May 2013

Heuristic Algorithms to Minimize Total Weighted Tardiness on the Single Machine and Identical Parallel Machines with Sequence Dependent Setup and Future Ready Time

Yue Xi
University of Wisconsin-Milwaukee

HEURISTIC ALGORITHMS TO MINIMIZE TOTAL WEIGHTED TARDINESS ON THE SINGLE MACHINE AND IDENTICAL PARALLEL MACHINES WITH SEQUENCE DEPENDENT SETUP AND FUTURE READY TIME

by

Yue Xi

A Dissertation Submitted in
Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy

in

Engineering

at

The University of Wisconsin-Milwaukee

May, 2013

ABSTRACT
HEURISTIC ALGORITHMS TO MINIMIZE TOTAL WEIGHTED TARDINESS ON THE SINGLE MACHINE AND IDENTICAL PARALLEL MACHINES WITH SEQUENCE DEPENDENT SETUP AND FUTURE READY TIME

by

Yue Xi

The University of Wisconsin-Milwaukee, 2013
Under the Supervision of Professor Jaejin Jang

This study generates heuristic algorithms to minimize the total weighted tardiness on the single machine and identical parallel machines with sequence dependent setup and future ready time. Due to the complexity of the considered problem, we propose two new Apparent Tardiness Cost based (ATC-based) rules. The performances of these two rules are evaluated on the single machine and identical parallel machines. Besides of these two rules, we also propose a look-ahead identical parallel machines heuristic (LAIPM). When a machine becomes idle, it selects a job to process from available jobs and near future jobs.

For the considered combination of scaling parameters, the proposed look-ahead heuristic is divided into three phases: in the first phases, we use the newly introduced dispatching rule, apparent tardiness cost with ready time and continuous setup (ATCRCS), to select the initial job for each machine. The second phase, composed of several iterations, schedules all rest jobs on machines. Each iteration starts identifying the critical machine (the machine with the smallest finish time) and its next job (the critical job). The look-ahead thresh for other machines (non-critical machines) equals to the sum

of the finish time of the critical job and the average setup time. The next job on the considered non-critical machine is chosen from jobs whose ready time is smaller or equal to the look-ahead thresh. Once all machines finish considering their next job selection, a possible iteration schedule is generated. The selected jobs are then used as inputs of *the job switching heuristic* which allows the selected jobs to be switched among machines and evaluated at different positions. Job switching heuristic generates another possible iteration schedule and compares it to the previously generated possible iteration schedule to determine the schedule of the considered iteration. After all jobs are scheduled on machines, the last phase uses a technique called pairwise exchange to further reduce the total weighted tardiness on each machine. Pairwise exchange technique orderly switches two jobs' position and selects the schedule with the smallest total weighted tardiness as the schedule for the considered combination of scaling parameters. The final schedule of the considered problem is the one with the smallest total weighted tardiness among the schedules generated by different scaling parameters combinations.

Different from other look-ahead heuristics, such as the look-ahead heuristic of Mao *et a*l. (1994) and Chang *et al*. (2004), the proposed look-ahead heuristic not only looks ahead (considers limited number of future jobs) but also looks back (schedules each selected job before the last job on each machine). To evaluate its performance, the proposed look-ahead heuristic is compared with available look-ahead heuristics and non look-ahead heuristic on 5103 randomly generated problems in minimizing the total weighted tardiness.

To my advisor, parents and family, who made all of this possible,
and for their endless help and encouragement

ACKNOWLEDGEMENTS

There are a number of people to whom I am deeply indebted and would like to acknowledge their contributions toward to this dissertation.

I would like to first give my utmost gratitude to my advisor, Dr. Jaejin Jang, whose enormous and endless efforts have enhanced my ability and personality. During my Ph.D. study, Dr. Jaejin Jang not only passed me knowledge but also instruct me how to think and solve problems in my research. Doing research with him, I learnt a lot things that I cannot get from the classroom learning and textbook. His nicely teaching style and dedicated research altitude deeply influence my study and research. I really admire his dedication and intellect.

I also would like to present my wholehearted appreciation to my committee numbers for their valuable time and instructions in my dissertation and study: Dr. Matthew Petering and Dr. Xiaohang Yue instructed me on the topic of system simulation. Dr. Hamid Seifoddini is an excellent professor and always helpful in the area of lean manufacturing and grouping technology. I would like to thank Dr. Xiang Fang for taking time out of her busy schedule to be one of my committee members. Finally, I deeply appreciate for the financial support from the department and our department chair, Dr. Aurn Garg. Thanks to put me in teaching.

