j = hc_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have*

$$z' \geq \min\left\{\frac{1}{2^t}, \frac{n-m}{(n-m+1)^t}\right\} z^*.$$

Proof. All we need to do is to show that $\min\{\frac{1}{2^t}, \frac{n-m}{(n-m+1)^t}\} \leq \frac{p}{(p+1)^t}$. It can be easily shown that $\frac{p}{(p+1)^t}$ is quasi-concave in p for any $t > 1$. As $1 \leq p \leq n-m$, the global minimizer of $\frac{p}{(p+1)^t}$ is either 1 or $n-m-1$. Plugging these two values into $\frac{p}{(p+1)^t}$ completes the proof. $\square$

The *a priori* lower bound of z' is the minimum of $\frac{1}{2^t}$ and $\frac{n-m}{(n-m+1)^t}$. If $t \geq 2$, we have $\frac{n-m}{(n-m+1)^t} < \frac{1}{2^t}$ because $\frac{p}{(p+1)^t}$ is decreasing in that case. However, as long as $t < 1$, $\frac{p}{(p+1)^t}$ is increasing at $p = 1$, and it is possible that $\frac{n-m}{(n-m+1)^t} > \frac{1}{2^t}$. More importantly, when t is sufficiently close to 1, $\frac{1}{2^t}$ will be the smaller one (as long as $n-m > 1$), and the approximation factor converges to $\frac{1}{2}$ as Theorem 1 suggests (for $t = 1$). This means that Theorem 3 actually extends Theorem 1 from $t = 1$ to $t \geq 1$. Finally, note that $\frac{1}{2^t} < \frac{1}{2}$ if $t > 1$, which implies that the approximation ratio is below $\frac{1}{2}$.

Lastly, we characterize the performance guarantee when the benefit is concave in workload.

Theorem 4. *If $b_j = hc^t$ for all $j \in J$ for some $h > 0$ and $t < 1$, we have $z' \geq \frac{K^{t-1}}{2^t} z^*$.*

Proof. We follow the same idea applied in the proofs of Theorems 1 and 2. First we establish an upper bound of z^*. When the relationship between job benefits and workloads are concave, the most beneficial way to consume all the capacity of one machine is to use K jobs with unit workload. The benefit of each machine hK in this case is an upper bound of z^*.

Now we consider the moment that CHBF first fails to assign a job to its first-priority machine. Let that job be job j and that machine be machine i. We prove by contradiction by assuming that that $z' < h\frac{K^t}{2^t}$. This implies that at this moment machine i's benefit is also lower than $h\frac{K^t}{2^t}$, and therefore the benefit of job j should be less than that as well (otherwise, it should have been assigned by CHBF). As a result, we know $c_j < \frac{K}{2}$, and the residual capacity of machine i is less than $K - \frac{K}{2} = \frac{K}{2}$, otherwise there would be a room to assigned job j to machine i. For machine i, however, the fact that its current benefit is lower than $h\frac{K^t}{2^t}$ implies that the currently occupied capacity must be at most $\frac{K}{2}$ (which happens when machine i is assigned only one job). This immediately implies that the residual capacity is above $\frac{K}{2}$, which contradicts with the fact derived above. Therefore, we have $z' \geq h\frac{K^t}{2^t} = \frac{K^{t-1}}{2^t}hK \geq \frac{K^{t-1}}{2^t}z^*$. $\square$

Note that $t < 1$ and $K \geq 2$ imply that $\frac{K^{t-1}}{2^t} = \frac{1}{2^t K^{1-t}} \leq \frac{1}{2^t \cdot 2^{1-t}} = \frac{1}{2}$, which means that the performance guarantee is no greater than $\frac{1}{2}$. Moreover, as t approaches 1, the guarantee approaches $\frac{1}{2}$. This indicates that Theorem 4 actually extends Theorem 1 from $t = 1$ to $t \leq 1$.

