國立臺灣大學管理學院資訊管理學系

碩士論文

Department of Information Management
College of Management
National Taiwan University
Master Thesis

考慮公平性之工作分配問題

Job Allocation with a Consideration of Fairness

劉騏瑋

Chi-Wei Liu

指導教授：孔令傑 博士

Adviser: Ling-Chieh Kung, Ph.D.

中華民國 105 年 9 月

September, 2016

doi:10.6342/NTU201603583

國立臺灣大學(碩、博)士學位論文
口試委員會審定書

題目：Job Allocation with a Consideration of Fairness

本論文係劉騏瑋君（學號 R03725034）在國立臺灣大學資訊管理學系、所完成之（博、碩）士學位論文，於民國 105 年 6 月 21 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

所　　長：

謝辭

碩士生活即將告一段落，完成這篇論文，最大的功臣莫過於我的指導教授孔令傑老師。當初決定請孔老師指導之後，老師儘管平日已公務繁忙，仍義不容辭的在大學部四下時就開始指導我與另外一位將進入同研究室的同學—偉宏，加強我們對於作業研究領域的認識。升上研究所後，老師也指導我一同參與研究計畫，不只是學習如何應用所學在各種領域，這也對於我的研究也有不少的助益。除此之外，老師對於研究上仔細的態度與不厭其煩的個性，一直讓我非常的欽佩。老師總是很細心地檢查我們的文章與簡報，讓我們能表現的更加專業，對於我的問題以及常犯的錯誤，也都能一次又一次和氣的講解或糾正。真的很感謝如此亦師亦友的老師，能接受老師的指導，我感到非常榮幸。此外，特別感謝黃奎隆及洪一薰兩位教授們的寶貴建議，讓我的論文內容能更加完整。

兩年的研究生生活，有一起做同一個領域一起崩潰的偉宏，講到如何最佳化設點策略交給你就行；有同樣來自南部的冠宇與何禾，偶爾的研究室深夜聊天，總讓人又充滿能量繼續隔天的研究生活。不可少的三位好夥伴，真的很感謝有你們的相伴。學弟妹們，雪兒、阿波、韋志、Fifi、Jeff、千瑜、宸安、意惠，很感謝有你們的加入，讓 IEDO Lab 成了研究生生活的一個家，也要特別感謝雪兒在研究上的協助，是這篇論文的幕後功臣。在台大的四加兩年學生生活，也很謝謝陪伴著我的好同學們，不管是在學業或是各種活動上，因為有你們一起歡笑、一起辛苦，讓我這一路留下許多美好的回憶。

最後，要感謝能讓我無後顧之憂的家人們，謝謝你們尊重我的各種決定並支持我北上求學，我會永遠記得，在我需要幫助時，台南有個我最珍愛的家。

劉騏瑋　謹啟

于台大資訊管理研究所

民國一百零五年九月

doi:10.6342/NTU201603583

摘要

工作分配在各領域中一直是一個重要議題。在製造業中，其中一個目標是希望能最大化最小利潤的機台。然而，完成工作除了能賺取利潤外，亦會為機台帶來工作量的負擔。這樣的特性，讓工作分配更為困難，儘管希望能多分配工作給機台使其利潤更高，但受到工作量的限制，機台僅能負擔一定工作量。我們希望能設計演算法，在兼顧公平性及工作量負擔下，有效分配工作。

在本研究中，我們考慮若干工作及若干機台，完成工作會帶來利潤，但也會為機台帶來工作量，各機台能負荷的最大工作量是一樣的。為了使各機台的利潤盡量地一致，我們的目標是最大化得到最小利潤的機台上的利潤，根據前人的最長處理時間優先演算法（LPT rule），我們提出兩個演算法。

我們證明了我們的第一個演算法在工作利潤與工作量有比例關係時，表現最差的情況會是$\frac{1}{2}$。除此之外，若工作利潤與工作量是凸函數或凹函數的關係時，我們亦能證明其存在最差表現的保證。透過數值研究分析，我們發現當工作環境有規模經濟效應時，採取第一種演算法表現較好；反之，若工作利潤與工作量呈現邊際效應的遞減，採取第二種演算法較好。了解這個發現後，我們便能知道該在何種環境使用哪一種演算法能讓工作分配較佳。

關鍵字：工作分配、近似演算法、公平性。

doi:10.6342/NTU201603583

Abstract

Job scheduling is an important issue applied in many fields. In the manufacturing industry, one of the objectives is to assign jobs to machines in order to maximize the minimum profit among machines. Nevertheless, jobs not only bring profits, but also bring in workloads. Each machine cannot be assigned too many jobs due to limited capacity. This characteristic introduces a new challenge to this allocation problem. We would like to propose efficient algorithms to assign jobs by taking fairness issue into consideration.

In this study, we consider the aforementioned job allocation problem. Our objective is to assign jobs to bring benefits to all the machines as equally as possible while ensuring that machines cannot be overloaded. In our model, we formulate this problem as a job assignment problem in which a set of jobs is to be assigned to a set of machines to maximize the benefit obtained by the machine earning the minimum benefit. The capacity of all machines are the same. We propose two list scheduling algorithms based on the longest processing time (LPT) rule.

We show that the first algorithm guarantees a $\frac{1}{2}$ worst-case performance when the job benefit is proportional to job workload. Moreover, our algorithm can have performance guarantees when the relationship between job benefits and workloads is convex or concave. Through numerical experiments, we also show that the algorithm works well when the jobs exhibit economy of scale but not so well when the jobs exhibit diminishing marginal benefits. The second algorithm is then shown to complement the first one in the latter case. Knowing this result, we can decide which algorithm to apply according to the environment.

Keywords: Job scheduling, Approximation algorithms, Fairness.

doi:10.6342/NTU201603583

Contents

doi:10.6342/NTU201603583

doi:10.6342/NTU201603583

List of Tables

doi:10.6342/NTU201603583

doi:10.6342/NTU201603583

Chapter 1

Introduction

1.1 Background and motivation

Job scheduling has been widely discussed over decades and applied in many fields. In the manufacturing industry, a company may want to make products as quickly as possible in order to earn more profits. For this purpose, the factory manager needs to design a production schedule to minimize the completion time. In some complicated environments, there may even be multiple stages for making products. Examples and more details are provided in Pinedo (2012). In the field of computer science, the assignment of jobs is a major concern of designing multi-processor systems. For examples, people may want to deal with minimizing average loading or design a schedule which can always perform well with unknown jobs appearing at any time. Beside manufacturing and computer science, there are also many job scheduling problems in one's daily life. All of these make job scheduling and allocation an interesting problem.

doi:10.6342/NTU201603583

In many cases, one needs to assign jobs to multiple factories, machines, or agents. Interestingly, while jobs bring workloads, they may still be welcome because people may earn profits by completing jobs. Therefore, the issue of allocation fairness emerges. As an example, a leading LED chip manufacturer in Taiwan, who owns around twenty factories in Taiwan and China, faced exactly this issue. Once per month, the company must assign existing jobs for producing different kinds of LED chips to multiple factories. Because factory managers are evaluated according to the amount of revenues generated by completing jobs (as well as other performance indicators), they really care about the number of jobs assigned to them. From this perspective, jobs are actually valuable *resources* that should be fairly assigned to factories. Unlike traditional resource allocation problems, however, in a job allocation problem, a factory manager also cannot accept too many jobs due to the associated workloads and one's capacity limitation. Job allocation is thus of new challenges and deserves careful investigations.