Finally, I would like to express my boundless gratefulness to my parents, who have gave me endless support in my life, and to my grandmother, Xiuyun Jia, who had kept

taking care of me. I also thank for my wife, for her warm supports, especially for bringing me the best gift in my life: Meina Xi, our daughter.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

HEURISTIC ALGORITHMS TO MINIMIZE TOTAL WEIGHTED TARDINESS ON THE SINGLE MACHINE AND IDENTICAL PARALLEL MACHINES WITH SEQUENCE DEPENDENT SETUP AND FUTURE READY TIME

1. INTRODUCTION

The configurations of the single machine and parallel machines are very common in both service system and production system. The task of the single machine scheduling is to determine the processing sequence of a series of jobs. While, in the parallel machines production, every machine has the same work function and every job can be processed by any machine. Parallel machines scheduling (PMS) mostly considers single operation jobs. The task of PMS is to decide each job's starting time and the machine to process it, so that a certain objective is achieved.

Allahverdi and Mittenthal (1994) group the parallel machines into three cases: *identical parallel machines*, where the processing time of a job is the same on all machines; *uniform parallel machines*, where the processing time of a job is determined by the speed factor of the machine; and *unrelated parallel machines*, where the processing time of a job on different machines can be different in an arbitral way.

Most research on the single machine scheduling and parallel machines scheduling assumes setup time can be either ignored or included in a part's processing time. This assumption is reasonable only when the setup time is independent from the job sequence. Sequence dependent setup, in which the length of setup time of a job depends on its immediately preceding job, is common and often important in production. Examples of

sequence dependent setup are found in petroleum production plants, printing plants, car spraying facilities, metallurgical industries and textile dying plants (Luo *et al.* 2006 and Arroyo *et al.* 2009).

Based on Panwalkar *et al.* (1973), about 70% of schedulers reported that a quarter of jobs they scheduled cannot ignore setup time. Krajewski *et al.* (1987) claim that effective management of sequence dependent setup is one of the critical factors to improve the performance of a manufacturing system.

The performance measures of the single machine scheduling and parallel machines scheduling are mostly either flow time related or tardiness related. The flow time related measures are closely related to job's waiting time for processing and inventory level in a shop; while the tardiness related criteria are relevant to penalties if the manufacturer can not meet predefined due dates. Not meeting with due dates may result in losing future customers. To measure the quality of a schedule from "tardiness" perspectives, several criteria have been used in literatures, such as minimizing the total (weighted) tardiness, minimizing the sum of (weighted) earliness and (weighted) tardiness, minimizing the number of tardy jobs and so on.

Aside from the concept of the sequence dependent setup, there is another "dependent" concept called "machine dependency" in the parallel machines scheduling. Machine dependency means that processing a job needs different setup times and different processing times due to different machine conditions. This concept is different

from unrelated parallel machines in which only the length of the processing time is decided by the condition of a machine in the arbitral way. This study explores heuristic algorithms to minimize the total weighted tardiness on the single machine and identical parallel machines with the sequence dependent setup and future ready time.

In this study, we first focus on the single machine scheduling. Two efficient apparent tardiness cost-based (ATC-based) rules are proposed. The performances of these two newly introduced dispatching rules are then evaluated on the single machine and identical parallel machines with other rules. Finally, we propose a look-ahead identical parallel machines (LAIPM) heuristic. All proposed heuristics mentioned in this research are all focused on minimizing the total weighted tardiness.

This study is organized as follows: Section 2 reviews literature related to the single machine scheduling and parallel machines scheduling with sequence dependent setup. The look-ahead control heuristics are surveyed at the end of section 2. Section 3 relates to the single machine scheduling. Two new ATC-based rules are introduced to minimize the total weighted tardiness. Section 4 is an experiment study carried out on the identical parallel machines. We evaluate performances of the proposed new ATC-based rules and other rules on the identical parallel machines. Section 5 proposes a look-ahead heuristic, LAIPM, which is also for the identical parallel machines scheduling. Finally, conclusions and future research are discussed in section 6.

2. LITERATURE REVIEW

This section reviews the single machine scheduling and parallel machines scheduling with sequence dependent setup. Look-ahead control heuristics are summarized at the end of this section.

