Coordinated Scheduling of Production and Distribution Operations

Models and Algorithms

生产与运输协同调度

模型与算法

By Jinwen Ou

欧锦文 ◎ 著

暨南大學出版社
JINAN UNIVERSITY PRESS

Coordinated Scheduling of Production and Distribution Operations

Models and Algorithms

生产与运输协同调度

模型与算法

By Jinwen Ou

欧锦文 ◎ 著

暨南大學出版社
JINAN UNIVERSITY PRESS

中国 · 广州

图书在版编目（CIP）数据

生产与运输协同调度：模型与算法 = *Coordinated Scheduling of Production and Distribution Operations*：*Models and Algorithms*/欧锦文著. —广州：暨南大学出版社，2015. 6
ISBN 978 - 7 - 5668 - 1435 - 7

Ⅰ. ①生…　Ⅱ. ①欧…　Ⅲ. ①交通运输经济—研究　Ⅳ. ①F5

中国版本图书馆 CIP 数据核字（2015）第 115481 号

出版发行：暨南大学出版社

地　址：中国广州暨南大学
电　话：总编室（8620）85221601
营销部（8620）85225284　85228291　85228292（邮购）
传　真：（8620）85221583（办公室）　85223774（营销部）
邮　编：510630
网　址：http：//www. jnupress. com　http：//press. jnu. edu. cn

排　版：广州市天河星辰文化发展部照排中心
印　刷：佛山市浩文彩色印刷有限公司

开　本：787mm ×960mm　1/16
印　张：11. 25
字　数：210 千
版　次：2015 年 6 月第 1 版
印　次：2015 年 6 月第 1 次

定　价：29. 80 元

（暨大版图书如有印装质量问题，请与出版社总编室联系调换）

Preface

The coordination of logistics activities in a supply chain has received a lot of attention in operations management research. In a typical supply chain, raw materials are transported to factories where items are produced, and finished products are either shipped to warehouses for temporary storage or delivered directly to the retailers or the customers. To achieve better operational performance, the coordination and integration of production and distribution is an important consideration.

In this book, we study five models of coordinated scheduling of production and distribution operations. The motivations of five models are different.

In Chapter 1, we consider an order acceptance and scheduling model with job delivery times and machine availability constraints. Order acceptance and scheduling (OAS) is to balance order acceptance (to increase revenue and guarantee satisfaction of customers) and order rejection (to reduce production or delivery load so as to meet limited production capacity restrictions and tight delivery requirements). In our model, due to specific production constraints of machine availability and the delivery times of finished orders, some orders (jobs) have to be rejected or outsourced.

In Chapter 2, we study an inbound logistics scheduling model with limited unloading capacity, which is motivated by the inbound logistics system for a major automobile manufacturing company in China. The research problem can be reduced to a parallel machine scheduling problem with multiple common unloading servers. In the proposed scheduling model, after processing a job, an unloading server will be needed to remove the job from the machine. Only after its current job is unloaded, a machine can start processing the next job.

In Chapter 3, we study machine scheduling with raw material pickup and finished job delivery. Traditional machine scheduling models focus on the optimization of schedules under various machine environments, processing characteristics and constraints, and performance measures. However, these models normally ignore the transportation arrangements of the incoming materials and the delivery arrangements of the finished products. In this chapter, we develop and analyze a scheduling model that takes into account the machine schedule and the pickup and delivery plan of the supply of the raw materials and the shipment of finished jobs.

In Chapter 4, we turn to the study of coordinated scheduling of customer orders with decentralized machine locations. In many production and transportation planning environments, the delivery of finished goods is constrained by the processing of different component parts of the product. It is also quite common that different components of a product have to be processed by their own dedicated machines or work centers. For example, in the production of personal computer systems, the computers and monitors are usually produced by different facilities at different locations. However, both the computers and monitors of the finished products must be bundled together before they can be delivered to customers. In order to obtain a system-wide optimal production and delivery plan, it is essential to consider the sequencing and scheduling of the tasks at each machine and the delivery arrangements of the finished tasks to their final destinations at the same time. When such an integrated plan is developed, the scheduler faces a tradeoff between providing quick deliveries and minimizing shipping costs. Quick deliveries minimize customers’ waiting time while low shipping costs directly benefit the company's bottom line. Thus, we consider a scheduling which reflects such production and delivery arrangements.

