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

iSMART 近似演算法：考慮公平性與機台歧異性之工作分配問題

iSMART: An Approximation Algorithm for Fair Job Allocation on Unrelated Machines

林怡安

Yi-An Lin

指導教授：孔令傑 博士

Adviser: Ling-Chieh Kung, Ph.D.

中華民國 106 年 8 月

August 2017

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

國立臺灣大學(碩、博)士學位論文
口試委員會審定書

An Approximation Algorithm for
Fair Job Allocation on Unrelated Machines

本論文係林怡安君（學號 r04725037）在國立臺灣大學資訊管理學系、所完成之碩士學位論文，於民國 106 年 6 月 22 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：[illegible]

[illegible] [illegible]

[illegible]

所　　長：[illegible]

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

誌謝

研究所的兩年過得很快，充滿壓力挑戰卻也令人成長，要寫一篇論文，對我來說就好比登天一樣困難，能夠完成真是多虧有大家的幫忙。

首先最要感謝我的指導教授，孔令傑，謝謝您不厭其煩的每周與我討論，指引我研究的方向，修正我的錯誤，一步一步帶領著我走到今天。謝謝您在教學、研究、導師三個領域都投入了百分之兩百的心力與時間，不斷朝著自己的理想勇敢邁進，您是我非常敬佩的偶像，但願您永遠身體健康、平安快樂。同時也要感謝口試委員們，李家岩老師、林春成老師、廖崇碩老師，在口試當天給予我寶貴的建議，使我的論文能夠更充實完善。以及黃奎隆老師於課堂中的教導和照顧。

接著感謝 IEDO LAB 的同居戰友、學長姐、學弟妹，有你們的陪伴才豐富了我的研究所時光，謝謝 FiFi 的大方餵食、Jeff 的冷笑話大全、千瑜的女神風範、阿波的一起崩潰、韋志的嘲諷搞笑、還有宸安的神速領悟，謝謝 Kiwi 的研究傳承、Hoho 的天使關懷、冠宇的溫暖支持、偉宏的阿諛諂媚，謝謝 Peter 的爽朗笑聲、佩蓉的快樂氣質、敬傑的超越凡人、鑑霖的張氏幽默，等等太多謝謝！

感謝台大資管、台大工工、清大工工、交大工工等所有雪兒的好朋友們，你們是我快樂的泉源，我想你們會是我最捨不得畢業的因素，以後一定要繼續保持聯絡，有機會的話要來找我玩。當然還要特別感謝老大這 6 年來的疼愛與鼓勵，你的積極上進是促使我想要更進一步的動力，很幸運這些日子能和你一起成長，讓我在跌跌撞撞中仍能再度站起，感謝有你！

最後感謝我的家人，是你們養育了我，不僅衣食無憂，更讓我在充滿愛的環境中長大，家永遠是我的避風港，而我會努力成為你們永遠的驕傲，我愛你們！

林怡安 謹致

于國立臺灣大學資訊管理研究所

民國一百零六年八月

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

摘要

排程最佳化與工作分配是經典的研究問題，如何找到有效的方法進行作業排程與機台分配來提高產能利用率進而獲利，科學家們仍不斷在尋找答案。然而，除了經濟考量，近年來公平性的議題也備受重視。每個員工、機台、工廠等(以下我們統稱為機台)都希望能在有限的資源下賺得最大利益，如何將工作公平地分配給各機台是管理者面臨的難題。為了解決這個問題，我們設計了一個近似演算法：iSMART，希望能在機台的工作品質或產能上限具有歧異性的情況下，最大化各機台中的最小收益產能比。此演算法根據一個特殊的公平性指標在迭代，每次都指派最有利的工作給此刻獲得最低分的機台。

經過推導，我們證明出當工作利益與工作負荷為線性關係時，iSMART 是一個 $\frac{1}{2}$ 因子近似演算法，在其表現最差的情況下也有最佳解的一半好。當工作利益與工作負荷為凹函數或凸函數關係時，也存在有最差極值。而根據數值分析，我們發現工作利益與負荷間的關係、機台品質、資源限制與工作機台比都會對於 iSMART 演算法的表現有所影響，這些管理意涵讓管理者可依照其產業特性來決定是否適合採用 iSMART 演算法。最終實驗顯示，iSMART 在追求公平性時並不會犧牲掉太大的總體利益，是一個穩定且有效率的演算法。

關鍵字：排程、工作分配、近似演算法、公平性、機台歧異性

doi:10.6342/NTU201704133

Abstract

Scheduling and job allocation have been widely studied in the past few decades. Designing effective methods to determine job schedules as well as machine assignments helps companies increase productivity and earn more benefits. However, not only economic objectives but also fairness become important issues in recent years. Each agent/machine/factory (hereafter “machine”) is willing to earn the most benefit while being restricted by its limited capacity. Managers face the problem of how to assign jobs to heterogeneous machines in a fair way while job benefits might be different due to diverse machine quality. To address this question, we develop an approximation algorithm: iSMART. Based on a specific fairness indicator, the benefit-capacity ratio, iSMART maximizes the minimum fairness score among all machines by assigning the most beneficial job to the machine with lowest score in each iteration.

By analyzing the algorithm, we prove that iSMART is a factor-½ approximation algorithm when benefits and workloads are in a linear relationship. There are also bounds when benefits and workloads are in a convex or concave relationship. Numerical studies show that the performance of iSMART is influenced by benefit-workload relationship, machine quality, capacity tightness, and the ratio of the number of jobs to the number of machines. This provides managerial implications for decision makers to determine when to adopt the algorithm. We also show that iSMART is a reliable algorithm which does not sacrifice too much efficiency while pursuing fairness.

Keywords: Scheduling, Job Allocation, Approximation Algorithm, Fairness, Machine Heterogeneity.

doi:10.6342/NTU201704133

Contents

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

List of Figures

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

List of Tables

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

Chapter 1

Introduction

1.1 Background and motivation

Scheduling and job allocation has been widely discussed over decades and applied in many fields. In the manufacturing industry, a factory manager needs to sequence jobs to be processed on a machine to earn more profit, i.e., a machine-oriented objective that maximizing machine benefit. In more complicated environments, there are more than one machine to be scheduled, e.g., maximizing the minimum machine benefit. Schedules might be created differently because of different job properties. For example, in an off-line problem, information of jobs is known before planning. In other fields, like computer science, the schedule of jobs assigned to multiple processors is a major concern. People try to minimize average loading while designing the operating system. Beside manufacturing and computer science, there are also many scheduling problems in one's daily life. All of these make scheduling and job allocation an important subject.

doi:10.6342/NTU201704133

Many approximation algorithms are developed since most scheduling and job allocation problems are NP-hard. The well-known longest processing time first (LPT) algorithm is designed by Graham (1966, 1969). Later on, Walter (2013) proves that LPT will always outperform restricted LPT (RLPT). A polynomial time approximation scheme (PTAS) is used by Alon et al. (1998). There are many more approximation algorithms for scheduling and job allocation studied in the literature.

The issue of fairness is also critical for allocation problems nowadays. Pointed out by Liu (2016), 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 decide which existing jobs should be assigned to which factories. Since factory managers are evaluated according to the amount of revenues generated by completing jobs (as well as other performance indicators), they take it really seriously. Some social enterprises face fair allocation problems as well. With the idea of "giving them jobs, not money," a social enterprise (or sometimes a government) may hire jobless or even homeless people as agents so that they may earn their livings. For example, "The Sock Mob Homeless Volunteer Network" in London and "The Big Issue Taiwan" provide jobs to the needs so that they can earn benefits by completing jobs. [1] For these social enterprises, the objective is neither limited to job completion nor revenue maximization, but to bring benefits to those in need and to distribute the benefits generated by job completion as equally as possible. In other words, the focus is on *fairness* not on *efficiency.*

Liu (2016) designs an approximation algorithm for job allocation problem with a

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

doi:10.6342/NTU201704133

consideration of fairness. The algorithm assigns a job to the lowest cumulative machine benefit at each iteration if the machine capacity is still enough to take the job workload. However, in his problem, job benefits are the same no matter assigned to which machine. Also, machines are identical, each machine has the same capacity. Thus, we extend from Liu (2016), where jobs machines may earn different benefits completing the same job because of different working quality. By the same time, capacities of machines are not the same, some are available all day long while some might require break times.

The emerging needs of fair allocation motivates us to study a job allocation problem with fairness as the main consideration. The model of our problem is made as general as we can. We design an effective and efficient algorithm that can fairly allocate jobs without violating the capacity constraint of each factory/machine/agent. We prove a bound of our algorithm to ensure its worst-case performance guarantee. Numerical experiments are also conducted to demonstrate its average performance. Based on the results, we finally draw managerial implications for practice.

1.2 Research objectives

In this study, we consider the aforementioned job allocation problem with fairness. For ease of exposition, we will call all factories/machines/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 fairness among machines. In the environment, each machine has its limited capacity and can afford only a certain amount of workloads. However, job benefits may be different while assigning to different machines

doi:10.6342/NTU201704133

since each machine has its own output quality. In order to earn more profits, a machine is willing to accept more jobs as long as there is enough capacity. The objective function in our problem is to maximize the minimum benefit per unit capacity among all machines.

To approach this problem, we first prove its NP-hardness. Knowing that the problem cannot be solved in a polynomial time, we then focus on obtaining an approximation algorithm. We design our algorithm by learning through the literature. We prove the existence of the worst-case guarantee and conduct numerical experiments for our algorithm under different scenarios.

1.3 Research plan

In the next chapter, we will review some relevant literature about scheduling and job allocation, approximation algorithms, and fairness. In Chapter 3, we will describe our optimization problem and show its NP-hardness. Analysis of our algorithm is provided in Chapter 4, showing that there exists performance guarantee using our algorithm in this problem. We then conduct numerical studies in Chapter 5 to test our algorithm's performance in practice and compared it to genetic algorithm. 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/NTU201704133

Chapter 2

Literature Review

2.1 Scheduling and job allocation

Scheduling and job allocation problems have been widely studied in the literature. There are many principles used to classify these problems. Such classifications are made based on objective functions, relationships between jobs and machines, and properties of jobs.

Pinedo (2012) classifies job scheduling problems into two classes according to different objective functions, those with a machine-oriented one and those with a job-oriented one. In the former, maximizing the minimum or minimizing the maximum completion time among all machines are commonly studied. Problems with weighted machines are also well discussed. In the latter, four typical criteria based on jobs are typically investigated. Researchers often minimize one of the lateness, tardiness, completion time, and flow time of jobs as the objective function.

There may be different relationships between jobs and machines. For each job with

doi:10.6342/NTU201704133

only one stage, the problems vary with the number of machines. Gupta and Kyparisis (1987) review scheduling problems that have only one single machine involved. If there are more than one machine processing jobs in the system, it is categorized as a parallel machine problem. When the machines are not exactly the same, it is called an unrelated parallel machine problem. However, there are problems with jobs that have multiple stages to be processed on predetermined machines. For these flow shop problems, Ruiz and Vzquez-Rodrguez (2010) discuss different methods and solution approaches. If different jobs are processed with different processing orders, the problem is called a job shop problem. Blazwicz et al. (1996) review a variety of studies discussing solution techniques for job shop problems.

Based on the properties of jobs, scheduling problems can also be categorized as off-line or on-line problems. In an off-line problem, information of jobs to be processed is all given before planning. In contrast, in an on-line problem, jobs arrive at random times, and relevant information of a job will be known only after the job arrives. A survey of on-line problems was done by Fiat and Woeginger (1998).