2.1 Single machine scheduling with sequence dependent setup

This section reviews the single machine scheduling with sequence dependent setup to minimize tardiness-related criteria. Literature on this topic is grouped into three by the objective: minimizing total tardiness, minimizing the sum of total earliness and tardiness, and minimizing other types of tardiness-related criteria.

2.1.1 Minimizing total tardiness

Pinedo (2002) remarks that minimizing total tardiness with sequence dependent setup ($1/s_{ij}/\sum T_j$) is strongly NP hard. Incorporating job weight or future ready time into $1/s_{ij}/\sum T_j$ makes the problem more difficult. Due to this challenge, many people try to obtain a near optimal solution by heuristic approaches. Liao *et a*l. (2012) divide the scheduling heuristic algorithms into two categories: the constructive approach and the improvement approach. These two approaches are also called construction method and interchange method by Wodecki (2008), respectively.

The constructive approach uses dispatching rules to build a schedule by fixing a job in a position one by one. Several apparent tardiness cost based rules are proposed to minimize the weighted tardiness without considering sequence dependent setup (Morton

and Rachamadugu 1982, Morton and Pentico 1983, and Lee and Pinedo 1997). To consider the sequence dependent setup, Raman *et al*. (1989) propose a modified ATC rule which considers the setup time in both the WSPT term and the slack term. Lee *et al*. (1997) propose the apparent tardiness cost with setups algorithm (ATCS). ATCS is reported as the best constructive algorithm for the $1/s_{ij}/\sum w_j T_j$ by Liao *et al.* (2012). For the batch production where several jobs can be processed together at the same time by a machine, Mason *et al*. (2002) propose batch apparent tardiness cost with setups (BATCS). At the decision time, jobs with the largest index values are selected together to form a batch. In their later research, Mason *et al*. (2005) propose another ATC-based rule which considers the utilization of the batch machine in the index. Pfund *et al*. (2008) modify the slack term of BATCS and propose BATCSmod. Table 1 summarizes above mentioned ATC-based rules on the single machine scheduling.

Table 1. ATC-based dispatching rules

Authors	Rule name	Ready time	Number of parameters	Environment
Rachamadugu and Morton (1982)	ATC	No	1 (k)	Single machine
Raman et al. (1989)	Modified ATC	No	1 (k)	Single machine
Lee *et al.* (1997) Lee and Pinedo (1997)	ATCS	No	2 (k_1 and k_2)	Single/Parallel machines
Vepsalainen and Morton (1987)	MATC	No	2 (b and k)	Flow shop , Job shop
Morton and Pentico (1993)	X-RM	Yes	2 (B and k)	Single machine
Mason *et al.* (2002)	BATCS	Yes	2 (k_1 and k_2)	Batch machine
Mason *et al.* (2005)	ATC-based	Yes	2 (k_1 and k_2)	Batch machine
Pfund *et al.* (2008)	BATCSmod	Yes	2 (k_1 and k_2)	Batch machine

The improvement approach or the interchange method starts with an initial solution and repeatedly strives to improve the current solution by local interchange. To minimize total tardiness with the consideration of sequence dependent setup, both Tan and Narasimhan (1997) and Lin and Ying (2008) suggest the simulated annealing method. Genetic algorithms are proposed by Tan *et al*. (2000), Franca *et al*. (2001), and Sioud *et al*. (2010). Gupta and Smith (2006) propose the greedy randomize adaptive search procedure (GRASP) and the problem space-based local search heuristic. To minimize the total weighted tardiness with the consideration of sequence dependent setup, Cicirello and Smith (2005) analyze the effectiveness of stochastic sampling approaches, such as value-biased stochastic sampling (VBSS), VBSS with hill-climbing, a limited discrepancy search, and heuristic-biased stochastic sampling, together with the simulated annealing. For the same problem, Liao and Juan (2007) and Anghinolfi and Paolucci

(2008) use ant colony optimization to get an improved solution. In an experiment study, Lin and Ying (2007) compare the performance of three popular meta-heuristics: genetic algorithm, simulated annealing, and tabu search. Lin and Ying (2008) also propose a simulated annealing-tabu procedure and test its performance for both total tardiness and total weighted tardiness problems. Kirlik and Oguz (2012) present the variable neighborhood search (NBV) to get a near optimal solution. They (Kirkik *et al*. (2012) later introduce a genetic algorithm that uses a newly proposed crossover operator.