5 Numerical studies

To understand how CHBF performs on our job allocation problem, we consider the following factors. The first factor is the relationship between job benefits and workloads. We consider four scenarios. In each of the first three scenarios, job benefit is linear, convex, and concave to the job workload, respective. In the last scenario, benefits and workloads are unrelated. The second factor is machine capacity. We consider three scenarios, each with unlimited, loose, and tight capacity. The third factor is straightforward: the number of machines and jobs representing instance scale. We adopt the following setting for each scenarios:

- Benefit-workload relationship: scenario L (for linear): $b_j = c_j$; scenario X (for convex): $b_j = c_j^2$; scenario A (for concave): $b_j = \sqrt{c_j}$; scenario R (for random): b_j is uniformly distributed in 0 and 50.
- Machine capacity: scenario N (for unlimited): $K = \infty$; scenario L (for loose): $K = \left(\frac{\sum_{j \in J} c_j}{m}\right)$; scenario T (for tight): $K = \frac{3}{4}\left(\frac{\sum_{j \in J} c_j}{m}\right)$.
- Instance scale: $(m, n) = (5, 20)$, $(5, 50)$, $(5, 500)$, $(15, 50)$, $(15, 500)$, and $(50, 500)$.

These three factors together generate 48 scenarios for numerical experiments. For each scenario, we generate 100 random instances by having c_j to be uniformly distributed in 0 and 50. For each instance, we use the branch-and-bound algorithm to find a solution if $(m, n) = (5, 20)$ or, due to memory limitation, solve the linear relaxation of the original problem otherwise. We also apply CHBF to generate a CHBF solution.

While the main focus of this study is to solve the fair allocation problem in (1), it is still important to ask to what degree efficiency is sacrificed when we maximize fairness. Therefore, we consider the efficiency problem by replacing the objective function in (1) by

$$\max \quad \sum_{i \in I} \sum_{j \in J} b_j x_{ij},$$

which is the total benefits earned by all machines. Note that this problem is a special case of the well-known multidimensional knapsack problem (Fréville, 2004). We call our main problem in (1) the *fairness problem* and the one with the alternative objective function the *efficiency problem*. For the fairness problem, we use w^{IP} and w^{CHBF} to denote the objective value for an optimal solution (or the solution to its linear relaxation when necessary) or an CHBF solution, respectively. Beside investigating how the CHBF solution deviates from a fairest solution (i.e., an optimal solution to (1)), we will also examine the efficiency gap between our reported solution and an optimal solution to the efficiency problem. Therefore, for the efficiency problem, we use z^{IP} and z^{CHBF} to denote the total revenues earned by an optimal solution (or the solution to its linear relaxation when necessary) or an CHBF solution, respectively.

The complete computational results are listed in Table 5 in the appendix. By averaging across all rows, we obtain that the average-case performance of CHBF on fairness is around 96%. While CHBF generates a good solution to the fair job allocation problem in (1), which is our main focus, it also attains around 98% of efficiency in average. We next process to examine how each factor affects the performance of CHBF. To do so, we average across the same levels of each factor to generate Tables 2, 3, and 4. For example, to generate the first row of Table 2, we take the average of all the 24 rows whose capacity are of level "N" in Table 5. Below we will concentrate on the fairness issue in our discussions. The column of efficiency performance in each table is for readers' reference.

Consider the capacity factor first. In Table 2, we observe that CHBF performs better if the machine capacity is looser (as $\frac{w^{CHBF}}{w^{IP}}$ increases). This is intuitive because a tighter capacity constraint require a more careful allocation of jobs to fully utilize the capacity. The CHBF algorithm, which is of a greedy nature, then becomes more unlikely to find the best way to allocate jobs. Nevertheless, even when the capacity is tight, in average CHBF can still be around 91% as good as an optimal solution.

Capacity	$\frac{z^{CHBF}}{z^{IP}}$	$\frac{w^{CHBF}}{w^{IP}}$
N	1	0.988
L	0.987	0.964
T	0.943	0.912

Table 2: Impact of machine capacity

We next investigate the impact of instance scale, n and m. While there is no obvious impact of n and m on the performance of CHBF, Table 3 reveals that CHBF performs better when $\frac{n}{m}$ becomes bigger. To understand this, note that a larger $\frac{n}{m}$ typically means that in an optimal solution there are more jobs assigned to a machine. This leaves a larger room for a greedy algorithm to make some errors.