According to the classification principles, our job allocation problem is categorized as having a machine-oriented objective function, jobs with only one stage processed on unrelated parallel machines, and an off-line problem. A unique feature of our problem is that each job has two attributes—workload and benefit—along with different machine capacities. We try to allocate jobs to machines while not violating machine capacity constraints so that the lowest machine score is maximized. However, when machine capacities are unlimited, workload is then not a restriction to the assignment and machine scores would be different in the final solution. Thus, we focus on problems that their

doi:10.6342/NTU201704133

machine capacities that are limited.

2.2 Approximation algorithms

Most of the scheduling problems are NP-hard. By the definition of NP-hardness, unless P = NP, we cannot solve the problem in polynomial time. As a result, some researchers propose exact algorithms by improving the generic branch-and-bound algorithms (Haouari and Jemmali, 2008; Walter et al., 2016). More studies are devoted to approximation algorithms or approximation schemes (Williamson and Shmoys, 2011).

Graham (1966, 1969) reports on a well-known minimum makespan problem for multiple identical machines. He designs the longest processing time first (LPT) algorithm, a listing algorithm that sorts all jobs by their processing times in the descending order, and then assigns jobs by this order once at a time to the currently least loaded machine (i.e., the machine having the earliest completion time at the moment). He proves that the performance guarantee of the algorithm is $\frac{4}{3}$.

Another similar approximation algorithm is called restricted LPT (RLPT). Walter (2013) proves that while solving the basic problem of non-preemptively scheduling independent jobs on identical parallel machines so that the minimum (or earliest) machine completion time is maximized, the minimum completion time of the LPT-schedule is at least as long as the minimum completion time of the RLPT-schedule. In other words, LPT will always outperform RLPT. He also shows that RLPT has an approximation factor $\frac{1}{m}$.

For the "dual" version of the minimum makespan problem, the objective function

doi:10.6342/NTU201704133

transforms to maximizing the minimum completion time among all machines. Deuermeyer et al. (1982) find that the LPT algorithm can also be adopted. The bound is shown to be $\frac{3}{4}$. Furthermore, the bound is improved to $\frac{3m-1}{4m-2}$ by Csirik et al. (1992), where m is the number of machines. It converges to $\frac{3}{4}$ when m approaches infinity.

Alon et al. (1998) observe that a polynomial time approximation scheme (PTAS) can be developed under some general assumptions. Classifying jobs by their processing times, the original instance is reduced to a new instance which contains two groups of jobs (big jobs and small jobs). The solution for the new instance is able to be converted to one for the original instance. It is shown that the algorithm can have a $1 + \epsilon$ bound given any arbitrary value of $\epsilon > 0$.

Some attention is directed toward *a posteriori* bounds for the makespan minimization problem. In particular, Blocher and Sevastyanov (2015) improve the *a posteriori* Coffman-Sethi 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) look for a tight bound for a two-machine problem, where the bound is affected by the index of the last job assigned to the critical machine. Following their ideas, we try to prove our bounds by considering the first job that cannot be directly assigned due to the capacity constraints. However, the bounds we are trying to find are *a priori*, not a posteriori.

Liu (2016) designs the capacitated highest-benefit job first (CHBF) algorithm for a minimum machine benefit maximization problem. The algorithm sorts all jobs by their benefits in a descending order and assigns a job to the minimum benefit machine in each iteration. The bound is $\frac{1}{2}$ when the job benefits are in a linear relationship to

doi:10.6342/NTU201704133

workloads. Inspired by Liu (2016), we propose an approximation algorithm for the fair allocation problem with the exhibition of machine capacity and features of jobs (benefits and workloads). We generalize the features to allow a job's benefits to be different while assigning to different machines, and machine capacities can be not the same.

2.3 Fairness

Fairness is an important issue for allocation problems, but the definition of fairness varies. There are different fairness indicators from past studies, and here we review three of them.

First of all, the Santa Claus problem has been studied by Bansal and Sviridenk (2006). In this problem, presents are prepared to be dispatched to kids. Since each kid has his own preference for different presents, how to maximize the utility of the kid with the minimum utility obtained from the allocated presents is the question. In this study, the fairness indicator is the minimum utility among all kids.

Second, Bertsimas et al. (2011) study two classic fairness schemes, proportional fairness and max-min schemes. Proportional fairness is the generalization of the Nash solution for a two-player problem, where a schedule is said to be fair if no reallocation can result in a new schedule that is proportionally fairer. It 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 with job benefits 7 and 5, respectively. If exchanging the two jobs makes the cumulative machine benefits become 6 and 6, then the aggregate proportion change is $\frac{6-7}{7} + \frac{6-5}{5} > 0$. We then say that the new schedule is proportionally fairer. In the setting of max-min fairness, a decision maker needs to do a sequence of optimization steps to maximize the

doi:10.6342/NTU201704133

minimum utility. At first, she maximizes the minimal utility of a player and makes sure that the utility of others at least arrives at the same level. Then, she maximizes the second minimal utility of a player in the same way. This procedure is repeated until the schedule can be no longer improved. The outcome of this procedure is considered as a fair schedule.

Third, the fairness indicator adopted by Deuermeyer et al. (1982) is the minimum total benefit generated among all machines, which they attempt to maximize. Enlightened by Deuermeyer et al. (1982), we revise the fairness indicator to be the smallest ratio among all machines, where the ratio equals to the total benefit generated by a machine divided by its capacity.

doi:10.6342/NTU201704133

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). For job j assigned to machine i, there is a workload $c_j > 0$ and a benefit $b_{ij} > 0$. That is, machine i has to spend c_j amount of time in order to earn b_{ij} for completing job j. The capacity of machine i is $K_i > 0$, which is the maximum amount of time it can spend. With the assumption that a job cannot be split to multiple machines, we try to assign jobs to machines which maximizes the minimum score among the machines. Here, we define the *score* of a machine as the total benefit of this machine divided by its capacity. More precisely, let

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

doi:10.6342/NTU201704133

be our decision variables. Our job allocation problem can be formulated as

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

The first constraint states that the total workloads assigned to each machine will not exceed its capacity, the second constraint ensures that each job is assigned to at most one machine, and the last constraint guarantees integrality. The objective function is to maximize the lowest score among all machines.

While solving the fair allocation problem in (3.1), we further investigate on how much efficiency is sacrificed when we maximize fairness. To see this, we consider the efficiency problem by replacing the objective function in (3.1) by

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

which is the total benefits generated by all machines. We call our main problem in (3.1) the *fairness problem* and the one with the alternative objective function the *efficiency problem*. To solve this, we propose an algorithm that effectively finds a near-optimal solution for our fairness problem and ensure a satisfactory efficiency level. We denote the solution to our algorithm as x_{ij}^{A}, the solution to genetic algorithm as x_{ij}^{G}, the optimal solution to the fairness problem as x_{ij}^{F}, and the optimal solution to the efficiency problem as x_{ij}^{E}. Let

$$s^{\mathrm{A}} = \min_{i\in I}\left\{\frac{\sum_{j\in J} b_{ij}x_{ij}^{\mathrm{A}}}{K_i}\right\}, \quad s^{\mathrm{G}} = \min_{i\in I}\left\{\frac{\sum_{j\in J} b_{ij}x_{ij}^{\mathrm{G}}}{K_i}\right\}, \quad s^{*} = \min_{i\in I}\left\{\frac{\sum_{j\in J} b_{ij}x_{ij}^{\mathrm{F}}}{K_i}\right\},$$

doi:10.6342/NTU201704133

Parameter	
I	The set of machines
J	The set of jobs
n	The number of jobs
m	The number of machines
K_i	The capacity of machine i ($K_i > 0$)
c_j	The finite workload of job j ($c_j > 0$)
b_{ij}	The finite benefit of job j assigned to machine i ($b_{ij} > 0$)

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

Table 3.1: List of notations

$$\text{and} \quad z^* = \sum_{i \in I} \sum_{j \in J} b_{ij} x_{ij}^{\text{E}}.$$

By plugging x_{ij}^{A} and x_{ij}^{G} separately into the objective function of the efficiency problem, we get $z^{\text{A}} = \sum_{i \in I} \sum_{j \in J} b_{ij} x_{ij}^{\text{A}}$ and $z^{\text{G}} = \sum_{i \in I} \sum_{j \in J} b_{ij} x_{ij}^{\text{G}}$, respectively. We will report how the objective value of our algorithm deviates from a fairest solution by calculating $\frac{s^{\text{A}}}{s^*}$ as well as the comparison of total benefits $\frac{z^{\text{A}}}{s^*}$. We will also report how genetic algorithm performs by calculating $\frac{s^{\text{G}}}{s^*}$ and $\frac{z^{\text{G}}}{s^*}$ to see if our algorithm is effective.

Table 3.1 lists the notations used in this model.

doi:10.6342/NTU201704133

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., $K_i \geq \sum_{j \in J} c_{ij}$, $i \in I$) machines and $n = 3m$ jobs with benefits $b_{ij} = b_j$ for all $i \in I, j \in J$. Suppose that there is a bound B which satisfies $\sum_{j=1}^{n} b_j = mB$. 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 completed. □

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

Proof. Our problem with one capacitated machine is a knapsack problem since there is only one machine with constant capacity and the objective function can be viewed as maximizing the total benefit the machine generates. □

doi:10.6342/NTU201704133

Chapter 4

Analysis

4.1 Iterative Score Maximization Algorithm with Repeated Testing algorithm—iSMART

To solve the problem, we design a heuristic algorithm to obtain an approximation solution. Inspired by the capacitated highest-benefit job first (CHBF) algorithm invented by Liu (2016), which assigns the job to the machine with the lowest cumulative benefit in each iteration, here we assign the job to the machine with the lowest score. We want to increase our objective value the most every time we assign a job, without violating the capacity constraints. Below we detail the steps.

In each iteration, we try to maximize the objective value, so we choose the machine with the lowest score and find the remaining unassigned jobs which generates the most benefit to that machine. If there are multiple machines with the same lowest score, we choose the one with minimum residual capacity. If the job cannot be assigned to the

doi:10.6342/NTU201704133

machine because the residual capacity on that machine is not enough to complete that job, we try the next job that generates the next most benefit to that machine. If all the remaining jobs cannot be assigned to that machine, then we choose the machine with the next lowest score and repeat the process. When a job cannot be assigned to any machine, it will be removed from the list of unassigned jobs. We reiterate this process until there is no job to assign. For ease of exposition, we denote this algorithm as the Iterative Score Maximization Algorithm with Repeated Testing—iSMART—algorithm. The pesdocode of iSMART is in Algorithm 1. We define $s_i = \frac{\sum_{j \in J} b_{ij} x_{ij}}{K_i}$ as the score for all machine i, l as the currently lowest score machine and h as the highest benefit job for machine l.

In practice, benefits generated by a job often correlate with machine quality and job complexity, i.e., $b_{ij} = q_i h c_j^t$, where q_i denotes machine quality, h is a scale factor and $t > 0$ elaborates the relationship between benefit and workload. Moreover, capacity varies from machine to machine, we assume $K_i = a_i K$, where $K > 0$ is the standard capacity of a machine and $a_i > 0$ illustrates the variation of each machine. There are usually way lot more jobs to be done with limited machine capacities, which is $\sum_{j \in J} c_j > \sum_{i \in I} K_i$. Without loss of generality, we assume that $c_1 \geq c_2 \geq \cdots \geq c_n$, $K_1 \geq K_2 \geq \cdots \geq K_m$ and $q_1 \geq q_2 \geq \cdots \geq q_m$. At the same time, $c_j \leq K_m$ for all $j \in J$, $c_n \geq 1$ and $K_i \geq 2$ for all $i \in I$ in all the following analyses.