The mathematical modeling approach is used to get an optimal solution. Kirlik and Oguz (2012) and Kirkik *et al*. (2012) present two models to minimize the total weighted tardiness on a single machine with sequence dependent setup. Compared to the mathematical model Kirlik and Oguz (2012), their later model (Kirlik *et al*. 2012) uses fewer variables.

2.1.2 Minimizing the sum of total earliness and tardiness

Without considering job weight, Hepdogan *et al*. (2009) solve an earliness and tardiness problem by a heuristic called meta-heuristic for randomized priority search (Meta-Raps). Rabadi *et al*. (2004) present an optimal branch-and-bound algorithm for the problem with sequence dependent setup. To minimize the sum of weighted earliness and weighted tardiness, Azizoglu and Webster (1997) propose a branch-and-bound algorithm and a beam search procedure for the problem with sequence independent setup times and an unrestricted common due date. Genetic algorithms and tabu search are used by Webster *et al*. (1998) and Kolahan and Liang (1998), respectively.

2.1.3 Minimizing other types of tardiness-related criteria

To minimize maximum tardiness, Ovacik and Uzsoy (1994) present a rolling horizon procedure (RHP), where the problem is decomposed into a series of smaller problems. Asano and Ohta (1999) propose a branch-and-bound algorithm, which considers both future ready time and machine down time. Nekoiemehr and Moslehi (2011) propose three dominance rules to minimize the sum of maximum earliness and maximum tardiness ($1/s_{ij}/ETmax$) by the branch-and-bound algorithm. Uzsoy *et al*. (1992) and Arroyo *et al*. (2011) also consider tardiness-related measures.

2.2 Parallel machines scheduling with sequence dependent setup

This section reviews the studies of parallel machines scheduling with sequence dependent setup. In literature, research about parallel machines scheduling can be grouped based on different criteria: **Machine type criterion,** where parallel machines are classified into identical machines, uniform machines, and unrelated machines; **Setup time criterion,** where the parallel machines scheduling studies are divided into: studies without setup time and studies with setup time. **Batch criterion,** where related papers categorize them into parallel batch production and parallel non-batch production; **Objective criterion,** where related papers group it by objectives, such as minimizing the total completion time, minimizing the total tardiness, and so on; or **Approach criterion,** where related papers group it by approaches, such as branch and bound, meta-heuristic, mathematical model, constructive heuristic and so on.

In this study, we use **batch criterion** to group parallel machines scheduling with sequence dependent setup time into *parallel non-batch machines scheduling with sequence dependent setup* and *parallel batch machines scheduling with sequence dependent setup.*

2.2.1 Parallel non-batch machines scheduling with sequence dependent setup

This section describes the sequence dependent setup literatures in the parallel non-batch machines scheduling. Articles are further grouped by the objective type.

(I) Minimizing total (weighted) tardiness

The constructive approach or the construction method builds a schedule by fixing a job in a position one by one. Apparent Tardiness Cost with Setups, ATCS, is an effect ATC-based dispatching rules to minimize the total weighted tardiness considering sequence dependent setup. Its effectiveness is proved on the parallel machines (Lee and Pinedo. 1997). Extending from ATCS and considering future ready time of a job, Pfund *et al*. (2008) propose ATCSR rule which outperforms other rules, such as EDD, WEDD, ATCS, BATCSmod (Pfund *et al*. 2008), and X-Rmod (Pfund *et al*. 2008).

When applying ATC-based rules, choosing or determining good scaling parameters is also important in literature. At least two types of approaches have been used to decide the scaling parameters' values. One is estimating one good grid, like the empirical value method (Rachamadugu and Morton 1982, Vepsalainen and Morton 1987), the regression method (Lee *et al.* 1997, Pfund *et al.* 2008), and the artificial neural network method