Some social enterprises nowadays also face fair allocation problems. With the idea of "giving them jobs, not money," a social enterprise (or sometimes a government) may hire jobless or even homeless people to do some part-time jobs so that they may earn their livings. For example, "The Sock Mob Homeless Volunteer Network" in London finds that the homeless is 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

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

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

doi:10.6342/NTU201603583

and revenue maximization. Instead, their objective is to bring benefits to those people who need jobs and to distribute the benefits generated by job completion to the people as equally as possible. In other words, the focus in not on *efficiency* but on *fairness*.

The emerging needs of fair allocation motivates us to study a job allocation problem with fairness as the main consideration. We would like to design an effective and efficient procedure which can fairly allocate jobs without violating the capacity constraint of each factory/machine/agent. We will theoretically prove a bound of our procedure to ensure its worst-case performance guarantee. Numerical experiments will then be conducted to demonstrate its average performance. Based on these analyses, we eventually draw managerial implications.

1.2 Research objectives

In this study, we consider the aforementioned job allocation problem. For ease of exposition, we will call those working agents as machines and examine the problem of assigning jobs to machines. The most important feature of our job allocation problem is that the decision maker should consider not only the overall profitability but also fairness among machines. In the environment, each machine has a limited capacity and can afford only a certain amount of 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 profits by completing jobs. Also, all jobs cannot be split to be assigned (i.e., the benefit is generated only when a job is fully completed by one machine). The objective function in our problem is set to maximize the minimum benefit generated by a machine.

doi:10.6342/NTU201603583

To approach this problem, our first step is to show its NP-hardness. Therefore, we focus on obtaining an approximation solution through a polynomial-time 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 then explicitly characterize its worst-case performance guarantee for various versions of our job allocation problem.

Besides finding a worst-case performance guarantee, we hope to understand how our algorithm performs in general and under what conditions it is effective. To this aim, we conduct numerical experiments to test the performance of our algorithm under different scenarios.

1.3 Research plan

In the next chapter, we will review some relevant literature about job scheduling and allocation, approximation algorithms, and allocation fairness. In Chapter 3, we will describe our optimization problem. The NP-hardness of the problem, the algorithm for solving the problem, and the worst-case performance guarantees of our algorithm are then provided in Chapter 4. We conduct numerical studies in Chapter 5 to test our algorithm's performance in practice. From the results, we then give managerial insights and suggestions for environments in which the algorithm are appropriate to be adopted. Chapter 6 concludes.

doi:10.6342/NTU201603583

Chapter 2

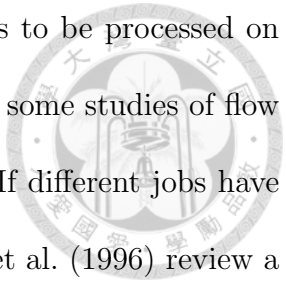
Literature Review

2.1 Scheduling and job allocation

Pinedo (2012) classifies job scheduling problems into two classes, those with a machine-based objective function and those with a job-based one. In the former, most studies try to maximize the minimum or minimize the maximum completion time of a machine. Some people may add weights to machines in their problems. In the latter, four typical criteria are are the lateness, tardiness, completion time, and flow time of jobs. Researchers typically minimize one of these four criteria as the objective function.

There may be different settings of the relationship between jobs and machines. For one kind of problem, each job has only one stage. Gupta and Kyparisis (1987) review scheduling problems that only has a single machine involved. If there are more than one machine that can process jobs in the system, it is categorized as parallel machine problem. Jobs in the two types of problems can be assigned to any machines in the environment.

doi:10.6342/NTU201603583



However, for another kind of problem, each job has multiple stages to be processed on predetermined machines. Ruiz and Vzquez-Rodrguez (2010) review some studies of flow shop problem and discuss their methods and solution approaches. If different jobs have different processing orders, it is called job shop problem. Blazwicz et al. (1996) review a variety of studies and discuss the solution techniques of job shop problem.

Based on the properties of jobs, the scheduling problem can be separated into two kinds. In an off-line problem, the jobs and their parameters are all given at the moment of planning. In contrast, in an on-line problem, the jobs will appear at random moments, and relevant information will be known by the decision maker only after the job appears. A survey of on-line problems is done by Fiat and Woeginger (1998).

According to the above classification principles, our problem is similar to a minimum completion time maximization off-line problem on parallel machines. A unique feature of our problem is that each job has two attributes, workload and benefit. We maximize the minimum benefit among all machines while satisfying the capacity constraints. The two-attribute setting makes our problem unique in the literature.

2.2 Approximation algorithms

Most of the scheduling problems are NP-hard. By the definition of NP-hardness, we know that we cannot solve the problem in polynomial time if $P \neq NP$. As a result, researchers look for polynomial-time heuristic algorithms for near-optimal solutions. When an algorithm has a worst-case performance guarantee, it is called an approximation algorithm (Williamson and Shmoys, 2011).

doi:10.6342/NTU201603583

Graham (1966, 1969) studies a well-known minimum makespan problem for multiple identical machines. He develops a listing algorithm, the LPT (longest processing time first) algorithm. At the beginning, it sorts all jobs in the descending order according to their processing times. Then, it picks a job once a time and assigns it to the currently least loaded machine (i.e., the machine currently having the earliest completion time). He proves that the algorithm has a bound $\frac{4}{3}$.

Deuermeyer et al. (1982) show that the same algorithm can be adopted for a "dual" version of the minimum makespan problem. The objective function becomes maximizing the minimum completion time among all machines. To prove by contradiction, they assume the existence of a counterexample and use weighting function to demonstrate its impossibility. The performance guarantee is shown to be $\frac{3}{4}$. Csirik et al. (1992) goes further from Deuermeyer et al. (1982) and use the same method to show that the performance guarantee can be improved to $\frac{3m-1}{4m-2}$, where m is the number of machines, which converges to $\frac{3}{4}$ as the number of machines approaches infinity.