Below we will discuss the iSMART algorithm in two classes of problems, each with three different worst-case performance guarantees for three different benefit-workload relationships. In the first class (C1), $q_i = 1$, job benefits do not variate between machines while machine capacities can be different. That is, $b_{ij} = b_j = hc_j^t$ with $K_i = a_i K$ for all $i \in I, j \in J$. In the second class (C2), $a_i = 1$, job benefits might be different within

doi:10.6342/NTU201704133

Algorithm 1 iSMART

```
 1: $J^{'} \leftarrow \emptyset$
 2: repeat
 3:     if $J^{'} = \emptyset$ then
 4:         $l \leftarrow \text{findLowestMachine}(I)$
 5:     end if
 6:     $h \leftarrow \text{findHighestJob}(J \setminus J', l)$
 7:     $J^{'} \leftarrow J^{'} \cup h$
 8:     if $K_l^{'} \geq c_h$ then
 9:         $x_{lh} \leftarrow 1,\ K_l^{'} \leftarrow K_l^{'} - c_h,\ J \leftarrow J \setminus h,\ J^{'} \leftarrow \emptyset$
10:     end if
11:     if $J^{'} = J$ then
12:         $I \leftarrow I \setminus l,\ J^{'} \leftarrow \emptyset$
13:     end if
14: until $I = \emptyset$ or $J = \emptyset$
15: return $\min_{i \in I}\{s_i\}$
```

Algorithm 2 findLowestMachine(I'')

```
1: $\min \leftarrow \infty, l \leftarrow 0$
2: for $i \in I''$ do
3:     if $s_i < \min$ or $s_i = \min\ \&\ K_i < K_l$ then
4:         $\min \leftarrow s_i,\ l \leftarrow i$
5:     end if
6: end for
7: return $l$
```

doi:10.6342/NTU201704133

Algorithm 3 findHighestJob(J'', l)

```
1: $\max \leftarrow 0, h \leftarrow 0$
2: for $j \in J''$ do
3:     if $b_{il} > \max$ then
4:         $\max \leftarrow b_{il}$, $h \leftarrow j$
5:     end if
6: end for
7: return $h$
```

machines while machine capacities are all the same. That is, $b_{ij} = q_i h c_j^t$ with $K_i = K$ for all $i \in I, j \in J$. Both of the two classes have three different relationships between benefit and workload: linear, convex, or concave. Linear benefit-workload relationship (R1) is for situations that benefits are proportional to workloads. Convex benefit-workload relationship (R2) occurs when jobs exhibit economy of scale, and thus a larger job generates relatively more benefits by the same machine. This is more likely the case if these jobs are manufacturing or production tasks. Finally, concave benefit-workload relationship (R3) models the situation in which the marginal benefit of a machine decreases in a unit of workload increase. This is a typical feature if the resulting products/services are to be sold to the end market. Chapter 4.4 is the time complexity of iSMART.

Class-Relationship	R1 ($t = 1$)	R2 ($t < 1$)	R3 ($t > 1$)
Liu (2016) ($q_i = 1$, $a_i = 1$)	-	-	-
C1 ($q_i = 1$)	§4.2.1	§4.2.2	§4.2.3
C2 ($a_i = 1$)	§4.3.1	§4.3.2	§4.3.3

Table 4.1: Section table

doi:10.6342/NTU201704133

4.2 Class 1: diverse machine capacity

For class 1 (C1), $q_i = 1$, job benefits are the same no matter assigned to which machine although machine capacities might be different. By assuming $c_1 \geq c_2 \geq \cdots \geq c_n$, we know that $b_1 \geq b_2 \geq \cdots \geq b_n$ for all machines according to $b_{ij} = q_i h c_j^t = h c_j^t = b_j$.

4.2.1 Relationship 1: linear benefit-workload relationship

We first prove that iSMART is a factor-$\frac{1}{2}$ approximation algorithm when job benefit and workload are proportional in the form $b_j = hc_j$ for some $h > 0$. Our algorithm can generate a solution whose objective value is at least $\frac{1}{2}$ of that from an optimal solution. We denote s^* as the objective value of an optimal solution and s^{A} as that of the iSMART solution.

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

$$s^{\mathrm{A}} \geq \frac{1}{2} s^*.$$

Proof. First, an obvious upper bound of s^* is $\frac{hK_i}{K_i} = h$, as all machines are assigned one job whose workload equals to its capacity. If we may prove that $s^{\mathrm{A}} \geq \frac{1}{2}h$, we will have $s^{\mathrm{A}} \geq \frac{1}{2}h \geq \frac{1}{2}s^*$, and the proof is complete. We prove by contradiction, suppose that $s^{\mathrm{A}} < \frac{1}{2}h$, and job j is the first job not assigned to its first-priority machine, say, machine i, the one with the lowest score currently. The cumulative benefit of machine i is then lower than $\frac{1}{2}hK_i$ at the time of assigning job j. This implies that $b_j < \frac{1}{2}hK_i$, since every job assigned before job j are with higher benefit, the cumulative benefit of machine i is also greater than b_j. Therefore, $c_j < \frac{K}{2}_i$ according to $b_j = hc_j$. However, if the cumulative

doi:10.6342/NTU201704133

benefit of machine i must also be lower than $\frac{1}{2}hK_i$, the total workloads of jobs that have been assigned to machine i must be less than $\frac{K}{2}_i$. This means the residual capacity of machine i is greater than $\frac{K}{2}_i$, and there is a room to assign job j. This violates our assumption that job j cannot be assigned to machine i. With this contradiction, the proof is complete, s^{A} must be greater or equal to half of the optimal s^*. □

Example 1 shows that the bound $\frac{1}{2}$ is tight.

Example 1. Let ϵ be a small positive number. Suppose that $q_i = 1$ for all i, $a_i = 1$ for all i, $h = 1$ and we have $2m$ jobs with workload $\frac{K}{2}$ and 1 job with workload $\frac{K}{2} + \epsilon$. We will use this setting in all the following examples. In this case, $K_i = K$ for all i and $b_j = c_j$ for all j. The iSMART algorithm will result in $m - 1$ machines being allocated 2 jobs but one machine leaving with only 1 job. The machine with only 1 job will earn score $s^{\mathrm{A}} = \frac{1}{2} + \frac{\epsilon}{K}$. However, the optimal solution is to allocate each machine with 2 jobs by ignoring the largest workload job, which results in $s^* = 1$. As a result, $\frac{s^{\mathrm{A}}}{s^*} = \frac{\frac{1}{2}+\frac{\epsilon}{K}}{1}$, which approaches $\frac{1}{2}$ as ϵ approaches 0.

4.2.2 Relationship 2: convex benefit-workload relationship

In this section, we prove that the iSMART algorithm has a performance guarantee when job benefits are convex in workloads in the form $b_j = hc_j^t$ for some $h > 0$ and $t > 1$. We define $\beta_{\mathrm{H}} \in (0, 1]$ according to the largest-workload job and smallest-capacity machine, and $\beta_{\mathrm{L}} \in (0, 1]$ according to the mth-largest-workload job and largest-capacity machine as

$$\beta_{\mathrm{H}} = \frac{c_1}{K_m},$$

doi:10.6342/NTU201704133

$$\beta_{\mathrm{L}} = \frac{c_m}{K_1}.$$

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*

$$s^* \leq \beta_{\mathrm{H}}^{t-1} h K_m^{t-1}.$$

Proof. Intuitively, we know that $s^* \leq \min\left\{\frac{hK_1^t}{K_1}, \frac{hK_2^t}{K_2}, \cdots, \frac{hK_m^t}{K_m}\right\} = hK_m^{t-1}$, where all machines are fully loaded with one job whose workload equals its capacity. This is clearly an upper bound. When $\beta_{\mathrm{H}} = 1$, we can easily understand that $s^* \leq hK_m^{t-1} = \beta_{\mathrm{H}}^{t-1} hK_m^{t-1}$.

When $0 < \beta_{\mathrm{H}} < 1$, the statement is still correct. For any machine, we know that the best way to consume machine capacity is to fill it by large workloads so it earns higher cumulative benefits according to $b_j = hc_j^t$. If all machines are filled with the same workloads, the final score will be restricted by the smallest-capacity machine. By our notation, the largest job's workload is $c_1 = \beta_{\mathrm{H}} K_m$. In this case, the final score will not be greater than assigning $\frac{1}{\beta_{\mathrm{H}}}$ jobs with the largest workload c_1 to the least capacitated machine. Thus, an upper bound of s^* is $\frac{\frac{1}{\beta_{\mathrm{H}}}\beta_{\mathrm{H}}^t hK_m^t}{K_m} = \beta_{\mathrm{H}}^{t-1} hK_m^{t-1}$. $\square$

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

$$s^{\mathrm{A}} \geq \min\left\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\right\} hK_m^{t-1}.$$

Proof. Suppose that job j is the first job that cannot be assigned to its first-priority machine, say, machine i, the one with the lowest score currently. We define p as the number of jobs that have been assigned to that minimum-benefit machine. For ease of exposition, let $\alpha = \min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}$ be the bound to be proved.

doi:10.6342/NTU201704133

By contradiction, we assume that $s^{\mathrm{A}} < \alpha h K_m^{t-1} \leq \alpha h K_i^{t-1}$. The cumulative benefit on machine i is thus less than $\alpha h K_i^t$. We know that the job j cannot be assigned to machine i, which has already been assigned p jobs by iSMART, we have $b_j \leq \frac{\alpha h K_i^t}{p}$, where the equality holds if and only if all the p jobs and job j are equally large. As $b_j = hc_j^t$, this implies that $c_j \leq \sqrt[t]{\frac{\alpha}{p}} K_i$. Therefore, the consumed capacity on machine i is at least $K_i - \sqrt[t]{\frac{\alpha}{p}} K_i$, otherwise machine i would have enough capacity for job j. Since benefits of jobs are now convex in their workloads, the least possible cumulative benefit of machine i is for the p jobs to be equally large. In this case, the cumulative benefit is $ph(\frac{K_i - \sqrt[t]{\frac{\alpha}{p}} K_i}{p})^t$.