In Chapter 5, we study the coordination of cargo unloading and storage arrangements at cargo terminals. We specifically consider a problem which is motivated by the operations of air cargo terminal operators at the Hong Kong International Airport (HKIA). In this problem, there are a number of outbound flights with confirmed air waybills. The terminal operator would like to schedule the arrival of the delivery trucks so that some of the shipments can be transferred directly to the departing flights, while other shipments will be transferred to the storage area of the terminal and will incur extra handling and storage costs. The objective is to obtain a feasible schedule so as to minimize the total cost of operations.

Contents

Chapter 1
Order Acceptance and Scheduling with Job Delivery Times and Machine Availability Constraints

1.1 Introduction

Order acceptance and scheduling (OAS) has attracted considerable attention from scheduling researchers as well as production managers who practice it in the past decades. The key issue in OAS is to balance order acceptance (to increase revenue and guarantee satisfaction of customers) and order rejection (to reduce production or delivery load so as to meet production capacity or delivery requirements). One important application of OAS arises in make-to-order production systems with limited production capacity or tight delivery requirements, where order rejection and scheduling decisions have to be made simultaneously. In practice, due to production capacity constraints and short delivery due dates, a typical manufacturer may have to reject some orders which require long processing or delivery times but make relatively small profits. Another important application of OAS occurs in scheduling when outsourcing is an alternative option. OAS combines outsourcing and production scheduling decisions together.

In traditional OAS models the manufacturer (machine) is usually assumed to have the capacity of processing orders continuously. But in reality, production could be interrupted due to some unpredictable reasons such as machine breakdown or maintenance, production setup, worker vacation and so on, which leads to the scenario that there are only a number of discontinuous time intervals available to process the orders in question. When production capacity and machine available time have restrictions, to reduce production load, the manufacturer may have to reject some orders which require long processing times but make relatively small profits, or outsource the production of some orders, in this case the profit of those outsourced orders is reduced. In this chapter, we study a single-machine OAS model with machine availability constraints and order delivery times. In the proposed model, the machine is available to process orders only within a number of given discontinuous time intervals, and the manufacturer is allowed to reject some orders. When an order is rejected, an order-dependent penalty will occur. Each order has a specific delivery times after its processing. The manufacturer needs to balance the production capacity and the delivery requirement (the time span of delivering all accepted orders to their customers) and the total penalty of all rejected jobs.

OAS has been studied extensively in the production scheduling literature. Guerrero and Kern(1988) provided the rational to accept and reject orders effectively. Slotnick and Morton (1996) and Ghosh (1997) studied single-machine OAS with the objective of maximizing revenue and minimizing lateness penalties. Slotnick and Morton (2007) extended their previous work (Slotnick and Morton, 1996) by replacing lateness with tardiness in the objective function. Rom and Slotnick (2009) developed a genetic algorithm for the same problem and compared it with the myopic heuristic given in Slotnick and Morton (2007). Nobibon and Leus (2011) studied an OAS model with "firm planned orders as well as potential orders". Oguz et al. (2010) included sequence-dependent setup times and two-level due-dates in an OAS model with weighted tardiness. Cesaret et al. (2012) developed a tabu algorithm for the same problem and compared it with the two heuristics proposed in Oguz et al. (2010). Yang and Geunes (2007) considered an OAS model with controllable processing times. Chen and Li (2008), Lee and Sung (2008a, 2008b) and Qi (2008, 2009, 2011) considered OAS models with outsourcing options. An excellent literature survey on the topic of OAS is provided by Slotnick (2011) recently.