For the same problem, Alon et al. (1998) prove that a PTAS (polynomial time approximation scheme) can be developed under some mild assumptions. By dividing jobs into two groups (big jobs and small jobs), they reduce the original instance into a new instance according to the processing time of each job. They obtain the solution of the new instance and show how to use that to construct a solution for the original instance. At last, they prove that given any arbitrary value of $\epsilon > 0$, the algorithm can be has a $1 + \epsilon$ bound.

Some researchers look for *a posteriori* bounds for the makespan minimization problem. In particular, Blocher and Sevastyanov (2015) improves the *a posteriori* Coffman-Sethi

doi:10.6342/NTU201603583

bound by considering the maximum number of jobs on a machine rather than the number of jobs on the critical machine. Also focusing on *a posteriori* bounds, Massab et al. (2016) characterize a tight bound for a two-machine 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, our bounds are *a priori*, not *a posteriori*.

In summary, we develop an approximation algorithm for our job allocation problem. Unlike the above scheduling problems studied in the literature, our problem exhibits the feature of job benefits. The resulting fairness issue differentiates this study from most previous studies.

2.3 Fairness

Fairness is an important issue for many allocation problem. To achieve fairness, the first question is always about how to measure the degree of fairness in the problem. We review some fairness indicators from past studies.

Bansal and Sviridenk (2006) discuss the so-called Santa Claus problem. In this problem, a decision maker tries to give presents to kids. Each kid has his own preference for different presents. In the Santa Claus problem, they maximize the utility of the kid with the minimum utility obtained from the allocated presents.

Bertsimas et al. (2011) study two classic fairness schemes, proportional fairness and max-min schemes. In the former setting, a schedule is said to be fair if no reallocation can result in a new schedule that is proportionally fairer. The definition of proportional

doi:10.6342/NTU201603583

fairness can be explained by the following example. Suppose that in a schedule we assign job 1 to machine 1 and job 2 to machine 2 and the benefits of machines 1 and 2 are 4 and 2, respectively. If exchanging the two jobs makes the benefits become 3 and 3, then the aggregate proportion change is $\frac{3-4}{4} + \frac{3-2}{2} > 0$. We then say that the new schedule is proportionally fairer. In the setting of max-min fairness, one needs to do a sequence optimization steps to maximize the minimum utility. At first, she maximizes the minimal utility of a player and ensures the utility of others all achieve that level. And then, he maximizes the second minimal utility of a player in the same way. This procedure will be repeated until the schedule cannot be improved. The resulting schedule will be considered a fair schedule.

In our problem, our fairness indicator is the same as Deuermeyer et al. (1982). We focus on the machine with the minimum total benefit and attempt to maximize its benefit. However, we will take job workload and machine capacity into consideration at the same time, which makes the problem more complicated. It is also clear that our objective function can be viewed as a special case of that studied by Bertsimas et al. (2011).

doi:10.6342/NTU201603583

doi:10.6342/NTU201603583

Chapter 3

Problem Description and Formulation

3.1 Model

In this study, we consider a problem of assigning n jobs (in set J) to m machines (in set I). 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

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

doi:10.6342/NTU201603583

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

$$\begin{aligned} \max \quad & \min_{i \in I} \left\{ \sum_{j \in J} b_j x_{ij} \right\} \\ \text{s.t.} \quad & \sum_{j \in J} c_j x_{ij} \leq K \quad \forall i \in I \\ & \sum_{i \in I} x_{ij} \leq 1 \quad \forall j \in J \\ & x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J. \end{aligned} \tag{3.1}$$

In this problem, the first constraint ensures that the total workload of jobs assigned to machine i will not exceed its capacity K_i, the second constraint ensures that a job can only generate its benefit once, and the last constraint guarantees integrality. The objective function is to maximize the benefit generated by the machine generating the lowest benefit.

While the main focus of this study is to solve the fair allocation problem in (3.1), it is still important to ask to what degree efficiency is sacrificed when we optimize fairness. To see this, we consider the efficiency problem by replacing the objective function in (3.1) by

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

which is the total benefits earned by all agents. We call our main problem in (3.1) the *fairness problem* and the one with the alternative objective function the *efficiency problem*. We will propose algorithms that can effectively and efficiently find a near-optimal solution for our fairness problem. Beside investigating how our reported solution deviate from a fairest solution (i.e., an optimal solution to (3.1)), we will also examine the efficiency gap between our reported solution and an optimal solution to the efficiency problem.

doi:10.6342/NTU201603583

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 3.1: List of notations

Table 3.1 lists the notations used in this model.

3.2 NP-hardness

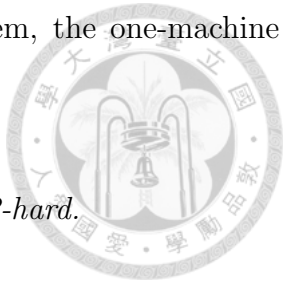
In this section, we show that our problem is NP-hard.

Theorem 1. *The job allocation problem in (3.1) is strongly NP-hard.*

Proof. Consider a special case with m uncapacitated (i.e., $K \geq \sum_{j \in J} c_j$) machines and $n = 3m$ jobs with benefits in an integer set $\{b_1, b_2, ..., b_n\}$. And there is a bound B that satisfies $\sum_{i=1}^{n} a_i = mB$. If the benefits of all jobs satisfy $\frac{B}{4} < b_j < \frac{B}{2}$, it is indeed a 3-Partition problem. As 3-Partition problem is strongly NP-hard, the proof is complete. □

doi:10.6342/NTU201603583

In fact, if there is only one capacitated machine in our problem, the one-machine problem is still (weakly) NP-hard.



Theorem 2. *The job allocation problem in (3.1) with $m = 1$ is NP-hard.*

Proof. Our problem with one capacitated machine is exactly a knapsack problem. □

doi:10.6342/NTU201603583

Chapter 4

Analysis

4.1 Capacitated highest-benefit job first algorithm

By the above, we would like to design an algorithm to obtain an approximation solution. Our algorithm is based on LPT (longest time first) rule with some straightforward modifications. We first sort all jobs in a non-increasing order according to their benefits. Then, we follow the order and in each iteration try assign each 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 to complete that job, we try the machine with the next lowest cumulative benefit. If a job cannot be assigned to any machine, it will be omitted. We repeat this process until the last job has been tried to be assigned. For ease of exposition, we denote this algorithm as the capacitated highest-benefit job first (CHBF) algorithm.