We now show $ph(\frac{K_i - \sqrt[t]{\frac{\alpha}{p}} K_i}{p})^t \geq \alpha h K_i^t$ to establish a contradiction. Some arithmetic shows that $ph(\frac{K_i - \sqrt[t]{\frac{\alpha}{p}} K_i}{p})^t \geq \alpha h K_i^t$ if and only if $\alpha \leq \frac{p}{(p+1)^t}$. As the function $\frac{p}{(p+1)^t}$ is quasi-concave in p for any $t > 1$, and the smallest and largest possible numbers of jobs on machine i is 1 and $n - m - 1$, it implies that the global minimum of $\frac{p}{(p+1)^t}$ is either 1 or $n - m - 1$. Thus, $\alpha = \min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\} \leq \frac{p}{(p+1)^t}$ will always be satisfied. The proof is then complete. □

The lower bound of s^{A} is the minimum of two values. Whether $\frac{1}{2^t}$ or $\frac{n-m-1}{(n-m)^t}$ will be the smaller one depends on t. It can be easily verified that if $t \geq 2$, we have $\frac{n-m-1}{(n-m)^t} < \frac{1}{2^t}$ because $\frac{p}{(p+1)^t}$ is decreasing in that case. However, as long as $t < 2$, $\frac{p}{(p+1)^t}$ is increasing at $p = 1$, and it is possible that $\frac{n-m-1}{(n-m)^t} > \frac{1}{2^t}$. 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*

$$s^{\mathrm{A}} \geq h\beta_{\mathrm{L}}^t K_m^{t-1}.$$

doi:10.6342/NTU201704133

Proof. According to the iSMART algorithm, we assign the first mth jobs one at a time to the m machines. After that, the lowest score will be $\min\left\{\frac{hc_1^t}{K_m}, \frac{hc_2^t}{K_{m-1}}, \cdots, \frac{hc_m^t}{K_1}\right\}$, which is exactly $\frac{hc_m^t}{K_1} = h\beta_{\mathrm{L}}^t K_1^{t-1} \geq h\beta^t K_m^{t-1}$. The prove is then complete. □

Our main result, Theorem 4, is a direct combination of the above three lemmas.

Theorem 4. *If $b_j = hc_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have*

$$s^{\mathrm{A}} \geq \frac{\max\left\{\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}, \beta_{\mathrm{L}}^t\right\}}{\beta_{\mathrm{H}}^{t-1}} s^*.$$

Proof. According to Lemma 2 and 3, we have $s^{\mathrm{A}} \geq \max\{\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}, \beta_{\mathrm{L}}^t\}hK_m^t$. Also, we know that $s^* \leq h\beta_{\mathrm{H}}^{t-1}K_m^t$ by Lemma 1. Thus, the proof can be completed by combining all the three lemmas. □

When β_{L} is large, β_{L}^t would dominate $\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}$, and the performance guarantee can be trivially found as $\frac{\beta_{\mathrm{L}}^t}{\beta_{\mathrm{H}}^{t-1}}$. On the contrary, when β_{L} is small, $\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}$ would be a bound, and Lemma 1 further decreases this bound by having a denominator β_{H}^{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 the iSMART algorithm in achieving a better worst-case performance guarantee.

Example 2 shows that the bound $\frac{\max\left\{\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}, \beta_{\mathrm{L}}^t\right\}}{\beta_{\mathrm{H}}^{t-1}}$ is tight.

Example 2. Having the same settings as example 1, we get $K_i = K$ for all i and $b_j = c_j^t$ for all j. The iSMART algorithm will result in $m-1$ machines being allocated 2 jobs but one machine leaving with only 1 job. The machine with only 1 job will earn score $s^{\mathrm{A}} = \frac{(\frac{K}{2}+\epsilon)^t}{K}$. However, the optimal solution is to allocate each machine with 2 jobs by ignoring the largest workload job, which results in $s^* = \frac{2(\frac{K}{2})^t}{K}$. As a result, $\frac{s^{\mathrm{A}}}{s^*} = \frac{(\frac{K}{2}+\epsilon)^t}{2(\frac{K}{2})^t}$,

doi:10.6342/NTU201704133

which approaches $\frac{1}{2}$ as ϵ approaches 0. In this case, $\frac{\max\left\{\min\{\frac{1}{2^t},\frac{2m+1-m-1}{(2m+1-m)^t}\},\frac{1}{2^t}\right\}}{\left(\frac{\frac{K}{2}+\epsilon}{K}\right)^{t-1}} = \frac{\frac{1}{2^t}}{\left(\frac{\frac{K}{2}+\epsilon}{K}\right)^{t-1}}$, which also approaches $\frac{1}{2}$ as ϵ approaches 0.

4.2.3 Relationship 3: concave benefit-workload relationship

Here we prove the performance guarantee when the benefit is concave in workload in the form $b_j = hc_j^t$ for all $j \in J$ for some $h > 0$ and $t < 1$.

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

$$s^{\mathrm{A}} \geq \frac{K_m^{t-1}}{2^t} s^*.$$

Proof. The same as the proof in Theorem 3 and Theorem 4. First, we establish an upper bound for s^*. When the relationship between job benefits and workloads are concave, the most beneficial way to consume all the capacity of machine i is to use K_i jobs with unit workload. If all machines are assigned jobs of workload 1 until there is no more capacity, we will have $\frac{K_i h 1^t}{K_i} = h$ as the objective value, which is clearly an upper bound of s^*.

Then, we consider the moment of that the iSMART algorithm fails to assign a job to its first-priority machine, say, job j fails to be assigned to machine i. We prove by contradiction, by assuming that $s^{\mathrm{A}} < \frac{K_m^{t-1}}{2^t} s^* \leq \frac{K_m^{t-1}}{2^t} h \leq \frac{K_i^{t-1}}{2^t} h$. This implies that at this moment, machine i is with the lowest score and its cumulative benefit is lower than $\frac{K_i^t}{2^t} h$. b_j should be less than $\frac{K_i^t}{2^t} h$ as well, otherwise, it should have been assigned by iSMART already. As a result, we know $c_j < \frac{K_i}{2}$, and the residual capacity of machine i is less than $K_i - \frac{K_i}{2} = \frac{K_i}{2}$ so that job j fails to be assigned to machine i. For machine i, however, the fact that its current benefit is lower than $\frac{K_i^t}{2^t} h$ implies that the currently occupied

doi:10.6342/NTU201704133

capacity must be at most $\frac{K_i}{2}$ (which happens when machine i is assigned only one job). This immediately implies that the residual capacity is above $\frac{K_i}{2}$, which contradicts with the fact derived above. Therefore, we have $s^{\mathrm{A}} \geq \frac{K_m^{t-1}}{2^t} s^*$. □

Example 3. By example 2, $\frac{s^{\mathrm{A}}}{s^*} = \frac{(\frac{K}{2}+\epsilon)^t}{2(\frac{K}{2})^t}$ approaches $\frac{1}{2}$ as ϵ approaches 0. In this case, let $K_m = K = 2$, our bound $\frac{K_m^{t-1}}{2^t} = \frac{2^{t-1}}{2^t} = \frac{1}{2}$ is also tight.

4.3 Class 2: diverse job benefit

For class 2 (C2), $a_i = 1$, $K_i = a_i K = K$, job benefits might vary when assigning to different machine due to different quality while all machine capacities are the same. By assuming $c_1 \geq c_2 \geq \cdots \geq c_n$, we know that $b_{i1} \geq b_{i2} \geq \cdots \geq b_{im}$ for all machine i according to $b_{ij} = q_i h c_j^t$. Meanwhile, when $q_1 \geq q_2 \geq \cdots \geq q_m$, we have $b_{1j} \geq b_{2j} \geq \cdots \geq b_{nj}$ for all job j.

4.3.1 Relationship 1: linear benefit-workload relationship

In this section, when job benefit and workload are proportional (i.e., $b_{ij} = q_i h c_j$ for some $h > 0$), we prove that iSMART is a factor-$\frac{1}{2}$ approximation algorithm. We can generate an objective value that is at least half as good as an optimal solution.

Theorem 6. *If $b_{ij} = q_i h c_j$ for all $j \in J$ for some $h > 0$, we have*

$$s^{\mathrm{A}} \geq \frac{1}{2} s^*.$$

Proof. First, an upper bound of s^* is $\frac{q_m h K}{K} = q_m h$, as all machines are assigned one job whose workload equals to its capacity. This bound is restricted to the lowest quality

doi:10.6342/NTU201704133

machine. If we can prove that $s^{\mathrm{A}} \geq \frac{1}{2}q_m h$, we will have $s^{\mathrm{A}} \geq \frac{1}{2}q_m h \geq \frac{1}{2}s^*$, and the proof is complete. We prove by contradiction, suppose that $s^{\mathrm{A}} < \frac{1}{2}q_m h$, and job j is the first job not assigned to its first-priority machine, say, machine i, the one with the lowest score currently. The cumulative benefit of machine i is then lower than $\frac{1}{2}q_m hK$ at the time of assigning job j. This implies that $b_{ij} < \frac{1}{2}q_m hK$, since every job assigned before job j are with higher benefit, the cumulative benefit of machine i is also greater than b_{ij}. Therefore, $c_j < \frac{q_m}{2q_i}K \leq \frac{K}{2}$ according to $b_{ij} = q_i h c_j$ and $q_1 \geq q_2 \geq \cdots \geq q_m$. However, if the cumulative benefit of machine i must also be lower than $\frac{1}{2}q_m hK$, the total workloads of jobs that have been assigned to machine i must be less than $\frac{K}{2}$ as well. This means the residual capacity of machine i is greater than $\frac{K}{2}$, and there is a room to assign job j. This violates our assumption that job j cannot be assigned to machine i. With this contradiction, s^{A} must be greater or equal to half of the optimal s^*. □

Example 4 shows that the bound $\frac{1}{2}$ is tight.

Example 4. Using the same settings in the previous examples, we assume ϵ be a small positive number, $q_i = 1$ for all i, $a_i = 1$ for all i, $h = 1$ and we have $2m$ jobs with workload $\frac{1}{2}K$ and 1 job with workload $\frac{1}{2}K + \epsilon$. In this case, $b_{ij} = c_j$ for all i and j. iSMART will result in $m-1$ machines being allocated 2 jobs but one machine leaving with only 1 job. The machine with only 1 job will earn score $s^{\mathrm{A}} = \frac{1}{2} + \frac{\epsilon}{K}$. However, the optimal solution is to allocate each machine with 2 jobs by ignoring the largest workload job. This results in $s^* = 1$. As a result, $\frac{s^{\mathrm{A}}}{s^*} = \frac{\frac{1}{2}+\frac{\epsilon}{K}}{1}$, which approaches $\frac{1}{2}$ as ϵ approaches 0.

doi:10.6342/NTU201704133

4.3.2 Relationship 2: convex benefit-workload relationship

Here we prove that the iSMART algorithm has a performance guarantee when job benefits are convex in workloads in the form $b_{ij} = q_i h c_j^t$ for some $h > 0$ and $t > 1$. Using the same definition as in class 1, $\beta_{\mathrm{H}} = \frac{c_1}{K_m} = \frac{c_1}{K}$ and $\beta_{\mathrm{L}} = \frac{c_m}{K_1} = \frac{c_m}{K}$, where now $K_1 = K_m = K$. We rely on three lemmas to build our main result.

Lemma 4. *If $b_{ij} = q_i h c_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have*

$$s^* \leq \beta_{\mathrm{H}}^{t-1} q_m h K^{t-1}.$$

Proof. Intuitively, we know that $s^* \leq \min\left\{q_1 h K^{t-1}, q_2 h K^{t-1}, \cdots, q_m h K^{t-1}\right\} = q_m h K^{t-1}$, where all machines are fully loaded with one job whose workload equals its capacity. This is clearly an upper bound. When $\beta_{\mathrm{H}} = 1$, we can easily understand that $s^* \leq q_m h K^{t-1} = \beta_{\mathrm{H}}^{t-1} q_m h K^{t-1}$.