OAS is equivalent to machine scheduling with rejection (MSR) in mathematics. Bartal et al. (2000) first introduced a multi-processor MSR model to minimize the makespan of all accepted jobs plus the total rejection penalty of all rejected jobs. Seiden (2001) and Hoogeveen et al. (2003) studied the on-line version and the off-line version in Bartal et al. (2000) when job preemption is allowed, respectively. Epstein et al. (2002) considered on-line MSR with unit-processing-time jobs and the total completion time of all accepted jobs. Engels et al. (2003) studied single-processor MSR with total weighted completion times of all accepted jobs, and recently, Kellerer and Strusevich (2013) improved the related results. MSR models with industrial applications are studied by Cheng and Sun (2009), Zhang et al. (2009,2010), Lu et al. (2008) and Lu et al. (2009), among others. A very recent survey on MSR is provided by Shabtay et al. (2013). All of the OAS and MSR models mentioned above have the potential assumption that machines can process jobs continuously, which differ from our model.

Our model is an extension of machine scheduling with availability constraints (MSAC). MSAC also has been studied extensively. The surveys of MSAC are provided by Schmidt (2000) and Ma et al. (2010). In this paper we consider a single-machine model with machine availability constraints. In the following we only review the previous work of MSAC with a single machine, and whose objective is limited to minimization of makespan. For convenience, we define MSAC-k to be the MSAC problem to minimize makespan with a single machine and k time intervals during which the machine is not allowed to process jobs. Lee (1996) showed that MSAC-1 has been NP-hard, and that MSAC-k is NP-hard in the strong sense if k is arbitrary. He et al. (2005) proposed a fully polynomial time approximation scheme (FPTAS) for MSAC-1. Breit et al. (2003) showed that no polynomial time approximation algorithm with a fixed performance ratio exists for MSAC-2. When every job also has a delivery time, Yuan et al. (2008) showed that MSAC- k can be solved by a pseudo-polynomial time algorithm if k is fixed. They also developed a polynomial time approximation scheme (PTAS) for MSAC-1. Kacem (2009) studied a similar problem to Yuan et al. (2008), and proposed an FPTAS by exploiting the well-known approach of Ibarra and Kim (1975). Wu and Lee (2003), Gawiejnowicz (2007), and Ji et al. (2006) studied MSAC with deterioration, where the processing time of a job is a linear function of its starting time. To the best of our knowledge, our model is the first to examine OAS with machine availability constraints.

Our model is also an extension of machine scheduling with job delivery times. Machine scheduling with job delivery times is also a classical scheduling model (see, for example, Hall and Shmoys, 1989, Jackson, 1955, Kise et al., 1979, Lawler et al., 1993, and among others), in which each job is delivered to its customer at its processing completed time by a specific vehicle with a given delivery time. For machine scheduling with delivery times, a classical result is that, the problem $1 \mid q_j \mid D_{\max}$ can be solved in $O(n\log n)$ time by using the "largest delivery time first" (LDT) rule (see Jackson, 1955). Lawler et al. (1993) prove that problem $1 \mid r_j, q_j \mid D_{\max}$ is strongly NP-hard. For the same problem, Kise et al. (1979) show that LDT rule is 2-competitive, and Hall and Shmoys (1989, 1992) develop two polynomial time approximation schemes. Mastrolilli (2003) develops an improved approximation scheme for the same problem. When preemption is allowed, it is shown in Baker et al. (1983) that problem $1 \mid prec, pmtn, r_j, q_j \mid D_{\max}$ can be solved in $O(n^2)$ time.