The performance of the CHBF algorithm naturally depends on the relationship be-

doi:10.6342/NTU201603583

tween job benefits and workloads. Its performance can be very bad if benefits and workloads are not regulated by any relationship. Therefore, our analysis will be limited to the case that a higher-workload job will always bring in a higher benefit. Below we will show that CHBF has three different worst-case performance guarantees for three different scenario. In the first, second, and third scenarios, the benefit of a job is a linear, convex, and concave function of its workload. The first scenario is for situations that benefits are proportional to workloads. The second one occurs when jobs exhibit economy of scale, and thus a larger job generates relatively more benefits. This is more likely the case if these jobs are manufacturing or production tasks. Finally, the last scenario models the situation in which the marginal benefit of a unit of workload is decreasing. This is a typical feature if the resulting products/services are to be sold to the end market.

Without loss of generality, we assume that $c_1 > c_2 > \cdots > c_n$ (and thus $b_1 > b_2 > \cdots > b_n$) in all the following analyses.

4.1.1 Linear benefit-workload relationship

We first prove that CHBF is a factor-$\frac{1}{2}$ algorithm, i.e., can generate a solution whose objective value is at least $\frac{1}{2}$ of that from an optimal solution, when job benefit and workload are proportional (i.e., $b_j = hc_j$ for some $h > 0$). We denote z^* as the objective value of an optimal solution and z' as that of the CHBF solution.[1]

Theorem 3. *If $b_j = hc_j$ for all $j \in J$ for some $h > 0$, we have $z' \geq \frac{1}{2}z^*$.*

Proof. Without loss of generality, we assume that $c_j \leq K$ for all $j \in J$. We know that

[1]In the appendix we show that Theorem 3 is still valid even if agents have different capacity levels.

doi:10.6342/NTU201603583

if all jobs can be assigned to machines under CHBF algorithm without being skipped due to limited capacity, capacity constraints will have no impact on the solution, and the schedule will be the same as that obtained by the LPT rule. It has been proved that LPT will have a performance guarantee $\frac{3}{4}$ (Deuermeyer et al., 1982). Therefore, below we will only focus on the case that at least one job cannot be assigned.

We first prove that $z' \geq \frac{1}{2}hK$. Suppose that there is a counterexample in which $z' < \frac{1}{2}hK$, and job l is the first job not assigned during the CHBF process. Let machine i be the one having the lowest cumulative benefit when CHBF tries (but fails) to assign job l. Because $c_l < K$, there must be at least one job assigned to machine i. Because $z' < \frac{1}{2}hK$, the cumulative benefit z_i of machine i at the moment of assigning job l is also lower than $\frac{1}{2}hK$.

For job l, we know $c_l < \frac{1}{2}K$. Otherwise, we will have $b_l = hc_l \geq \frac{1}{2}hK > z_i$, which implies that job l is more profitable than will be larger than any job that has been assigned to machine i. This violates the rule of CHBF. Therefore, we have $c_l < \frac{1}{2}K$. However, if $z_i < \frac{1}{2}hK$, the total workloads of jobs that have been assigned to machine i is no greater than $\frac{1}{2}K$. Now, if $c_l < \frac{1}{2}K$, job l can actually be assigned to machine i. This violates our assumption that job l cannot be assigned. Collectively, we have $z' \geq \frac{1}{2}hK$.

Now, combining the above result and the obvious fact that $z^* \leq hK$, we obtain $z' \geq \frac{1}{2}hK \geq \frac{1}{2}z^*$. □

The next theorem shows that the bound $\frac{1}{2}$ is tight.

Theorem 4. *The performance guarantee $\frac{1}{2}$ of CHBF in Theorem 3 is tight.*

Proof. Let ϵ be a small positive number. Suppose that we have $2m$ jobs with workload

doi:10.6342/NTU201603583

$\frac{1}{2}K$ 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. □

4.1.2 Convex benefit-workload relationship

In this section, we prove that CHBF has a performance guarantee when job benefits are convex in workloads in the form $b_j = hc_j^t$ for some $h > 0$ and $t > 1$. One parameter that is relevant is the workload of the mth-largest-benefit job. Let $c_m = \beta K$ for some $\beta \in (0, 1]$. We rely on three lemmas to build our main result.

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

$$hK^t\beta^{t-1} \geq z^*,$$

where $\beta = \frac{c_m}{K}$.

Proof. To establish an upper bound, note that z^* can be the largest if all machines are fully loaded (i.e., have no residual capacity). When the relationship between job benefits and workloads is convex, the most beneficial way to consume all the capacity of one machine is to use only one job. In each machine contains exactly one job of workload $c_j = K$, we will have hK^t as the objective value. This is clearly an upper bound.

To further improve this bound, note that we may not have m jobs whose workloads are all so high. Suppose that the mth job's workload is only βK for some $\beta \leq \frac{1}{2}$. In this

doi:10.6342/NTU201603583

case, the most beneficial way to utilize the capacity on the machine being assigned job m is to assign $\lfloor \frac{1}{\beta} \rfloor$ jobs to it, each with workload βK. Its cumulative benefit is thus no greater than $\lfloor \frac{1}{\beta} \rfloor h(\beta K)^t > \frac{1}{\beta} h(\beta K)^t = hK^t \beta^{t-1}$. This is an upper bound of z^*. □

Lemma 2. *If $b_j = hc_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have*

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

Proof. We prove this result by considering the moment of our first failure. The failure happens when we cannot assign job j to the least cumulative machine i at that moment. Let p be the number of jobs that have been assigned on machine i at that moment. For ease of exposition, let $\alpha = \min\{\frac{1}{2^t}, \frac{n-m}{(n-m+1)^t}\}$ be the bound to be proved.

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

We now show $ph(\frac{K - \sqrt[t]{\frac{\alpha}{p}} K}{p})^t \geq \alpha hK^t$ to establish a contradiction. Some arithmetic shows that $ph(\frac{K - \sqrt[t]{\frac{\alpha}{p}} K}{p})^t \geq \alpha hK^t$ if and only if $\alpha \leq \frac{p}{(p+1)^t}$. As a function of p, it can be verified that $\frac{p}{(p+1)^t}$ is quasi-concave in p for any $t > 1$. We know that the smallest and largest possible numbers of jobs on machine i is 1 and $n - m$, respectively. This implies

doi:10.6342/NTU201603583

that the global minimum of $\frac{p}{(p+1)^t}$ is either 1 or $n-m$, and thus $\alpha = \min\{\frac{1}{2^t}, \frac{n-m}{(n-m+1)^t}\} \leq \frac{p}{(p+1)^t}$ will always be satisfied. The proof is then complete. □

The lower bound of z' is the minimum of two values. Whether $\frac{1}{2^t}$ or $\frac{n-m}{(n-m+1)^t}$ will be the smaller one depends on t. It can be easily verified that 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 $\frac{n-m}{(n-m+1)^t} > \frac{1}{2^t}$. Note that when t approaches 1, eventually $\frac{1}{2^t}$ will be the smaller one (as long as $n-m > 1$), and this bound converges to $\frac{1}{2}$ as Theorem 3 suggests for the linear benefit-workload relationship (where $t = 1$).