When $\beta_{\mathrm{H}} < 1$, the statement is still correct. For any machine, we know that for every machine the best way to consume its capacity is to fill it by large workloads so it earns higher cumulative benefits according to $b_{ij} = q_i h c_j^t$ for all i. By our notation, the largest job's workload is $c_1 = \beta_{\mathrm{H}} K$. In this case, the final score will not be greater than assigning $\frac{1}{\beta_{\mathrm{H}}}$ jobs with the largest workload c_1 to the lowest quality machine. Thus, an upper bound of s^* is $\frac{\frac{1}{\beta_{\mathrm{H}}} \beta_{\mathrm{H}}^t q_m h K^t}{K} = \beta_{\mathrm{H}}^{t-1} q_m h K^{t-1}$. $\square$

Lemma 5. *If $b_{ij} = q_i h c_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have*

$$s^{\mathrm{A}} \geq \min\left\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\right\} q_m h K^{t-1}.$$

Proof. Suppose that job j is the first job that cannot be assigned to its first-priority machine, say, machine i, the one with the lowest score currently. We define p as the

doi:10.6342/NTU201704133

number of jobs that have been assigned to that minimum-benefit machine. For ease of exposition, let $\alpha = \min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}$ be the bound to be proved.

By contradiction, we assume that $s^{\mathrm{A}} < \alpha q_m h K^{t-1}$. The cumulative benefit on machine i is thus less than $\alpha q_m h K^t$. We know that the job j cannot be assigned to machine i, which has already been assigned p jobs by iSMART, we have $b_{ij} \leq \frac{\alpha q_m h K^t}{p}$, where the equality holds if and only if all the p jobs and job j are equally large. As $b_{ij} = q_i h c_j^t$, this implies that $c_j \leq \sqrt[t]{\frac{\alpha q_m}{p q_i}} K \leq \sqrt[t]{\frac{\alpha}{p}} K$. Therefore, the consumed capacity on machine i is at least $K - \sqrt[t]{\frac{\alpha}{p}} K$, otherwise machine i would have enough capacity for 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 $ph(\frac{K - \sqrt[t]{\frac{\alpha}{p}} K}{p})^t \geq \alpha q_m h K^t$ to establish a contradiction. Some arithmetic shows that $ph(\frac{K - \sqrt[t]{\frac{\alpha}{p}} K}{p})^t \geq \alpha q_m h K^t$ 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 $n-m-1$, respectively. This implies that the global minimum of $\frac{p}{(p+1)^t}$ is either 1 or $n-m-1$, and thus $\alpha = \min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\} \leq \frac{p}{(p+1)^t}$ will always be satisfied. The proof is then complete. □

Lemma 6. *If $b_j = h c_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have*

$$s^{\mathrm{A}} \geq q_m h \beta_{\mathrm{L}}^t K^{t-1}.$$

Proof. According to the iSMART algorithm, we assign the first mth jobs one at a time to the m machines. After that, the lowest score will be $\min\left\{\frac{q_1 h c_1^t}{K}, \frac{q_2 h c_2^t}{K}, \cdots, \frac{q_m h c_m^t}{K}\right\}$, which is exactly $\frac{q_m h c_m^t}{K} = q_m h \beta_{\mathrm{L}}^t K^{t-1}$. The prove is then complete. □

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

doi:10.6342/NTU201704133

Theorem 7. *If $b_{ij} = q_i h c_j^t$ for all $j \in J$ for some given $h > 0$ and $t > 1$, we have*

$$s^{\mathrm{A}} \geq \frac{\max\left\{\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}, \beta_{\mathrm{L}}^t\right\}}{\beta_{\mathrm{H}}^{t-1}} s^*.$$

Proof. According to Lemma 5 and 6, we have $s^{\mathrm{A}} \geq \max\{\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}, \beta_{\mathrm{L}}^t\} q_m h K^t$. Also, we know that $s^* \leq \beta_{\mathrm{H}}^{t-1} q_m h K^t$ by Lemma 4. Thus, the proof can be completed by combining all the three lemmas. □

Example 5 shows that this bound is tight.

Example 5. Having the same settings as example 4, we get $b_{ij} = c_j^t$ for all i and j. iSMART will result in $m - 1$ machines being allocated 2 jobs but one machine leaving with only 1 job. The machine with only 1 job will earn score $s^{\mathrm{A}} = \frac{(\frac{K}{2}+\epsilon)^t}{K}$. However, the optimal solution is to allocate each machine with 2 jobs by ignoring the largest workload job. This results in $s^* = \frac{2(\frac{K}{2})^t}{K}$. As a result, $\frac{s^{\mathrm{A}}}{s^*} = \frac{(\frac{K}{2}+\epsilon)^t}{2(\frac{K}{2})^t}$, which approaches $\frac{1}{2}$ as ϵ approaches 0. In this case, $\frac{\max\left\{\min\{\frac{1}{2^t}, \frac{2m+1-m-1}{(2m+1-m)^t}\}, \frac{1}{2^t}\right\}}{\left(\frac{\frac{K}{2}+\epsilon}{K}\right)^{t-1}} = \frac{\frac{1}{2^t}}{\left(\frac{\frac{K}{2}+\epsilon}{K}\right)^{t-1}}$, which also approaches $\frac{1}{2}$ as ϵ approaches 0.

4.3.3 Relationship 3: concave benefit-workload relationship

Here we prove the performance guarantee when the benefit is concave in workload in the form $b_{ij} = q + ihc_j^t$ for all $j \in J$ for some $h > 0$ and $t < 1$.

Theorem 8. *If $b_{ij} = q_i h c_j^t$ for all $j \in J$ for some $h > 0$ and $t < 1$, we have*

$$s^{\mathrm{A}} \geq \frac{K^{t-1}}{2^t} s^*.$$

Proof. The same as the proof in Theorem 6 and Theorem 7. First, we establish an upper bound for s^*. When the relationship between job benefits and workloads are concave,

doi:10.6342/NTU201704133

the most beneficial way to consume all the capacity of machine i is to use K jobs with unit workload. If all machines are assigned jobs of workload 1 until there is no more capacity, we will have $\min\left\{\frac{Kq_1h1^t}{K}, \frac{Kq_2h1^t}{K}, \cdots, \frac{Kq_mh1^t}{K}\right\} = q_mh$ as the objective value, which is clearly an upper bound of s^*.

Then, we consider the moment of that the iSMART algorithm 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 $s^{\mathrm{A}} < \frac{K^{t-1}}{2^t}s^* \leq \frac{K^{t-1}}{2^t}q_mh$. This implies that at this moment, machine i is with the lowest score and its cumulative benefit is lower than $\frac{K^t}{2^t}q_mh$, and therefore the job benefit b_{ij} should be less than that as well (otherwise, it should have been assigned by iSMART already). As a result, we know $c_j < \frac{K}{2}\sqrt[t]{\frac{q_m}{q_i}} \leq \frac{K}{2}$, 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 $\frac{K^t}{2^t}q_mh$ 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 $s^{\mathrm{A}} \geq \frac{K^{t-1}}{2^t}s^*$. □

Example 6. This bound is also tight since here in class 2, $K_m = K$, and by example 3 we know that it is tight.

We summarize all the worst-case performance guarantees we obtain in Table 4.2. Numerical results are provided in Chapter 5.

doi:10.6342/NTU201704133

Class-Relationship	R1 ($t = 1$)	R2 ($t < 1$)	R3 ($t > 1$)
C1 ($q_i = 1$)	$\frac{1}{2}$	$\frac{\max\left\{\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}, \beta_{\mathrm{L}}^t\right\}}{\beta_{\mathrm{H}}^{t-1}}$	$\frac{K_m^{t-1}}{2^t}$
C2 ($a_i = 1$)	$\frac{1}{2}$	$\frac{\max\left\{\min\{\frac{1}{2^t}, \frac{n-m-1}{(n-m)^t}\}, \beta_{\mathrm{L}}^t\right\}}{\beta_{\mathrm{H}}^{t-1}}$	$\frac{K^{t-1}}{2^t}$

Table 4.2: Bounds

4.4 Time complexity analysis

In this section, we briefly derive the time complexity of iSMART. In each iteration, iSMART finds the lowest score machine and the job benefits this machine the most. That is, scanning through m machines and n jobs. We repeat the iteration n times so that all jobs are assigned. The time complexity of iSMART is then $O(mn^2)$. Numerical results of time are provided in Section 5.4.

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

Chapter 5

Numerical Study

5.1 Experiment settings

In our numerical study, we check both solution performance and time performance of iSMART on this fair job allocation problem by running experiments on a personal computer with Windows 10, 8GB RAM and Intel i7-6770 3.4GHz CPU. In all experiments, we set $c_j \sim U(0, 100)$, i.e., c_j is a randomly chosen real number between 0 and 100. We set up several factors to see how the solution varies in different settings. 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 machine quality. We consider two scenarios, where all machines are with the same quality, or there will be a variation. The third factor is capacity tightness. We consider two scenarios, one with loose machine capacity, and one with tight machine capacity. The fourth factor is capacity variation. We consider two scenarios, where all machines are with the same capacity, or there will

doi:10.6342/NTU201704133

be a variation. We adopt the following settings for each factor:

- Benefit-workload relationship (labelled as T): scenario L (for linear): $b_{ij} = q_i c_j$; scenario X (for convex): $b_{ij} = q_i c_j^2$; scenario A (for concave): $b_{ij} = q_i \sqrt{c_j}$; scenario R (for unrelated): $b_{ij} \sim U(0, 100)$.
- Machine quality (labelled as Q): scenario I (for identical): $q_i = 1$; scenario D (for diverse): $q_i \sim U(0.8, 1.2)$.
- Capacity tightness (labelled as C): scenario L (for loose): $K_i = a_i \left(\frac{\sum_{j \in J} c_j}{m}\right)$; scenario T (for tight): $K_i = \frac{3}{4} a_i \left(\frac{\sum_{j \in J} c_j}{m}\right)$.
- Capacity variation (labelled as A): scenario I (for identical) $a_i = 1$; scenario D (for diverse): $a_i \sim U(0.8, 1.2)$.

For the comparison of solution quality, we combine the above factors with several m and n ($m = 5$ with $n = 25,\ 50,\ 150$ and $m = 20$ with $n = 100,\ 200,\ 600$), as a total of 192 scenarios, each with 100 instances. For the comparison of computation time, we randomly select 100 instances for each different problem scales ($m = 5$ with $n = 20,\ 40,\ 60, ...,\ 400$ and $n = 400$ with $n = 5,\ 10,\ 15, ...,\ 40$).

5.2 Benchmark algorithms

We compare iSMART with IP or LP and genetic algorithm to see its performance. An IP solution is an optimal solution to our problem; we solve it with AMPL using the solver CPLEX. However, due to problem complexity and memory limitation, only one scenario in our study is IP solvable: $m = 5$ with $n = 20$ for computation time. When we are

doi:10.6342/NTU201704133

unable to get an IP solution, we release the binary constraint of our problem and solve an LP relaxation.

For genetic algorithm, We implement it as follows. First, we randomly create a pool of 100 feasible solutions. In each iteration we choose two pairs of parents (four chromosomes) from the best half of our pool and one pair of parents (two chromosomes) from the other half. Then we perform a crossover on each pair of parents by randomly select a cross-point which divides the selected solutions into the head part and tail part. Six child solutions are then created by connecting the head part to the another tail part between each pair of parents. All child solutions are given a 1% chance to mutate. When mutation happens, one job will change its destined machine randomly. We then check the feasibility of child solutions. At the end of each iteration, we compare feasible child solutions with those in the pool and replace the lower ones so that the pool remain 100 feasible solutions. This step makes the solutions in the pool better and better after iterations. The above process will be repeated 2000 times. At the end, the algorithm will report the best solution in the pool.