We now describe our problem formally as follows. Given a set of n jobs $J = \{J_1,$

$J_2, \ldots, J_n\}$ and a single machine, each job $J_j \in J$ is associated with a processing time p_j, a job delivery time q_j and a rejection penalty w_j. Job J_j is either accepted and then processed on the machine, or it is rejected by the machine and then a rejection penalty w_j is paid. Penalty w_j is regarded as the cost of losing or outsourcing job J_j. Also given $2m + 1$ non-negative integers $a_0, a_1, \ldots, a_{2m}$ such that $0 = a_0 < a_1 < \ldots < a_{2m}$, we assume that in the planning horizon the machine is available to process jobs within time intervals $[a_{2m}, +\infty)$ and $[a_{2i-2}, a_{2i-1}]$ for $i = 1, 2, \ldots, m$, but it is not allowed to process any job during time interval (a_{2i-1}, a_{2i}) for any $i = 1, 2, \ldots, m$. In other words, there are m UIs (time intervals during which the machine is not allowed to process jobs) and $m + 1$ AIs (time intervals during which the machine is allowed to process jobs) in the planning horizon. Let A be the set of jobs accepted, and $R = J\backslash \mathrm{A}$ be the set of jobs rejected. Let C_j be the completion time of job J_j if $J_j \in A$, and let $D_j = C_j + P_j$ be the completion time of delivery of job J_j to its customer. The problem is to determine A and a feasible schedule of jobs in A on the machine so as to minimize the objective function $\mathrm{Z} = \max\{D_j \mid J_j \in A\} + \sum_{J_j \in R} w_j$, i. e., the completion time of the delivery of all accepted jobs plus the total penalty of all rejected jobs.

In our model, we assume that the machine can process at most one job at a time, that job preemption is not allowed, and that p_j and w_j are non-negative integers for $j = 1, 2, \ldots, n$. Following the standard classification scheme for scheduling problems [7,18], the proposed scheduling problem is denoted as

$$1 \mid \mathrm{FB}, \mathrm{rej}, q_j \mid D_{\max} + \sum w_j,$$

where "FB" means that accepted jobs must be processed in free time slots to avoid forbidden intervals, while "rej" means that job rejection is allowed. When m (i. e., the number of UIs) is a fixed number, we denote the problem as

$$1 \mid \mathrm{FB}(m), \mathrm{rej}, q_j \mid D_{\max} + \sum w_j.$$

When $q_j = 0$ for each job, i. e., the job delivery time of each job is zero, we denote the problem as

$$1 \mid \mathrm{FB}, \mathrm{rej} \mid C_{\max} + \sum w_j.$$

Machine scheduling with forbidden intervals (machine maintenances) has been widely studied in the past two decades. NP-hardness results of problem $1 \mid FB \mid C_{\max}$ are provided by Lee (1996) and Scharbrodt et al. (1999). For problem $1 \mid FB(1) \mid C_{\max}$, the performance ratio of the LPT heuristic is shown to be $\frac{4}{3}$, and the bound is tight (Lee, 1996). A fully polynomial time approximation scheme is given by He et al. (2005). Some related work on scheduling with forbidden intervals can also be found in the study of Cassady and Kutanoglu (2003), Chen(2006), Graves and Lee(1999). A survey on machine scheduling with forbidden intervals is provided by Schmidt (2000). Study of a relaxed version in which machine maintenances are not fixed previously can be found in Akturk et al. (2003, 2004), Qi (2007), Qi et al. (1999). Note that when job preemption is allowed, problem $1 \mid FB(m), pmtn, q_j \mid D_{\max}$ is solved in $O(n\log n + m\log m)$ time by Yuan and Lin (2005).

For convenience, we introduce the following notations, which will be used throughout the paper:

$P = \sum_{J_j \in J} p_j$ = the total processing time of all the jobs;

$W = \sum_{J_j \in J} w_j$ = the total rejection penalty of all the jobs;

$U = \{J_j \in J \mid w_j > p_j\}$ = the set of jobs with processing times less than their rejection penalty;

$V = \{J_j \in J \mid w_j \leqslant p_j\}$ = the set of jobs with processing times no less than their rejection penalty;

$P(U) = \sum_{J_j \in U} p_j$ = the total processing time of all the jobs in U;

σ^* = the optimal solution;

A^* = the set of all accepted jobs in σ^*;

$R^* = J \backslash A^*$ = the set of all rejected jobs in σ^*;

$C^*_{\max}$ = the makespan of A^*, i. e., the completion time of the last accepted job in A^*;

$D_{\max}$ = the completion time of delivery for all the jobs in A^*;