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

$$z' \geq h\beta^t K^t,$$

where $\beta = \frac{c_m}{K}$.

Proof. According to CHBF rule, we assign the first mth jobs one at a time to the m machines. After that, each machines' benefit will be at least $h(\beta K)^t$, which implies that $z' \geq h\beta^t K^t$. □

We now state our main result, which is a direct combination of the above three lemmas.

Theorem 5. *If $b_j = hc_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have*

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

Proof. According to Lemma 2 and 3, we have $z' \geq \max\{\min\{\frac{1}{2^t}, \frac{n-m}{(n-m+1)^t}\}, \beta^t\} hK^t$. Also, we know $hK^t\beta^{t-1} \geq z^*$ by Lemma 1. Thus, the proof can be completed by combining all the three lemmas. □

doi:10.6342/NTU201603583

When β is large, β^t would dominate $\min\{\frac{1}{2^t}, \frac{n-m}{(n-m+1)^t}\}$, and the performance guarantee can be trivially found as β. On the contrary, when β is small, $\min\{\frac{1}{2^t}, \frac{n-m}{(n-m+1)^t}\}$ would be a bound, and Lemma 1 helps us further improve this bound by having the denominator β^{t-1}. Note that it is possible for the worst-case performance guarantee to be above $\frac{1}{2}$. In other words, the convex relationship between benefits and workloads may actually help CHBF in achieving a better worst-case performance guarantee.

4.1.3 Concave benefit-workload

Here we prove the performance guarantee when the benefit is concave in workload in the form $b_j = hc^t$ for some $h > 0, t < 1$. Let $c_m = \beta K$ for some $\beta \in (0, 1]$.

Lemma 4. *If $b_j = hc^t$ for all $j \in J$ for some $h > 0, t < 1$, we have*

$$z' \geq \frac{K^{t-1}}{2^t} z^*,$$

where $\beta = \frac{c_m}{K}$.

Proof. The same as the proof in convex. We would like to establish an upper bound. 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. In each machine contains exactly K jobs of workload 1, we will have hK as the objective value. This is clearly an upper bound.

Then, we consider the moment of the first failure. The failure happens when we cannot assign job j to the least cumulative machine i at that moment. Let p be the number of jobs that have been assigned on machine i at that moment. For ease of exposition, let $\alpha = \frac{K^{t-1}}{2^t}$ be the bound to be proved.

doi:10.6342/NTU201603583

We prove by contradiction by assuming that that $z' < \alpha hK$. This implies that at the failure moment machine i's cumulative benefit is also lower than αhK. Then, the benefit of job j should be less than the benefit αhK of machine i. Otherwise, it should be assigned first. As the result, we know the corresponding workload of the job j is $c_j \leq (\alpha K)^{\frac{1}{t}}$ and the left capacity is $K_i' \geq K - (\alpha K)^{\frac{1}{t}}$. If not, the machine i would have enough capacity for job j.

Based on the characteristic of concave, the possibility of the least cumulative benefit z' is $h(K - (\alpha K)^{\frac{1}{t}})^t$. We show that it is at least equal to αhK. To see this, by our calculations, it indicates $\alpha \leq \frac{K^{t-1}}{2^t}$ which will be satisfied. As the result, it contradicts to the assumption. Combine the proof, $z' \geq \frac{K^{t-1}}{2^t} hK \geq \frac{K^{t-1}}{2^t} z^*$. □

4.2 Modified capacitated highest-benefit job first algorithm

Because CHBF is just a greedy algorithm, it may perform badly under some situations. As the result, we embed linear programming into the CHBF algorithm and develop another algorithm. We call this algorithm as the modified capacitated highest-benefit job first (MCHBF) algorithm.

The idea is simple. We first relax the integer constraints to obtain a relaxation of the

doi:10.6342/NTU201603583

given problem. In this case, our relaxed problem can be reduced to

$$\begin{aligned} \max \quad & w \\ \text{s.t.} \quad & w \le \sum_{j \in J} b_j x_{ij} \quad \forall i \in I \\ & \sum_{j \in J} c_j x_{ij} \le K \quad \forall i \in I \\ & \sum_{i \in I} x_{ij} \le 1 \quad \forall j \in J \\ & x_{ij} \ge 0 \quad \forall i \in I, j \in J, \end{aligned} \tag{4.1}$$

which is a linear program. We then use a linear programming solver to obtain an optimal solution to the relaxed program. If the result shows that job j is assigned to machine i ($x_{ij} = 1$) completely, we assign job j to machine i. We then schedule the left jobs by the CHBF algorithm.

Though we do not have any theoretical properties for MCHBF, it will be shown in the next chapter that MCHBF may outperform well than CHBF in some scenarios. Possible reasons will then be discussed.

doi:10.6342/NTU201603583

doi:10.6342/NTU201603583

Chapter 5

Numerical study

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 which the job benefit is linear in, non-decreasing and convex in, non-decreasing and concave in, and unrelated with the job workload. The second factor is agent capacity. We consider three scenarios, one with loose capacity, one with tight capacity, and one with no limited capacity. These two factors together generate twelve scenarios for numerical experiments. 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 unrelated): $b_j \sim U(0, 50)$.

- Capacity tightness: scenario N (for unlimitation): $K = \infty$; scenario L (for loose): $K = \left(\frac{\sum_{j \in J} c_j}{m}\right)$; scenario T (for tight): $K = \frac{3}{4}\left(\frac{\sum_{j \in J} c_j}{m}\right)$.

doi:10.6342/NTU201603583

For each of the twelve combinations of scenarios, we generate 100 experiments with several combination of m and n ($m = 5$ with $n = 20,\ 50,\ 500$, $m = 15$ with $n = 50,\ 500$, and $m = 50$ with $n = 500$). For each experiments, we use branch-and-bound algorithm to find a solution [1] , CHBF to generate a CHBF-based solution, and use the LP solution and CHBF generate MCHBF-based solution. We denote solution of "fairness" version as w^{IP}, w^{LP}, w^{CHBF} w^{MCHBF} and solution of "efficiency" version as z^{IP}, z^{LP}, z^{CHBF} z^{MCHBF}. They are then compared to generate Table A.1-A.4 in Appendix.