5.3 Comparison of solution quality

Too see how the solution quality is, we compare the solution between LP, iSMART and GA. We denote solution of "fairness" version as s^{L}, s^{A}, and s^{G}; solution of "efficiency" version as z^{L}, z^{A}, and z^{G}. They are then compared to generate Tables A.1-A.6 in Appendix. For the remaining, we use LP solutions because IP solutions are not attainable. To understand how each factor affects the performance of iSMART, we calculate the

doi:10.6342/NTU201704133

average performance of each scenario and then generate Tables 5.1, 5.2, 5.3, and 5.4. Moreover, Table 5.5 shows how iSMART is affected by different m and n.

In Table 5.1, in the "fairness" version, iSMART and GA perform the best when benefits are linear in workloads. For the iSMART algorithm, the performance falls when it comes to convex and unrelated relationship between benefits and workloads, and performs the least desirable in concave relationship. The reason behind this observation could be the way iSMART assign jobs, it chooses 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 iSMART to assign jobs to achieve "fairness" and "efficiency". When 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 iSMART. If the relationship between benefit and workload is concave, job benefits are lower when workloads are higher. iSMART puts the low cost-performance jobs first to the currently inferior machine, that might be why it performs worse than in linear and convex. However, for genetic algorithm, concave relationship is in its second best, while convex and unrelated relationship get the third and fourth, respectively.

In "efficiency" version, iSMART ranks benefit-workload relationship in the same sequence as that of fairness version. For genetic algorithm, the result is slightly different from that in fairness version, GA now performs better in concave than convex. In any case, data shows that iSMART performs relatively well than GA in both "fairness" and "efficiency" versions. We can also see that while pursuing fairness, efficiency remains in a high level and did not sacrifice too much using the iSMART algorithm.

Note that the convexity or concavity of the relationship between benefit and workload

doi:10.6342/NTU201704133

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, the product is of diminishing marginal benefit as workloads increase. This understanding will help managers to decide whether to choose the iSMART algorithm as their solution tool when they face the fair job allocation problem studied in this paper.

T	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
L	0.992	0.982	0.994	0.979
X	0.932	0.807	0.971	0.879
A	0.812	0.888	0.896	0.944
R	0.931	0.766	0.991	0.905

Table 5.1: Impact of the benefit-workload relationship (T)

Table 5.2 shows that iSMART has a better performance when machine qualities are the same, which is when the conversion rate is stable among machines. This will be helpful for iSMART to put the highest benefit job in the lowest score machine in each iteration, because it consumes a constant capacity no matter assigned to which machine.

Q	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
I	0.944	0.866	0.969	0.935
D	0.890	0.855	0.957	0.919

Table 5.2: Impact of machine quality (Q)

Table 5.3 shows that iSMART performs better when the capacity is loose comparing to tight. This is an intuitive result, because jobs can be assigned more appropriately by

doi:10.6342/NTU201704133

iSMART when it is more flexible to assign jobs.

C	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
L	0.945	0.899	0.985	0.965
T	0.888	0.822	0.941	0.889

Table 5.3: Impact of capacity tightness (C)

Table 5.4 is the results of capacity variation. The data reports that there is no big difference if there is a variation of capacity among machines or not.

A	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
I	0.917	0.860	0.963	0.962
D	0.917	0.861	0.963	0.928

Table 5.4: Impact of capacity variance (A)

Furthermore, by examining Table 5.5, we find that the performance ratios of iSMART is higher when $\frac{n}{m}$ increases. We believe that with more options of job candidates to be selected, our algorithm can assign more valuable jobs to machines.

$\frac{n}{m}$	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
5	0.896	0.820	0.957	0.910
10	0.922	0.867	0.966	0.931
30	0.932	0.896	0.967	0.940

Table 5.5: Impact of number of jobs and machines ($\frac{n}{m}$)

In summary, the average performance of all instances of iSMART is larger than 0.9,

doi:10.6342/NTU201704133

which is better than the average performance of GA. Numerical experiments show that iSMART is a good algorithm for this problem with high robustness. In particular, iSMART performs better when job benefits are linear or convex to workloads, machine qualities are stable, machine capacities are loose, and $\frac{n}{m}$ is large. These results will suggest decision makers when to apply the iSMART algorithm is suitable.

5.4 Comparison of computation time

We compare the average computation time for different problem scales in Tables 5.6 and 5.7. Figures 5.1 and 5.2 shows that the numerical experiments fit our analysis of time complexity in Section 4.4. With a fixed m, when n becomes larger, the computation time of iSMART increases in a quadratic manner. On the other hand, with a fixed n, when m becomes larger, the computation time of iSMART increases linearly. Moreover, we compared with iSMART and LP in Figures 5.3 and 5.4 and see that our algorithm runs fast while LP spends much longer time in solving when n and m are larger.

doi:10.6342/NTU201704133

n	IP	LP	iSMART	GA
20	55503.3	2.0	0.6	1372.5
40	-	3.8	1.2	2373.0
60	-	7.2	1.7	3427.3
80	-	12.7	2.2	4450.7
100	-	16.7	3.2	5418.3
120	-	26.1	4.2	6505.0
140	-	31.5	4.7	7441.4
160	-	35.2	5.9	8425.3
180	-	46.3	6.8	9572.8
200	-	52.5	8.2	10536.4
220	-	59.6	10.2	11743.7
240	-	60.3	10.5	12750.1
260	-	72.4	12.2	13884.4
280	-	81.7	13.9	14907.3
300	-	96.4	15.1	15741.2
320	-	107.9	16.6	17012.4
340	-	126.5	17.9	18429.6
360	-	133.6	19.7	19206.7
380	-	150.7	21.5	20269.4
400	-	160.6	23.3	21052.2

Table 5.6: Computation time (milliseconds) with fixed $m = 5$

doi:10.6342/NTU201704133

m	LP	iSMART	GA
5	160.6	23.3	21052.2
10	548.0	39.0	21276.6
15	953.9	51.1	21232.9
20	1655.2	69.6	21059.2
25	2606.2	83.9	21368.3
30	3875.1	104.5	21182.0
35	5500.3	117.8	21379.0
40	7894.7	135.6	21686.0

Table 5.7: Computation time (milliseconds) with fixed $n = 400$

doi:10.6342/NTU201704133

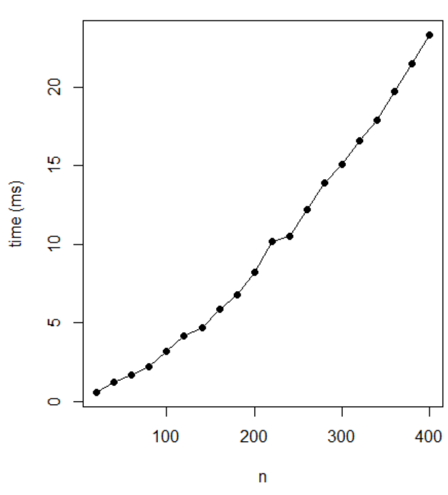

Figure 5.1: Computation time (ms) of iSMART with fixed $m = 5$

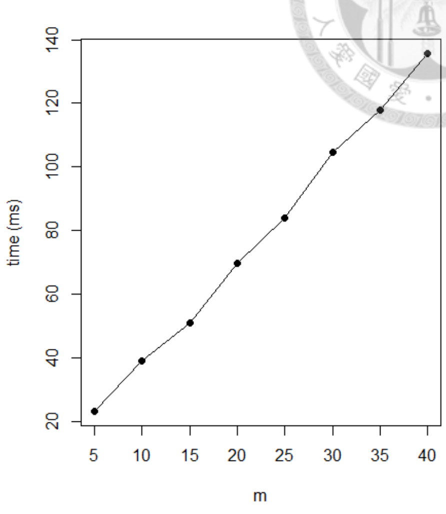

Figure 5.2: Computation time (ms) of iSMART with fixed $n = 400$

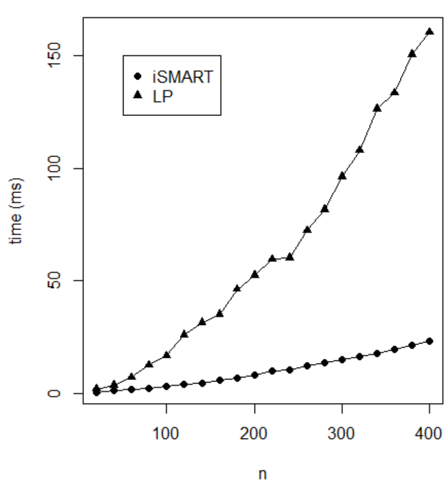

Figure 5.3: Computation time (ms) of iSMART and LP with fixed $m = 5$

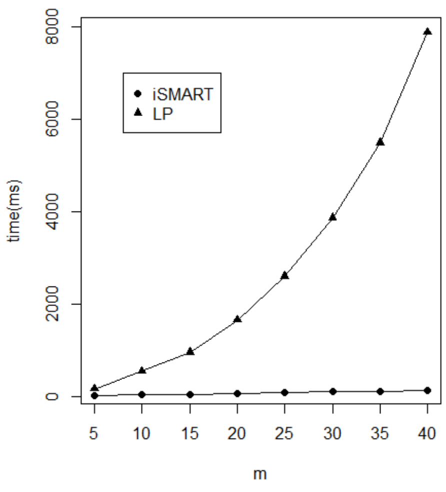

Figure 5.4: Computation time (ms) of iSMART and LP with fixed $n = 400$

doi:10.6342/NTU201704133

Chapter 6

Conclusion and Future Works

In this study, we consider a fair job allocation problem on unrelated machines as the main concern. Inspired by the previous literatures, we formulate an integer programming problem which maximizes the lowest fairness score of a machine subject to capacity constraints. Since the optimal solution cannot be solved in polynomial time, we then develop our own algorithm to solve this kind of problem. We prove that the performance guarantee of our algorithm is at least $\frac{1}{2}$ when the job benefits are linear ($t = 1$) in workloads, $\frac{\max\left\{\min\{\frac{1}{2^t},\frac{n-m-1}{(n-m)^t}\},\beta_{\mathrm{L}}^t\right\}}{\beta_{\mathrm{H}}^{t-1}}$ when the job benefits are convex ($t > 1$) in workloads, and $\frac{K_m^{t-1}}{2^t}$ when the job benefits are concave ($t < 1$) in workloads. Moreover, the numerical study indicates that our algorithms works better when benefits are linear or convex than concave, machine qualities are the same than different, capacity is loose than tight, and $\frac{n}{m}$ is big than small. It also reported that our algorithm did not sacrifice too much efficiency in order to pursue fairness.