(Kim *et al*. 1995, Park *et al*. 2000). The other tries many different grids and selects the best combination of scaling parameters, like the grid method (Pfund *et al*. 2008, Drieel and Monch 2009, 2011). The grid method is very useful in that it is not only used to determine a final schedule but also to provide input information for other methods, like the regression method and other heuristics that get improved solution (Christoph *et al.* 2007, Drieel and Monch 2009, 2011). Especially, a few grid settings (the range of scaling parameters and the size of the gap between grids) are proposed in literature. Lee *et al*. (1997) present a grid setting: k_1 = (0.2, 0.4, 0.6,..., 6.4), and k_2 = (0.1, 0.2, 0.3,..., 1.6). To ensure that "the frequently occurring values were not the extreme values in the grid." Pfund *et al*. (2008) propose a wider search range for k_1 and k_2, they also proposing settings for their newly introduced parameter, k_3:

k_1: 0.2,0.6,0.8,1,1.2,1.4,1.6,1.8,2,2.4,2.8,3.2,3.6,4,4.4,4.8,5.2,5.6,6,6.4,6.8,7.2

k_2: 0.1,0.3,0.5,0.7,0.9,1.1,1.3,1.5,1.7,1.9,2.1

k_3: 0.001,0.0025,0.004,0.005,0.025,0.04,0.05,0.25,0.4,0.6,0.8,1,1.2

Following Pfund's setting, Drieel and Monch (2009) propose narrower search ranges: [0.2, 6], [0.1, 1.9] and [0.001, 1.2] and consider fewer grids: 7, 4 and 5 grids for k_1, k_2 and k_3, respectively. In their latest research (Driessel and Monch 2011), they use an even more coarse grid setting for k_3 whose search range is set in [0.001, 1] and only four grids are considered in it. The grid settings (for three scaling parameters) in literature are shown in table 2. In the research of Drieel and Monch (2009, 2011), the grid method generates an initial solution for their proposed variable neighborhood search procedures. In this study,

we used the setting of Pfund *et al*. (2008) to evaluate the proposed ATC-based rules and other ATC-based rules including ATCSR. For the proposed look-ahead heuristic (LAIPM), we uses the setting as that of Drieel and Monch (2009), because their setting has a medium number of grids and searching in a similar range as Pfund *et al*. (2008). Also using their setting avoids subjective analysis.

Table 2. Grids setting comparison

	Pfund *et al*. (2008)	Driessel and Monch (2009)	Driessel and Monch (2011)
k_1	22 values in [0.2, 7.2]	7 values in [0.2, 6]	5 values in [0.01, 1.5]
k_2	11 values in [0.1, 2.1]	4 values in [0.1, 0.9]	4 values in [0.1, 1.9]
k_3	13 values in [0.001, 1.2]	5 values in [0.001, 1.2]	4 values in [0.01, 1.0]

The improvement approach or *the interchange method* starts with an initial solution and repeatedly strives to improve the current solution by local interchange. Fowler and Horng (2003) present a hybrid genetic algorithm to minimize total weighted tardiness on identical parallel machines. Tamimi and Rajan (1997) propose a genetic algorithm for uniform parallel machines scheduling with sequence dependent setup. Different from Flowler and Horng (2003), they dynamically modify mutation rate, crossover rate and insertion rate. Chen (2009) proposed a hybrid method for unrelated parallel machines. Their experiments show that simulated annealing effectively improve the initial solution that obtained by ATCS.

(II) Minimizing total (weighted) completion time

Felipe (2005) presents a constructive heuristic to assign jobs iteratively with the minimum adjusted processing time (sum of setup time and process time). Their approach uses an improved enumeration to assign jobs either at the beginning or after a partial job sequence generated on a machine. Kurz and Askin (2001) present an integer programming model to minimize total completion time for identical parallel machines. In their heuristic, once jobs are assigned to the machines, a traveling sales man problem is formulated to find an optimal job sequence on each machine. In their research, the distance between each pair of cities correspondents to sequenced dependent setup.

Weng *et al*. (2001) minimize the total weighted completion time on unrelated parallel machines. Several heuristics are presented. By their experiment, the best heuristic assigns one job at a time based on the ratio of a job's processing time plus setup time to its weight. Fowler and Horng (2003) propose a hybrid genetic algorithm to minimize total weighted completion time. The algorithm is also tested to minimize the total weighted tardiness. In their hybrid approach, the genetic algorithm assigns jobs to machines; dispatching rules are then used to schedule jobs on each individual machine.

(III) Bi-criteria

Balakrishna *et al*. (1999) minimize the sum of weighted earliness and weighted tardiness on uniform parallel machines. This objective is meaningful for a Just-In-Time (JIT) production where both earliness and tardiness are deemed as low efficiency. A mixed integer programming model is formulated to solve the small size problem. Radhakrishnan and Ventura (2000) also minimize sum of weighted earliness and

weighted tardiness, but for identical parallel machines. They also propose a mathematical model to find optimal solution. For a larger problem, they suggest simulated annealing. For the same problem, Feng and Lau (2005) also suggest meta-heuristic approach due to the complexity of the solved problem. Their proposed heuristic outperforms that of Radhakrishnan and Ventura (2000).