$Z = \max\{D_j \mid J_j \in A\} + \sum_{J_j \in R} w_j$ = the objective function value of σ^*;

$a_{2m+1} = a_{2m} + P$ = the largest possible makespan of all accepted jobs in σ^*;

$L_i = a_{2i} - a_{2i-1}$ = the length of the i-th UI ($i = 1, 2, \ldots, m$);

$L'_i = a_{2i-1} - a_{2i-2}$ = the length of the i-th AI ($i = 1, 2, \ldots, m+1$);

ε: any given small constant such that $\frac{1}{\varepsilon}$ is a positive integer.

The rest of the chapter is organized as follows. In Section 1.2 we develop a pseudo-polynomial algorithm to solve problem $1 \mid FB(m), rej, q_j \mid D_{\max} + \sum w_j$. In Section 1.3 we provide non-approximability results to problem $1 \mid FB(2), rej, q_j \mid D_{\max} + \sum w_j$. In Section 1.4 we present an FPTAS for the special case $1 \mid FB(1), rej, \mid C_{\max} + \sum w_j$. In Section 1.5 we present a polynomial time approximation scheme (PTAS) for the special case $1 \mid FB(1), q_j \mid D_{\max}$. In Section 1.6 we develop a $(2+\varepsilon)$-approximation for the special case with $L_i' = L'$, $L_i = L$ for $i = 1, 2, \ldots, m$ and $q_j = 0$ for $j = 1, 2, \ldots, n$. Some concluding remarks are provided in Section 1.7.

1.2 An Optimal Pseudo-polynomial Algorithm When m Is Fixed

Note that if m is arbitrary, then problem $1 \mid FB, rej, q_j \mid D_{\max} + \sum w_j$ is shown to be NP-hard in the strong sense by reducing from 3-PARTITION (even when each $q_j = 0$), and thus an optimal pseudo-polynomial algorithm for $1 \mid FB, rej, q_j \mid D_{\max} + \sum w_j$ does not exist (see Garey and Johnson, 1979). Therefore, in the remainder of this section, we study the special case when the value of m is fixed. We present a pseudo-polynomial algorithm to solve $1 \mid FB(m), rej, q_j \mid D_{\max} + \sum w_j$.

It is clear that the accepted jobs that are scheduled within the same AI follow the "largest delivery time first" rule. A schedule is called an LDT schedule if the accepted jobs that are scheduled within each AI follow the LDT rule. In the development of our algorithm, we only need to determine an optimal LDT schedule. Reindex the n jobs in an LDT order, i. e., $q_1 \geqslant q_2 \geqslant \ldots \geqslant q_n$. We now develop our pseudo-polynomial time algorithm for problem $1 \mid FB(m), rej, q_j \mid D_{\max} + \sum w_j$. In the following of this section, when we mention a solution or schedule, we always mean an LDT solution or schedule. Without loss of generality, we can assume that $L'_i < P$ for $i = 1, 2, \ldots, m$, i. e., the length of the i-th AI is no greater than the total processing time of the n jobs (note that if $L'_i < P$ for some $i \in \{1, 2, \ldots, m\}$, then it is clear that no accepted job will be scheduled within or after the $(i+1)$-th AI; in such a case, we only need to let $m = i$ and only consider the first i AIs). Furthermore, we also can assume that there are some accepted jobs scheduled within the last AI $[a_{2m}, a_{2m+1}]$ (if not, then for some $i \in \{1,2,\ldots,m-1\}$, no accepted job will be scheduled after the i-th AI; in such a case, we spend $O(m)$ time to make a guess on the value of i, and only consider the first i AIs by letting $m=i$).

Our DP will keep tracking the total processing time of the jobs scheduled within each of the $m+1$ AIs when considering the n jobs one by one. Let $q_{\max} = \{q_j \mid J_j \in J\}$. To simplify our DP, we add a dummy job J_0 with zero processing time and zero job delivery time. The dummy job J_0 is fixed to be scheduled within the last AI $[a_{2m}, a_{2m+1}]$so that it is the first job to be scheduled in the AI (i. e. , the completion time of the delivery of J_0 is equal to a_{2m}).