To briefly understand how each factor affects the performance of CHBF and MCHBF, we combine the scenarios to generate Table 5.1, 5.2, and 5.3. In Table 5.1, we can observe that CHBF and MCHBF perform better if the capacity is looser. This is an intuitive result, the jobs can be assigned appropriate by CHBF due to flexibility of agent capacity.

capacity	$\frac{z(x^{CHBF})}{z^{IP}}$	$\frac{z(x^{MCHBF})}{z^{IP}}$	$\frac{w^{CHBF}}{w^{IP}}$	$\frac{w^{MCHBF}}{w^{IP}}$
N	1	1	0.988	0.971
L	0.987	0.978	0.964	0.937
T	0.943	0.973	0.912	0.931

Table 5.1: Numerical results of capacity tightness

Table 5.2 shows that the performance ratios of CHBF and MCHBF are higher when $\frac{n}{m}$ becomes bigger. It's also an intuitive result. With more options of selecting jobs, our algorithm can assign more valuable jobs to machines. More interesting is that no matter for "fairness" or "efficiency" version, CHBF performs better than MCHBF when $\frac{n}{m}$ is less while it is better to applied MCHBF when $\frac{n}{m}$ is large.

[1]For $m = 5$, $n = 20$, we can find an IP solution. Due to memory limitation, we find a LP solution instead in other case.

doi:10.6342/NTU201603583

$\frac{n}{m}$	$\frac{z(x^{CHBF})}{z^{IP}}$	$\frac{z(x^{MCHBF})}{z^{IP}}$	$\frac{w^{CHBF}}{w^{IP}}$	$\frac{w^{MCHBF}}{w^{IP}}$
100	0.983	0.999	0.980	0.997
33.333	0.983	0.998	0.977	0.989
10	0.981	0.991	0.965	0.962
4	0.971	0.964	0.942	0.907
3.333	0.961	0.960	0.899	0.863

Table 5.2: Numerical results of number of agents and jobs

Finally, by observing Table 5.3, we find that for "efficiency" version, CHBF and MCHBF perform the best when benefits are unrelated or linear in workloads. The performance becomes worse when the benefits become convex in workloads and even worse when benefits are concave in workloads. And they are all larger than 0.98 except for using CHBF when benefits are concave in workloads. This result shows us that it would not sacrifice too much efficiency using CHBF or MCHBF. For "fairness" version, CHBF and MCHBF all perform the best when benefits are linear in workloads. Moreover, CHBF performs well when benefits are convex in workloads while performing the worst when benefits are concave in workloads. MCHBF performs more or less the same except when benefits are linear in workloads. The reason behind this observation is as follows. Our algorithm assigns jobs based on jobs' benefit. If benefits are linear in workloads, the value of each job tends to be the same. It means that it is easier for CHBF to assign jobs to achieve "fairness" or "efficiency". If benefits are convex in workload, high-benefit jobs and low-benefit jobs tend to have relatively similar workloads, and thus machine capacity does not introduce a huge difficulty to the performance of CHBF. However, MCHBF does not perform that well because of using LP solution. As for the case when benefits are

doi:10.6342/NTU201603583

concave in workloads, high-benefit jobs all require workloads that are much larger than those of low-benefit jobs. It is then less likely that these jobs should be assigned first, as CHBF does. Therefore, CHBF does not perform well. This is also understandable as some patterns almost always help a deterministic algorithm improve this performance.

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

benefit-workload	$\frac{z(x^{CHBF})}{z^{IP}}$	$\frac{z(x^{MCHBF})}{z^{IP}}$	$\frac{w^{CHBF}}{w^{IP}}$	$\frac{w^{MCHBF}}{w^{IP}}$
R	0.992	0.988	0.947	0.936
L	0.990	0.988	0.979	0.971
X	0.988	0.980	0.976	0.934
A	0.936	0.979	0.918	0.945

Table 5.3: Numerical results of the benefit-workload relationship

In summary, although the minimum performance of CHBF and MCHBF are 0.583 and 0.55 through all instances, the average performance of all instances are larger than 0.8. In particular, CHBF performs well when job benefits are convex or linear to workloads, and MCHBF performs better than CHBF when job benefits are concave to workloads. These results will suggest decision makers about which algorithm to apply to their own problems.

doi:10.6342/NTU201603583

Chapter 6

Conclusion and Future Work

In this study, we consider a job allocation problem with fairness as the main concern. We formulate the problem as an integer program to maximize the lowest benefit of an machine subject to capacity constraints. By modifying a classic known to be effective for the uncapacitated version of our problem, we develop our own algorithm for our job allocation problem. We then prove the performance guarantee is $\frac{1}{2}$ when the job benefits are linear in workloads. And this worst-case performance will be tight when the capacity of each agent is the same. Also, we prove the performance guarantee is β^{t-1} $(t > 1)$ or $\frac{K^{t-1}}{2^t}$ $(t < 1)$ when the job benefits are convex or concave in workloads. Besides, we establish another algorithm by applying LP solution into our original algorithm. We then examine the performance and find that our two algorithms all work better when capacity is loose than tight, or benefits are linear or convex than concave. In particular, we show that the algorithm is more appropriate when the production environment exhibit significant economy of scale.

Some further investigations may further improve in this study. One promising direc-

doi:10.6342/NTU201603583

tion is to look for more worse-case performance guarantees of our problem, either for the general problem or under some conditions. If we can prove more performance guarantee, it will add significant value to the literature. Another direction is that we may try to modify our algorithm by the ideas coming up with when we prove the bounds. It may be helpful under some conditions.

doi:10.6342/NTU201603583

Appendix A

Supplemental Proofs and Results of the Numerical Studies

Theorem 6. *Suppose that* $K_i \neq K_j$ $\{i \neq j | i, j \in I\}$ *and* $b_j = hc_j$ *for some* $h > 0$*, then the CHBF solution is at least one half as good as an optimal solution. More precisely, let* z^* *be the objective value of an optimal solution and* z' *be that of the CHBF solution, we have* $z' \geq \frac{1}{2}z^*$.

Proof. Without loss of generality, we set all $c_j \leq K_i$ $\{K_i \leq K_j, i \neq j | i, j \in I\}$ and $b_j = c_j$. We know that if all the jobs can be assigned to machines under CHBF algorithm, capacity constraints will have no impact on the solution, and the schedule will be the same as LPT rule does. It has been proved that LPT will have performance guarantee $\frac{3}{4}$ by Deuermeyer et al. (1982), therefore we only focus on the case that there are still some jobs not assigned.

We first prove that $z' \geq \frac{1}{2}K_i$. Assume that there is a counterexample $z' < \frac{1}{2}K_i$, and job l has not been assigned. We assume the machine j has the lowest cumulative benefit

doi:10.6342/NTU201603583

z'. There are two cases: (1) $c_l \leq \frac{1}{2}K_i$, (2) $c_l > \frac{1}{2}K_i$.

In case (1), if $c_l \leq \frac{1}{2}K_i$, the machine j must have enough capacity for the assignment of job l. It means that job l should be assigned to machine j to improve the solution according to the rule of CHBF algorithm, which contradicts to the assumption that job l has not been assigned.