Some further investigations may further improve in this study. One promising direction is to extend our problem settings, for example, job workloads might be different

doi:10.6342/NTU201704133

among machines. It would be more general if it is extended. Another direction is to remove some assumptions, such as $\sum_{j \in J} c_j > \sum_{i \in I} K_i$. If our algorithm still works well and there exist a performance guarantee, it would be more powerful to be adopted in practice.

doi:10.6342/NTU201704133

Appendix A

Supplemental Results of the Numerical Studies

doi:10.6342/NTU201704133

						average				minimum			
m	n	T	Q	C	A	$\frac{s^A}{s^L}$	$\frac{s^G}{s^L}$	$\frac{z^A}{z^L}$	$\frac{z^G}{z^L}$	$\frac{s^A}{s^L}$	$\frac{s^G}{s^L}$	$\frac{z^A}{z^L}$	$\frac{z^G}{z^L}$
5	25	L	I	L	I	0.957	0.937	0.972	0.953	0.902	0.882	0.927	0.909
5	25	L	I	L	D	0.975	0.960	0.988	0.974	0.927	0.897	0.953	0.914
5	25	L	I	T	I	0.964	0.962	0.977	0.976	0.882	0.929	0.916	0.945
5	25	L	I	T	D	0.970	0.963	0.984	0.975	0.902	0.899	0.938	0.910
5	25	L	D	L	I	0.992	0.993	0.981	0.937	0.958	0.947	0.912	0.874
5	25	L	D	L	D	0.990	0.994	0.986	0.962	0.961	0.951	0.945	0.884
5	25	L	D	T	I	0.989	0.997	0.979	0.955	0.950	0.969	0.929	0.889
5	25	L	D	T	D	0.988	0.997	0.981	0.957	0.942	0.974	0.933	0.892
5	25	A	I	L	I	0.914	0.914	0.958	0.949	0.861	0.890	0.908	0.887
5	25	A	I	L	D	0.927	0.935	0.965	0.968	0.791	0.870	0.868	0.916
5	25	A	I	T	I	0.827	0.890	0.854	0.935	0.709	0.823	0.776	0.874
5	25	A	I	T	D	0.798	0.897	0.842	0.936	0.711	0.850	0.751	0.866
5	25	A	D	L	I	0.812	0.906	0.955	0.921	0.669	0.858	0.928	0.862
5	25	A	D	L	D	0.842	0.920	0.947	0.936	0.723	0.852	0.860	0.867
5	25	A	D	T	I	0.738	0.881	0.839	0.905	0.568	0.819	0.743	0.854
5	25	A	D	T	D	0.735	0.883	0.820	0.906	0.619	0.806	0.756	0.854
5	25	X	I	L	I	0.967	0.885	0.991	0.939	0.894	0.712	0.970	0.825
5	25	X	I	L	D	0.965	0.888	0.993	0.948	0.885	0.743	0.955	0.790
5	25	X	I	T	I	0.890	0.732	0.934	0.799	0.757	0.502	0.854	0.668
5	25	X	I	T	D	0.894	0.736	0.947	0.797	0.796	0.532	0.888	0.623
5	25	X	D	L	I	0.913	0.873	0.959	0.899	0.833	0.756	0.934	0.805
5	25	X	D	L	D	0.919	0.886	0.959	0.907	0.777	0.702	0.925	0.741
5	25	X	D	T	I	0.836	0.734	0.918	0.779	0.705	0.564	0.828	0.656
5	25	X	D	T	D	0.844	0.737	0.935	0.778	0.719	0.606	0.862	0.641
5	25	R	I	L	I	0.937	0.872	0.990	0.962	0.805	0.716	0.969	0.841
5	25	R	I	L	D	0.928	0.879	0.991	0.959	0.765	0.739	0.964	0.829
5	25	R	I	T	I	0.863	0.756	0.969	0.848	0.675	0.605	0.916	0.699
5	25	R	I	T	D	0.871	0.760	0.969	0.853	0.722	0.558	0.930	0.672
5	25	R	D	L	I	0.925	0.877	0.991	0.958	0.743	0.729	0.962	0.902
5	25	R	D	L	D	0.920	0.875	0.991	0.956	0.700	0.741	0.958	0.832
5	25	R	D	T	I	0.867	0.758	0.971	0.851	0.658	0.613	0.938	0.704
5	25	R	D	T	D	0.855	0.758	0.968	0.847	0.595	0.610	0.922	0.653

Table A.1: The average and worst-case performance of iSMART and GA - $m = 5, n = 25$

doi:10.6342/NTU201704133

						average				minimum			
m	n	T	Q	C	A	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
5	50	L	I	L	I	0.990	0.967	0.993	0.975	0.969	0.945	0.974	0.957
5	50	L	I	L	D	0.994	0.983	0.998	0.991	0.985	0.951	0.990	0.956
5	50	L	I	T	I	0.989	0.988	0.993	0.993	0.951	0.969	0.977	0.978
5	50	L	I	T	D	0.993	0.987	0.996	0.992	0.985	0.970	0.990	0.977
5	50	L	D	L	I	0.998	0.999	0.995	0.968	0.993	0.973	0.986	0.925
5	50	L	D	L	D	0.998	0.998	0.993	0.977	0.986	0.974	0.964	0.937
5	50	L	D	T	I	0.997	0.999	0.996	0.980	0.987	0.991	0.983	0.908
5	50	L	D	T	D	0.998	0.999	0.996	0.981	0.984	0.996	0.990	0.928
5	50	A	I	L	I	0.969	0.956	0.985	0.975	0.950	0.936	0.965	0.937
5	50	A	I	L	D	0.969	0.967	0.982	0.988	0.897	0.938	0.903	0.958
5	50	A	I	T	I	0.810	0.923	0.827	0.946	0.780	0.897	0.783	0.913
5	50	A	I	T	D	0.813	0.920	0.830	0.944	0.725	0.874	0.759	0.892
5	50	A	D	L	I	0.814	0.930	0.975	0.946	0.709	0.875	0.958	0.914
5	50	A	D	L	D	0.834	0.941	0.954	0.952	0.707	0.853	0.861	0.916
5	50	A	D	T	I	0.713	0.897	0.810	0.914	0.608	0.843	0.761	0.855
5	50	A	D	T	D	0.717	0.896	0.808	0.914	0.606	0.837	0.745	0.854
5	50	X	I	L	I	0.996	0.936	0.999	0.970	0.989	0.895	0.992	0.926
5	50	X	I	L	D	0.993	0.933	0.998	0.965	0.966	0.838	0.988	0.888
5	50	X	I	T	I	0.971	0.796	0.980	0.830	0.937	0.715	0.947	0.738
5	50	X	I	T	D	0.964	0.791	0.980	0.827	0.938	0.673	0.959	0.715
5	50	X	D	L	I	0.947	0.918	0.964	0.932	0.870	0.865	0.940	0.877
5	50	X	D	L	D	0.945	0.921	0.961	0.929	0.849	0.828	0.927	0.806
5	50	X	D	T	I	0.898	0.781	0.964	0.807	0.818	0.685	0.943	0.696
5	50	X	D	T	D	0.887	0.785	0.964	0.810	0.825	0.678	0.939	0.690
5	50	R	I	L	I	0.967	0.923	0.998	0.978	0.877	0.866	0.991	0.918
5	50	R	I	L	D	0.966	0.922	0.998	0.974	0.764	0.824	0.989	0.898
5	50	R	I	T	I	0.921	0.780	0.987	0.839	0.792	0.614	0.960	0.677
5	50	R	I	T	D	0.926	0.781	0.988	0.844	0.722	0.620	0.896	0.720
5	50	R	D	L	I	0.968	0.919	0.998	0.978	0.879	0.855	0.992	0.929
5	50	R	D	L	D	0.969	0.927	0.998	0.978	0.885	0.814	0.988	0.901
5	50	R	D	T	I	0.922	0.791	0.988	0.851	0.737	0.697	0.969	0.735
5	50	R	D	T	D	0.916	0.787	0.988	0.850	0.746	0.655	0.955	0.740

Table A.2: The average and worst-case performance of iSMART and GA - $m = 5, n = 50$

doi:10.6342/NTU201704133

						average				minimum			
m	n	T	Q	C	A	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
5	150	L	I	L	I	0.999	0.990	0.999	0.993	0.996	0.982	0.997	0.987
5	150	L	I	L	D	0.999	0.995	1.000	0.998	0.998	0.985	0.999	0.988
5	150	L	I	T	I	0.999	0.998	0.999	0.999	0.996	0.996	0.998	0.997
5	150	L	I	T	D	0.999	0.998	1.000	0.999	0.998	0.996	0.999	0.997
5	150	L	D	L	I	1.000	1.000	1.000	0.990	0.999	0.999	0.999	0.976
5	150	L	D	L	D	1.000	0.999	0.995	0.992	0.998	0.983	0.966	0.969
5	150	L	D	T	I	1.000	1.000	1.000	0.997	0.999	1.000	0.999	0.981
5	150	L	D	T	D	1.000	1.000	1.000	0.997	0.998	0.999	0.999	0.993
5	150	A	I	L	I	0.994	0.982	0.997	0.992	0.989	0.972	0.992	0.986
5	150	A	I	L	D	0.965	0.983	0.970	0.993	0.878	0.961	0.888	0.978
5	150	A	I	T	I	0.809	0.931	0.812	0.944	0.778	0.905	0.783	0.916
5	150	A	I	T	D	0.806	0.930	0.811	0.942	0.754	0.889	0.761	0.899
5	150	A	D	L	I	0.804	0.928	0.984	0.965	0.694	0.851	0.974	0.947
5	150	A	D	L	D	0.816	0.940	0.949	0.965	0.673	0.843	0.855	0.938
5	150	A	D	T	I	0.693	0.882	0.795	0.918	0.590	0.796	0.763	0.876
5	150	A	D	T	D	0.691	0.878	0.796	0.917	0.594	0.798	0.747	0.862
5	150	X	I	L	I	1.000	0.972	1.000	0.990	0.999	0.956	1.000	0.976
5	150	X	I	L	D	0.999	0.963	1.000	0.979	0.995	0.884	0.997	0.906
5	150	X	I	T	I	0.993	0.830	0.994	0.849	0.988	0.791	0.990	0.810
5	150	X	I	T	D	0.991	0.831	0.994	0.848	0.986	0.782	0.990	0.798
5	150	X	D	L	I	0.952	0.933	0.963	0.953	0.903	0.871	0.933	0.918
5	150	X	D	L	D	0.958	0.936	0.962	0.945	0.884	0.859	0.914	0.896
5	150	X	D	T	I	0.910	0.805	0.976	0.830	0.854	0.751	0.963	0.781
5	150	X	D	T	D	0.906	0.804	0.975	0.829	0.849	0.725	0.959	0.753
5	150	R	I	L	I	0.989	0.961	1.000	0.993	0.958	0.937	0.999	0.981
5	150	R	I	L	D	0.990	0.955	1.000	0.983	0.950	0.849	0.997	0.887
5	150	R	I	T	I	0.952	0.788	0.992	0.824	0.862	0.694	0.980	0.726
5	150	R	I	T	D	0.951	0.786	0.992	0.823	0.863	0.691	0.969	0.731
5	150	R	D	L	I	0.990	0.961	1.000	0.993	0.965	0.926	0.999	0.976
5	150	R	D	L	D	0.988	0.954	1.000	0.984	0.941	0.850	0.998	0.898
5	150	R	D	T	I	0.952	0.786	0.991	0.821	0.870	0.729	0.979	0.764
5	150	R	D	T	D	0.952	0.789	0.992	0.828	0.894	0.709	0.972	0.757