$\frac{n}{m}$	$\frac{z^{CHBF}}{z^{IP}}$	$\frac{w^{CHBF}}{w^{IP}}$
100	0.983	0.980
33.333	0.983	0.977
10	0.981	0.965
4	0.971	0.942
3.333	0.961	0.899

Table 3: Impact of the ratio of number of jobs to number of machines

Finally, by observing Table 4, we find that CHBF in average can achieve around 98% of a fairest schedule when the benefit-workload relationship is linear. It also has a similar performance when the relationship is convex. If job benefits and workloads are unrelated, naturally it is hard for CHBF to find the best structure of an instance. The performance then drops to around 95% (though the efficiency is still high). Interestingly, CHBF performs the worst when the relationship is concave. The reason behind this fact is as follows. Recall that CHBF assigns jobs by first sorting jobs' according to their benefits. When benefits are concave in workloads, a high-benefit job is actually not "workload-effective," as its benefit-workload ratio $\frac{b_j}{c_j}$ is smaller than a low-benefit job. In fact, when CHBF sorts jobs in the descending order of b_j, it assigns those jobs of low $\frac{b_j}{c_j}$ before doing so to those of high $\frac{b_j}{c_j}$. While this may be fine when there is ample capacity, this wastes a lot of valuable capacity when capacity is tight. This explains the bad average performance, both for fairness and efficiency. On the contrary, a high benefit implies a high benefit-workload ratio when job benefit is convex to workload, and thus CHBF's performance can be quite good.

benefit-workload	$\frac{z^{CHBF}}{z^{IP}}$	$\frac{w^{CHBF}}{w^{IP}}$
L	0.990	0.979
X	0.988	0.976
R	0.992	0.947
A	0.936	0.918

Table 4: Impact of the benefit-workload relationship

Note that the convexity or concavity of the benefit-workload relationship has an important managerial implication. When the relationship is convex, basically the production environment is of significant economy of scale so that the marginal benefit increases as workloads increase. On the contrary, when the relationship is concave, basically the environment is of diminishing marginal benefit. This understanding will help practitioners to decide whether CHBF is a good decision-support tool if they face the job allocation problem studied in this paper.

6 Conclusions

In this study, we consider a job allocation problem with allocation fairness as the main focus. We formulate the problem as an integer program to maximize the lowest benefit among all machines subject to capacity constraints. By modifying the classic LPT rule, which is known to be effective for the uncapacitated version of our problem, we develop our own algorithm CHBF. We then prove the performance guarantee is $\frac{1}{2}$ when the job benefit is linear to workload. For convex and concave relationships, we also establish different performance guarantees depending on the number of jobs, number of machines, degree of convexity (or concavity), and machine capacity. The average-case performance is demonstrated to be much better than the worst-case performance guarantee in each scenario through numerical studies. Moreover, we show that CHBF is more appropriate when the machine capacity is loose than tight, the ratio of number of jobs to number of machines is large than small, and when the production environment exhibits economy of scale than diminishing marginal benefit.

Some further investigations may further improve this study. One promising direction is to look for better worst-case performance guarantees of our problem, either for the general problem or under some conditions. Another direction is to consider a different algorithm. For example, as we have pinpointed the reason for CHBF to be not so effective when job benefit is convex to workload, we may want to sort jobs in the descending order of their benefit-workload ratio. This calls for further investigation.

References

Alon, N., Y. Azar, G.J. Woeginger, T. Yadid. 1998. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling* **1**(1) 55–66.

Blocher, J.D., S. Sevastyanov. 2015. A note on the Coffman-Sethi bound for LPT scheduling. *Journal of Scheduling* **18**(3) 325–327.

Csirik, J., H. Kellerer, G. Woeginger. 1992. The exact LPT-bound for maximizing the minimum completion time. *Operations Research Letters* **11**(5) 281–287.

Deuermeyer, B.L., D.K. Friesen, M.A. Langston. 1982. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Algebraic Discrete Methods* **3**(2) 190–196.

Fréville, A. 2004. The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research* **155**(1) 1–21.

Friesen, D.K., B.L. Deuermeyer. 1981. Analysis of greedy solutions for a replacement part sequencing problem. *Mathematics of Operations Research* **6**(1) 74–87.

Graham, R.L. 1966. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal* **45**(9) 1563–1581.

Graham, R.L. 1969. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics* **17**(2) 416–429.