We now present our DP. Denote the problem instance containing only the first j jobs $J_1, J_2, \ldots, J_j$ by I_j. For any $j=1,2,\ldots,n$, $D = a_{2m}, a_{2m}+1, \ldots, a_{2m+1}+q_{\max}$, $i = 1,2,\ldots,m+1$ and $g_i = 0,1,\ldots,L'_i$, define $G_j(g_1, g_2, \ldots, g_{m+1}, D)$ as the minimum total cost to I_j such that the total processing time of all accepted jobs assigned to AI $[a_{2i-2}, a_{2i-1}]$ is equal to g_i for $i=1,2,\ldots,m+1$, and the completion time of the delivery of all accepted jobs among the first j jobs is equal to D. To develop recursive relations, consider the following two cases:

Case A: Job J_j is rejected. In this case, we have $G_j(g_1, g_2, \ldots, g_{m+1}, D) = G_{j-1}(g_1, g_2, \ldots, g_{m+1}, D) + w_j$.

Case B: Job J_j is accepted and scheduled to the i-th AI for some $i=1,2,\ldots, m+1$. In this case, J_j is the last accepted job to be processed within the i-th AI currently, and we should have $g_i \geqslant p_j$ *and* $a_{2i-2} + g_i + q_j \leqslant D$. Thus, we have $G_j(g_1, \ldots, g_i, \ldots, g_{m+1}, D) = G_{j-1}(g_1, \ldots, g_i - p_j, \ldots, g_{m+1}, D)$.

Combining the two cases above, we have the following dynamic program DP_1:

(1) Recurrence relation: For $j=1,2,\ldots,n$, $i=1,2,\ldots,m+1$ and $g_i = 0, 1, \ldots, L'_i$, $G_j(g_1, g_2, \ldots, g_{m+1}, D) =$

$$\min\left\{G_{j-1}(g_1, g_2, \ldots, g_{m+1}, D) + w_{\mathrm{j}},\ \min_{\substack{i=1,2,\ldots,m+1\\ p_j \leqslant g_i \leqslant D-q_j-a_{2i-2}}} \{G_{j-1}(g_1, \ldots, g_i - p_j, \ldots, g_{m+1}, D)\}\right\}.$$

(2) Boundary condition:

$$G_0(g_1, g_2, \ldots, g_{m+1}, D) = \begin{cases} 0, & \textit{if } g_1 = \ldots = g_{m+1} = 0 \textit{ and } D = a_{2m}; \\ +\infty, & \text{otherwise.} \end{cases}$$

(3) Objective:

$\min\{G_n(g_1, g_2, \ldots, g_{m+1}, D) \mid 1 \leqslant i \leqslant m+1, 0 \leqslant g_i \leqslant L_i', a_{2m} \leqslant D \leqslant a_{2m+1} + q_{\max}$ $\Sigma(i = 1, 2, \ldots, m+1) g_i \leqslant P\}$.

Consider the complexity of DP_1. Note that we assume $L'_i < P$ for $i = 1, 2, \ldots, m$. As a result, the recursive function has at most $O(nP^{m+1}(P + q_{\max}))$ states for $G_j(g_1, g_2, \ldots, g_{m+1}, D)$, and each recursion can be calculated in $O(m+1) = O(1)$ time. Therefore, DP_1 has a complexity of $O(nP^{m+2} + nP^{m+1}q_{\max})$. We then have the following theorem.

Theorem 1.1 Problem $1 \mid \mathrm{FB}(m), \mathrm{rej}, q_j \mid D_{\max} + \Sigma w_j$ can be solved in a pseudo-polynomial time.

1.3 Non-approximability Results

In this section, we will provide some negative results of the development of approximation algorithms for problem $1 \mid \mathrm{FB}, \mathrm{rej}, q_j \mid D_{\max} + \Sigma w_j$. As what we will see it later, when there are two or more UIs, then no approximation exists even for some very special cases.