In case (2), notice that $c_l > \frac{1}{2}K_i > z'$, which indicates that all jobs assigned to machine j are less profitable than job l. Followed by the CHBF rule, we assign more profitable jobs first. It means that when we assign job l, no job has been assigned to machine j. Then, job l can be assigned to machine j, which has no benefit at that time. It contradicts to the assumption that job l has not been assigned.

Using the fact that $z^* \leq K_i$ and by above, we obtain $z' \geq \frac{1}{2}K_i \geq \frac{1}{2}z^*$. □

doi:10.6342/NTU201603583

m	n	Capacity	B-W	$\frac{z^{IP}}{z^{LP}}$	$\frac{z(x^{CHBF})}{z^{IP}}$	$\frac{z(x^{MCHBF})}{z^{IP}}$	$\frac{w^{IP}}{w^{LP}}$	$\frac{w^{CHBF}}{w^{IP}}$	$\frac{w^{MCHBF}}{w^{IP}}$
5	20	L	L	0.994	0.960	0.938	0.994	0.940	0.890
5	20	L	A	0.982	0.958	0.939	0.982	0.962	0.939
5	20	L	X	0.999	0.982	0.920	0.999	0.975	0.852
5	20	L	R	0.995	0.987	0.977	0.995	0.910	0.871
5	20	T	L	0.997	0.963	0.966	0.997	0.945	0.945
5	20	T	A	0.981	0.845	0.922	0.981	0.822	0.875
5	20	T	X	0.935	0.988	0.963	0.935	0.991	0.943
5	20	T	R	0.989	0.967	0.940	0.989	0.874	0.840
5	20	N	L	1	1	1	1	0.971	0.943
5	20	N	A	1	1	1	1	0.959	0.936
5	20	N	X	1	1	1	1	0.987	0.905
5	20	N	R	1	1	1	1	0.972	0.941
5	50	L	L	–	0.993	0.988	–	0.989	0.980
5	50	L	A	–	0.985	0.971	–	0.970	0.955
5	50	L	X	–	0.999	0.978	–	0.996	0.927
5	50	L	R	–	0.998	0.995	–	0.970	0.955
5	50	T	L	–	0.992	0.996	–	0.987	0.992
5	50	T	A	–	0.829	0.959	–	0.815	0.924
5	50	T	X	–	0.978	0.969	–	0.970	0.915
5	50	T	R	–	0.989	0.980	–	0.925	0.927
5	50	N	L	–	1	1	–	0.995	0.992
5	50	N	A	–	1	1	–	0.989	0.984
5	50	N	X	–	1	1	–	0.999	0.991
5	50	N	R	–	1	1	–	0.995	0.992
5	500	L	L	–	1	0.999	0.994	1	0.998
5	500	L	A	–	1	0.997	0.982	0.999	0.995
5	500	L	X	–	1	0.998	0.999	1	0.994
5	500	L	R	–	1	1	0.995	0.996	0.996
5	500	T	L	–	1	1	0.997	1	1
5	500	T	A	–	0.805	0.996	0.981	0.804	0.993
5	500	T	X	–	0.998	0.998	0.935	0.998	0.994
5	500	T	R	–	0.988	0.999	0.989	0.966	0.991
5	500	N	L	–	1	1	1	1	0.999
5	500	N	A	–	1	1	1	1	1
5	500	N	X	–	1	1	1	1	1
5	500	N	R	–	1	1	1	1	1

Table A.1: The average performance of CHBF and MCHBF - $m = 5$

doi:10.6342/NTU201603583

m	n	Capacity	B-W	$\frac{z^{IP}}{z^{LP}}$	$\frac{z(x^{CHBF})}{z^{IP}}$	$\frac{z(x^{MCHBF})}{z^{IP}}$	$\frac{w^{IP}}{w^{LP}}$	$\frac{w^{CHBF}}{w^{IP}}$	$\frac{w^{MCHBF}}{w^{IP}}$
15	50	L	L	–	0.946	0.933	–	0.903	0.882
15	50	L	A	–	0.932	0.942	–	0.863	0.860
15	50	L	X	–	0.971	0.946	–	0.904	0.807
15	50	L	R	–	0.987	0.976	–	0.859	0.826
15	50	T	L	–	0.969	0.972	–	0.943	0.948
15	50	T	A	–	0.876	0.921	–	0.823	0.815
15	50	T	X	–	0.890	0.888	–	0.788	0.771
15	50	T	R	–	0.962	0.940	–	0.831	0.767
15	50	N	L	–	1	1	–	0.965	0.931
15	50	N	A	–	1	1	–	0.956	0.929
15	50	N	X	–	1	1	–	0.983	0.877
15	50	N	R	–	1	1	–	0.966	0.938
15	500	L	L	–	1	0.999	–	0.999	0.999
15	500	L	A	–	0.998	0.996	–	0.996	0.989
15	500	L	X	–	1	0.998	–	1	0.985
15	500	L	R	–	1	1	–	0.987	0.984
15	500	T	L	–	1	1	–	0.999	1
15	500	T	A	–	0.810	0.994	–	0.808	0.979
15	500	T	X	–	0.995	0.995	–	0.994	0.978
15	500	T	R	–	0.992	0.997	–	0.943	0.973
15	500	N	L	–	1	1	–	1	0.999
15	500	N	A	–	1	1	–	0.999	0.998
15	500	N	X	–	1	1	–	1	0.988
15	500	N	R	–	1	1	–	1	0.999
50	500	L	L	–	0.996	0.994	–	0.993	0.990
50	500	L	A	–	0.990	0.992	–	0.972	0.927
50	500	L	X	–	1	0.995	–	0.998	0.951
50	500	L	R	–	1	0.998	–	0.956	0.944
50	500	T	L	–	0.996	0.999	–	0.992	0.998
50	500	T	A	–	0.821	0.989	–	0.800	0.922
50	500	T	X	–	0.986	0.989	–	0.978	0.943
50	500	T	R	–	0.993	0.988	–	0.900	0.915
50	500	N	L	–	1	1	–	0.995	0.994
50	500	N	A	–	1	1	–	0.980	0.986
50	500	N	X	–	1	1	–	0.999	0.996
50	500	N	R	–	1	1	–	0.996	0.994