Haouari, M., M. Jemmali. 2008. Maximizing the minimum completion time on parallel machines. *4OR: A Quarterly Journal of Operations Research* **6**(4) 375–392.

Massabò, I., G. Paletta, A.J. Ruiz-Torresh. 2016. A note on longest processing time algorithms for the two uniform parallel machine makespan minimization problem. *Journal of Scheduling* **19**(2) 207–211.

Mokotoff, E. 2004. An exact algorithm for the identical parallel machine scheduling problem. *European Journal of Operational Research* **152**(3) 758–769.

Pinedo, M. 2012. *Scheduling Theory, Algorithms, and Systems*. Springer, Berlin, Germany.

Walter, R. 2013. Comparing the minimum completion times of two longest-first scheduling-heuristics. *Central European Journal of Operations Research* **21**(1) 125–139.

Walter, R., M. Wirth, A. Lawrinenko. 2017. Improved approaches to the exact solution of the machine covering problem. *Journal of Scheduling* **20**(3) 147–164.

Williamson, D.P., D.B. Shmoys. 2011. *The Design of Approximation Algorithms*. Cambridge University Press, London, UK.

Appendix: the complete computational results

m	n	Capacity	B-W	$\frac{z^{CHBF}}{z^{IP}}$	$\frac{w^{CHBF}}{w^{IP}}$	m	n	Capacity	B-W	$\frac{z^{CHBF}}{z^{IP}}$	$\frac{w^{CHBF}}{w^{IP}}$
5	20	L	L	0.96	0.94	15	50	L	L	0.946	0.903
5	20	L	A	0.958	0.962	15	50	L	A	0.932	0.863
5	20	L	X	0.982	0.975	15	50	L	X	0.971	0.904
5	20	L	R	0.987	0.91	15	50	L	R	0.987	0.859
5	20	T	L	0.963	0.945	15	50	T	L	0.969	0.943
5	20	T	A	0.845	0.822	15	50	T	A	0.876	0.823
5	20	T	X	0.988	0.991	15	50	T	X	0.89	0.788
5	20	T	R	0.967	0.874	15	50	T	R	0.962	0.831
5	20	N	L	1	0.971	15	50	N	L	1	0.965
5	20	N	A	1	0.959	15	50	N	A	1	0.956
5	20	N	X	1	0.987	15	50	N	X	1	0.983
5	20	N	R	1	0.972	15	50	N	R	1	0.966
5	50	L	L	0.993	0.989	15	500	L	L	1	0.999
5	50	L	A	0.985	0.97	15	500	L	A	0.998	0.996
5	50	L	X	0.999	0.996	15	500	L	X	1	1
5	50	L	R	0.998	0.97	15	500	L	R	1	0.987
5	50	T	L	0.992	0.987	15	500	T	L	1	0.999
5	50	T	A	0.829	0.815	15	500	T	A	0.81	0.808
5	50	T	X	0.978	0.97	15	500	T	X	0.995	0.994
5	50	T	R	0.989	0.925	15	500	T	R	0.992	0.943
5	50	N	L	1	0.995	15	500	N	L	1	1
5	50	N	A	1	0.989	15	500	N	A	1	0.999
5	50	N	X	1	0.999	15	500	N	X	1	1
5	50	N	R	1	0.995	15	500	N	R	1	1
5	500	L	L	1	1	50	500	L	L	0.996	0.993
5	500	L	A	1	0.999	50	500	L	A	0.99	0.972
5	500	L	X	1	1	50	500	L	X	1	0.998
5	500	L	R	1	0.996	50	500	L	R	1	0.956
5	500	T	L	1	1	50	500	T	L	0.996	0.992
5	500	T	A	0.805	0.804	50	500	T	A	0.821	0.8
5	500	T	X	0.998	0.998	50	500	T	X	0.986	0.978
5	500	T	R	0.988	0.966	50	500	T	R	0.993	0.9
5	500	N	L	1	1	50	500	N	L	1	0.995
5	500	N	A	1	1	50	500	N	A	1	0.98
5	500	N	X	1	1	50	500	N	X	1	0.999
5	500	N	R	1	1	50	500	N	R	1	0.996

Table 5: The average performance of CHBF