Table A.3: The average and worst-case performance of iSMART and GA - $m = 5, n = 150$

doi:10.6342/NTU201704133

						average				minimum			
m	n	T	Q	C	A	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
20	100	L	I	L	I	0.965	0.902	0.978	0.953	0.936	0.806	0.951	0.902
20	100	L	I	L	D	0.982	0.905	0.993	0.960	0.958	0.852	0.976	0.917
20	100	L	I	T	I	0.961	0.928	0.977	0.967	0.918	0.880	0.945	0.933
20	100	L	I	T	D	0.984	0.930	0.993	0.969	0.963	0.895	0.983	0.945
20	100	L	D	L	I	0.998	0.980	0.994	0.942	0.993	0.925	0.979	0.909
20	100	L	D	L	D	0.998	0.982	0.993	0.951	0.979	0.934	0.975	0.897
20	100	L	D	T	I	0.997	0.992	0.992	0.957	0.987	0.968	0.961	0.926
20	100	L	D	T	D	0.997	0.992	0.994	0.959	0.984	0.965	0.983	0.935
20	100	A	I	L	I	0.916	0.854	0.963	0.953	0.883	0.811	0.934	0.930
20	100	A	I	L	D	0.935	0.856	0.981	0.959	0.860	0.816	0.932	0.924
20	100	A	I	T	I	0.832	0.829	0.867	0.932	0.725	0.785	0.789	0.901
20	100	A	I	T	D	0.785	0.824	0.840	0.932	0.747	0.791	0.800	0.899
20	100	A	D	L	I	0.769	0.810	0.967	0.923	0.716	0.770	0.951	0.895
20	100	A	D	L	D	0.772	0.815	0.958	0.927	0.679	0.759	0.902	0.889
20	100	A	D	T	I	0.701	0.784	0.847	0.901	0.587	0.742	0.790	0.870
20	100	A	D	T	D	0.678	0.786	0.819	0.901	0.603	0.738	0.758	0.870
20	100	X	I	L	I	0.981	0.774	0.995	0.931	0.946	0.704	0.976	0.889
20	100	X	I	L	D	0.975	0.768	0.997	0.927	0.943	0.652	0.985	0.855
20	100	X	I	T	I	0.905	0.629	0.931	0.786	0.821	0.533	0.905	0.722
20	100	X	I	T	D	0.917	0.639	0.966	0.792	0.853	0.548	0.932	0.714
20	100	X	D	L	I	0.887	0.752	0.956	0.892	0.801	0.654	0.947	0.834
20	100	X	D	L	D	0.882	0.749	0.955	0.893	0.810	0.659	0.940	0.832
20	100	X	D	T	I	0.809	0.619	0.918	0.769	0.724	0.532	0.888	0.710
20	100	X	D	T	D	0.801	0.611	0.946	0.763	0.716	0.527	0.930	0.678
20	100	R	I	L	I	0.922	0.649	0.996	0.960	0.807	0.541	0.991	0.915
20	100	R	I	L	D	0.925	0.651	0.996	0.959	0.801	0.560	0.987	0.868
20	100	R	I	T	I	0.847	0.530	0.982	0.836	0.624	0.428	0.968	0.756
20	100	R	I	T	D	0.852	0.524	0.982	0.831	0.652	0.417	0.963	0.739
20	100	R	D	L	I	0.928	0.658	0.996	0.958	0.809	0.576	0.988	0.922
20	100	R	D	L	D	0.926	0.649	0.995	0.954	0.750	0.547	0.982	0.880
20	100	R	D	T	I	0.847	0.522	0.982	0.834	0.645	0.411	0.967	0.765
20	100	R	D	T	D	0.854	0.530	0.982	0.835	0.642	0.396	0.969	0.739

Table A.4: The average and worst-case performance of iSMART and GA - $m = 20, n = 100$

doi:10.6342/NTU201704133

						average				minimum			
m	n	T	Q	C	A	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
20	200	L	I	L	I	0.992	0.953	0.995	0.979	0.984	0.930	0.991	0.964
20	200	L	I	L	D	0.997	0.960	0.999	0.988	0.987	0.890	0.992	0.967
20	200	L	I	T	I	0.992	0.977	0.995	0.990	0.977	0.966	0.985	0.979
20	200	L	I	T	D	0.997	0.977	0.999	0.990	0.990	0.962	0.997	0.984
20	200	L	D	L	I	1.000	0.998	0.999	0.973	0.997	0.984	0.997	0.955
20	200	L	D	L	D	0.999	0.997	0.996	0.980	0.998	0.965	0.980	0.960
20	200	L	D	T	I	0.999	0.999	0.999	0.985	0.998	0.994	0.997	0.971
20	200	L	D	T	D	0.999	0.999	0.999	0.985	0.996	0.994	0.997	0.969
20	200	A	I	L	I	0.971	0.910	0.988	0.977	0.960	0.894	0.974	0.967
20	200	A	I	L	D	0.969	0.914	0.986	0.984	0.910	0.884	0.926	0.969
20	200	A	I	T	I	0.801	0.865	0.818	0.941	0.785	0.846	0.797	0.926
20	200	A	I	T	D	0.800	0.870	0.824	0.942	0.769	0.835	0.787	0.918
20	200	A	D	L	I	0.752	0.843	0.977	0.948	0.689	0.812	0.970	0.921
20	200	A	D	L	D	0.766	0.853	0.962	0.953	0.667	0.817	0.904	0.927
20	200	A	D	T	I	0.662	0.812	0.798	0.913	0.628	0.772	0.775	0.885
20	200	A	D	T	D	0.670	0.816	0.803	0.915	0.603	0.769	0.764	0.889
20	200	X	I	L	I	0.998	0.864	0.999	0.968	0.994	0.825	0.998	0.946
20	200	X	I	L	D	0.995	0.859	0.999	0.971	0.985	0.808	0.996	0.928
20	200	X	I	T	I	0.977	0.720	0.984	0.825	0.960	0.665	0.964	0.789
20	200	X	I	T	D	0.968	0.718	0.985	0.826	0.948	0.665	0.979	0.783
20	200	X	D	L	I	0.919	0.819	0.955	0.927	0.875	0.784	0.939	0.907
20	200	X	D	L	D	0.912	0.817	0.955	0.928	0.856	0.757	0.932	0.878
20	200	X	D	T	I	0.852	0.689	0.966	0.802	0.803	0.631	0.953	0.758
20	200	X	D	T	D	0.852	0.692	0.964	0.806	0.810	0.633	0.957	0.756
20	200	R	I	L	I	0.959	0.757	0.999	0.982	0.866	0.696	0.998	0.966
20	200	R	I	L	D	0.959	0.755	0.999	0.977	0.869	0.679	0.996	0.938
20	200	R	I	T	I	0.896	0.620	0.993	0.836	0.773	0.545	0.985	0.795
20	200	R	I	T	D	0.908	0.614	0.992	0.834	0.727	0.535	0.984	0.765
20	200	R	D	L	I	0.957	0.756	0.999	0.981	0.878	0.691	0.997	0.963
20	200	R	D	L	D	0.961	0.759	0.999	0.979	0.842	0.677	0.996	0.934
20	200	R	D	T	I	0.902	0.623	0.992	0.837	0.768	0.558	0.981	0.785
20	200	R	D	T	D	0.904	0.610	0.992	0.832	0.748	0.507	0.983	0.753

Table A.5: The average and worst-case performance of iSMART and GA - $m = 20, n = 200$

doi:10.6342/NTU201704133

						average				minimum			
m	n	T	Q	C	A	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$	$\frac{s^{\mathrm{A}}}{s^{\mathrm{L}}}$	$\frac{s^{\mathrm{G}}}{s^{\mathrm{L}}}$	$\frac{z^{\mathrm{A}}}{z^{\mathrm{L}}}$	$\frac{z^{\mathrm{G}}}{z^{\mathrm{L}}}$
20	600	L	I	L	I	0.999	0.988	0.999	0.995	0.998	0.976	0.937	0.991
20	600	L	I	L	D	1.000	0.978	1.000	0.998	0.999	0.886	1.000	0.993
20	600	L	I	T	I	0.999	0.997	1.000	0.999	0.998	0.995	0.999	0.998
20	600	L	I	T	D	1.000	0.997	1.000	0.999	0.999	0.994	1.000	0.997
20	600	L	D	L	I	1.000	1.000	1.000	0.992	1.000	0.998	1.000	0.985
20	600	L	D	L	D	1.000	0.999	0.996	0.994	1.000	0.972	0.978	0.980
20	600	L	D	T	I	1.000	1.000	1.000	0.998	1.000	1.000	1.000	0.996
20	600	L	D	T	D	1.000	1.000	1.000	0.998	1.000	1.000	1.000	0.995
20	600	A	I	L	I	0.994	0.956	0.998	0.993	0.991	0.947	0.995	0.989
20	600	A	I	L	D	0.977	0.951	0.983	0.995	0.908	0.897	0.915	0.984
20	600	A	I	T	I	0.809	0.900	0.812	0.941	0.797	0.887	0.800	0.926
20	600	A	I	T	D	0.805	0.900	0.812	0.941	0.786	0.875	0.792	0.920
20	600	A	D	L	I	0.743	0.860	0.982	0.966	0.678	0.816	0.979	0.957
20	600	A	D	L	D	0.748	0.862	0.963	0.966	0.685	0.826	0.897	0.956
20	600	A	D	T	I	0.650	0.812	0.790	0.914	0.596	0.771	0.776	0.901
20	600	A	D	T	D	0.652	0.812	0.789	0.913	0.606	0.754	0.767	0.894
20	600	X	I	L	I	1.000	0.931	1.000	0.991	1.000	0.915	1.000	0.983
20	600	X	I	L	D	0.999	0.918	1.000	0.989	0.997	0.857	0.999	0.953
20	600	X	I	T	I	0.994	0.787	0.995	0.848	0.991	0.755	0.991	0.820
20	600	X	I	T	D	0.991	0.789	0.995	0.849	0.987	0.746	0.993	0.821
20	600	X	D	L	I	0.928	0.859	0.956	0.953	0.879	0.821	0.946	0.944
20	600	X	D	L	D	0.927	0.852	0.955	0.948	0.890	0.812	0.940	0.913
20	600	X	D	T	I	0.871	0.734	0.973	0.829	0.824	0.695	0.965	0.809
20	600	X	D	T	D	0.870	0.734	0.973	0.831	0.837	0.690	0.964	0.800
20	600	R	I	L	I	0.983	0.863	1.000	0.995	0.913	0.825	1.000	0.988
20	600	R	I	L	D	0.984	0.855	1.000	0.989	0.927	0.799	0.999	0.951
20	600	R	I	T	I	0.930	0.694	0.991	0.818	0.840	0.638	0.986	0.773
20	600	R	I	T	D	0.936	0.696	0.992	0.820	0.842	0.642	0.983	0.766
20	600	R	D	L	I	0.986	0.861	1.000	0.995	0.942	0.833	1.000	0.988
20	600	R	D	L	D	0.984	0.858	1.000	0.992	0.923	0.803	0.999	0.947
20	600	R	D	T	I	0.935	0.697	0.992	0.819	0.836	0.645	0.983	0.778
20	600	R	D	T	D	0.930	0.695	0.992	0.821	0.822	0.646	0.982	0.767

Table A.6: The average and worst-case performance of iSMART and GA - $m = 20, n = 600$

doi:10.6342/NTU201704133

doi:10.6342/NTU201704133

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/NTU201704133

minimum processor finish time in a multiprocessor system. *SIAM. J. on Algebraic and Discrete Methods* **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.

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

Liu, C.-W. 2016. Job allocation with a consideration of fairness. Master's thesis, National Taiwan University, Taipei, Taiwan.

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.

doi:10.6342/NTU201704133

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. 2016. Improved approaches to the exact solution of the machine covering problem. *Journal of Scheduling* **20**(2) 147–164.

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

doi:10.6342/NTU201704133