We first study the approximability of problem $1 \mid \mathrm{FB}(2), \mathrm{rej} \mid C_{\max} + \Sigma w_j$. In this problem, there are only two UIs, and the delivery time of each job is equal to zero. As what we will show later, the problem $1 \mid \mathrm{FB}(2), \mathrm{rej} \mid C_{\max} + \Sigma w_j$ does not admit a polynomial time approximation algorithm with a constant worst-case bound. As we will use the well-known NP-hard problem PARTITION in our analysis, for convenience, we state PARTITION in advance here (see Garey and Johnson, 1979): Given a set of t positive integers $S = \{e_1, e_2, \ldots, e_t\}$ such that $\sum_{i=1}^{t} e_i = 2B$. The question is to decide if there is a subset $S' \subseteq S$ such that $\sum_{e_i \in S'} e_i = \sum_{e_i \in S \setminus S'} e_i = B$.

Theorem 1.2 Problem $1 \mid \mathrm{FB}(2), \mathrm{rej} \mid C_{\max} + \Sigma w_j$ does not admit a polynomial time approximation algorithm with worst-case bound better than 2^n unless $P = NP$.

Proof: We use the gap reduction from PARTITION. Given an instance I of PARTITION, construct an instance I' of $1 \mid \mathrm{FB}(2), \mathrm{rej} \mid C_{\max} + \Sigma w_j$ as follows. There are $n =$

t jobs, where the processing time and rejection penalty of job J_j are $(p_j, w_j) = (e_j, 2^n(2B+1))$ for $j = 1, \dots, t$. There are $m = 2$ UIs with $(a_1, a_2) = (B, B+1)$ and $(a_3, a_4) = (2B+1, 2^n(2B+1))$. Such an instance can be constructed in polynomial time considering the optimal solution value of instance I'. It is obvious that the optimal solution value of I' is no less than $2B+1$. If we reject one job, the total penalty is no less than $2^n(2B+1)$. Also, if there exists a job processed after time $2^n(2B+1)$, the corresponding makespan also will be no less than $2^n(2B+1)$. Thus, to obtain a solution value of $2B+1$ is equivalent to find a subset of jobs with a total processing time of B, so that all of the jobs in the subset are processed on the machine within interval $[0, B]$ without any idle time, while all of the rest of the other jobs are processed on the machine within interval $[B+1, 2B+1]$ also without any idle time. This is equivalent to a solution to PARTITION. If it fails to find out such a subset of jobs, then solution value is at least $2^n(2B+1)$, which is 2^n times of $2B+1$. This completes the proof. ■

Through the proof above, we can see that if a_{2m}, the ending point of the last UI, is much larger than $P(U) = \sum_{J_j \in U} p_j$, the total processing time of the jobs in $U = \{J_j \in J \mid w_j > p_j\}$, then it is impossible to develop a polynomial approximation algorithm to problem $1 \mid FB(2), rej \mid C_{\max} + \sum w_j$ with a constant worst-case bound. However, as we will show it later, if $P(U) > a_{2m}$, i. e., the total processing time in U is greater than the ending point of the last UI, then a very simple heuristic with a worst-case bound of 2 can be developed to solve problem $1 \mid FB(2), rej \mid C_{\max} + \sum w_j$. The heuristic is denoted by H_1, which is described as follows: Reject all jobs in V, and let all jobs in U be accepted and processed within time interval $[a_{2m}, +\infty)$ as early as possible in any job sequence. Let Z_{H_1} be the objective function value of the solution generated by heuristic H_1. We have the following theorem.

Theorem 1. 3 H_1 is a 2-approximation algorithm to problem $1 \mid FB(2), rej \mid C_{\max} + \sum w_j$ when $P(U) > a_{2m}$.

Proof: Note that we have $Z^* > a_{2m}$ if $P(U) > a_{2m}$. Remember that A^* and R^* are the set of all accepted jobs and the set of all rejected jobs in the optimal solution, respectively. We then have