Table A.2: The average performance of CHBF and MCHBF - $m = 15, 50$

doi:10.6342/NTU201603583

m	n	Capacity	B-W	$\frac{z^{IP}}{z^{LP}}$	$\frac{z(x^{CHBF})}{z^{IP}}$	$\frac{z(x^{MCHBF})}{z^{IP}}$	$\frac{w^{IP}}{w^{LP}}$	$\frac{w^{CHBF}}{w^{IP}}$	$\frac{w^{MCHBF}}{w^{IP}}$
5	20	L	L	0.980	0.898	0.809	0.980	0.865	0.676
5	20	L	A	0.957	0.885	0.840	0.957	0.892	0.839
5	20	L	X	0.989	0.904	0.701	0.989	0.887	0.571
5	20	L	R	0.980	0.950	0.905	0.980	0.742	0.550
5	20	T	L	0.981	0.877	0.891	0.981	0.848	0.863
5	20	T	A	0.942	0.747	0.832	0.942	0.583	0.666
5	20	T	X	0.833	0.938	0.813	0.833	0.887	0.671
5	20	T	R	0.968	0.886	0.828	0.968	0.665	0.553
5	20	N	L	1	1	1	1	0.912	0.838
5	20	N	A	1	1	1	1	0.896	0.868
5	20	N	X	1	1	1	1	0.943	0.686
5	20	N	R	1	1	1	1	0.919	0.803
5	50	L	L	–	0.979	0.971	–	0.971	0.945
5	50	L	A	–	0.970	0.938	–	0.945	0.906
5	50	L	X	–	0.994	0.836	–	0.988	0.757
5	50	L	R	–	0.991	0.967	–	0.872	0.834
5	50	T	L	–	0.967	0.985	–	0.962	0.969
5	50	T	A	–	0.794	0.910	–	0.786	0.869
5	50	T	X	–	0.953	0.922	–	0.947	0.843
5	50	T	R	–	0.955	0.952	–	0.769	0.809
5	50	N	L	–	1	1	–	0.979	0.977
5	50	N	A	–	1	1	–	0.970	0.966
5	50	N	X	–	1	1	–	0.995	0.878
5	50	N	R	–	1	1	–	0.985	0.977
5	500	L	L	–	1	0.996	1	1	0.980
5	500	L	A	–	0.999	0.991	0.998	1	0.988
5	500	L	X	–	1	0.988	1	1	0.971
5	500	L	R	–	1	0.995	0.968	0.996	0.980
5	500	T	L	–	1	1	1	1	1
5	500	T	A	–	0.792	0.991	0.791	0.804	0.988
5	500	T	X	–	0.997	0.993	0.997	0.998	0.984
5	500	T	R	–	0.979	0.994	0.930	0.966	0.978
5	500	N	L	–	≈ 1	1	1	1	0.992
5	500	N	A	–	≈ 1	1	0.999	1	0.999
5	500	N	X	–	≈ 1	1	1	1	0.997
5	500	N	R	–	≈ 1	1	1	1	0.991

Table A.3: The minimum performance CHBF and MCHBF- $m = 5$

doi:10.6342/NTU201603583

m	n	Capacity	B-W	$\frac{z^{IP}}{z^{LP}}$	$\frac{z(x^{CHBF})}{z^{IP}}$	$\frac{z(x^{MCHBF})}{z^{IP}}$	$\frac{w^{IP}}{w^{LP}}$	$\frac{w^{CHBF}}{w^{IP}}$	$\frac{w^{MCHBF}}{w^{IP}}$
15	50	L	L	–	0.902	0.844	–	0.824	0.766
15	50	L	A	–	0.854	0.863	–	0.721	0.780
15	50	L	X	–	0.926	0.842	–	0.738	0.699
15	50	L	R	–	0.969	0.879	–	0.756	0.588
15	50	T	L	–	0.904	0.904	–	0.861	0.871
15	50	T	A	–	0.837	0.831	–	0.631	0.707
15	50	T	X	–	0.803	0.819	–	0.591	0.591
15	50	T	R	–	0.924	0.888	–	0.592	0.553
15	50	N	L	–	1	1	–	0.920	0.850
15	50	N	A	–	1	1	–	0.929	0.858
15	50	N	x	–	1	1	–	0.947	0.692
15	50	N	R	–	1	1	–	0.915	0.874
15	500	L	L	–	0.999	0.998	–	0.998	0.998
15	500	L	A	–	0.996	0.992	–	0.991	0.973
15	500	L	X	–	1	0.991	–	1	0.944
15	500	L	R	–	1	0.998	–	0.937	0.950
15	500	T	L	–	0.999	1	–	0.998	0.999
15	500	T	A	–	0.789	0.986	–	0.788	0.958
15	500	T	X	–	0.992	0.985	–	0.992	0.957
15	500	T	R	–	0.984	0.994	–	0.890	0.950
15	500	N	L	–	1	1	–	0.999	0.999
15	500	N	A	–	1	1	–	0.998	0.996
15	500	N	X	–	1	1	–	1	0.957
15	500	N	R	–	1	1	–	0.999	0.999
50	500	L	L	–	0.996	0.992	–	0.987	0.982
50	500	L	A	–	0.990	0.982	–	0.964	0.870
50	500	L	X	–	1	0.999	–	0.997	0.927
50	500	L	R	–	1	0.999	–	0.847	0.895
50	500	T	L	–	0.996	0.990	–	0.983	0.994
50	500	T	A	–	0.821	0.809	–	0.789	0.851
50	500	T	X	–	0.986	0.972	–	0.961	0.926
50	500	T	R	–	0.993	0.988	–	0.779	0.859
50	500	N	L	–	1	1	–	0.989	0.990
50	500	N	A	–	1	1	–	0.971	0.980
50	500	N	X	–	1	1	–	0.998	0.993
50	500	N	R	–	1	1	–	0.991	0.989

Table A.4: The minimum performance of CHBF and MCHBF - $m = 15, 50$

doi:10.6342/NTU201603583

Bibliography

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

Bansal, N., M. Sviridenk. 2006. The santa claus problem. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*. Seattle, WA, USA, 31–40.

Bertsimas, D., V.F. Farias, N. Trichakis. 2011. The price of fairness. *Operation Research* **59**(1) 17–31.

Blazwicz, J., W. Domschke, E. Pesch. 1996. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research* **93**(1) 1–33.

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

doi:10.6342/NTU201603583

minimum processor finish time in a multiprocessor system. *Society for Industrial and Applied Mathematics* **3**(2) 190–196.

Fiat, A., G.J. Woeginger. 1998. *Online Algorithms: The State of the Art*. Springer, Berlin, Germany.

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.

Gupta, S.K., J. Kyparisis. 1987. Single machine scheduling research. *Omega* **15**(3) 207–227.

Massab, 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.

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

Ruiz, R., J.A. Vzquez-Rodrguez. 2010. The hybrid flow shop scheduling problem. *European Journal of Operational Research* **105**(1) 1–18.

Williamson, D.P., D.B. Shmoys. 2011. *The design of approximation algorithms*. Cambridge University Press, London, UK.

doi:10.6342/NTU201603583