Heady and Zhu (1998) present a heuristic method to minimize the sum of earliness and tardiness for identical parallel machines with sequence dependent setup time. They don't consider job weight. For a small problem, they compare the performance of the heuristic with the optimal solution from integer programming.

(IV) Other criteria

To maximum machine utilization, Christos and Milton (1988) propose a heuristic to minimize machine interference. Hirashi *et al*. (2002) address identical parallel machines scheduling with sequence dependent setup time. They maximize the weighted number of jobs that are completed before their due dates. Kim *et al*. (2002) propose a restricted tabu search to reduce search effort significantly without eliminating the promising solutions. They minimize the maximum lateness on identical parallel machines by considering sequence dependent setups. Anglian *et al*. (2005) minimize total setup time for identical parallel machines. They use fuzzy mathematical programming.

Table 3 shows research related to the parallel non-batch machines scheduling with sequence dependent setup time

Table 3. Parallel non-batch machines with sequence dependent setup time

Objective functions	References	Machines type	Approach	Others
1. Total weighted tardiness	Lee et al (1997)	Identical parallel machines	Dispatching rules (ATCS)	
	Lee et al (1997)	Identical parallel machines	Three stages method	ATCS is used to find the initial solution for simulated anealing
	Park et al (2000)	Identical parallel machines	Dispatching rules (ATCS)	Neural network to imporve ATCS
	Fowler et al (2003)	Identical parallel machines	Hybrid genertic algorithm (GA)	Genetic algorithm assigns jobs to machine, then dispatching rules are used to sequence jobs on each machine
	Tamimi et al (1997)	Uniform parallel machines	Genetic Algorithm	Dynamic crossover rate
2. Total tardiness	Chen et al (2006)	Unrelated parallel machines	Hybride Aproach (ATCS+SA)	Simulated annealing , Lee's ATCS find initial solution
3. The total completion time	Felipe et al (2005)	Unrelated parallel machines	Constructive method	An improved emueration
	Kurz et al (2001)	Identical parallel machines	Integer programming	Some jobs's release time $\neq 0$
4. Total weighted completion time	Weng et al (2001)	Unrelated parallel machines	Evaluate seveal hueristics	Comparisive experiments
	Fowler et al (2003)	Identical parallel machines	Hybrid genertic algorithm (GA)	A two stages method

5. Bi-Creteria				
Sum of weighted earliness and weighted tardiness	Balakrishna et al (1999)	Uniform parallel machines	Mixed integer programming	Bender's decomposition procesure (for large size problem)
	Radhakrishnan et al (2000)	Identical parallel machines	Mathematical programming	
	Feng et al (2005)	Identical parallel machines	Meta-huristic	Outperforms Radhakrishnan et al's work (2000)
Sum of earliness and tardiness	Heady et al	Identical parallel machines	Hueristic	Solution compared with that from the integer programming
6. Others				
Maximize the weighted number of jobs that are completed at their due date	Hirashi et al (2002)	Identical parallel machines		A maximum objective function
Minimize the maximum lateness	Kim et al (2003)	Identical parallel machines	Restricted tabu search	Reduce computation effort without losing promising solutions
Minimize mean completion time	Michael et al (2001)	Identical parallel machines	Hybrid genertic algorithm (GA)	A two stages method

Minimize total setup time	Anglian et al (2005)	Identical parallel machines	Fuzzy	programming

2.2.2. Parallel batch machines scheduling with sequenced dependent setup

Most works in this category are tardiness related. Both Karp (1972) and Ho and Chang (1995) claim that minimizing total tardiness of two identical machines (non-batch machine) even without setup is NP hard. Due to the complexity of the problem, people tend to find quality solutions but rather an optimal solution.

Meta-heuristics are popular methods to find near optimal solutions in complicated scheduling environments. Because they impose severe computation burden compared with convenience dispatching rules, meta-heuristics may not be suitable when quick solutions are needed within a short time. Kim *et al*. (2002) address unrelated parallel batch machines scheduling with sequence dependent setup time. In their study, the jobs within in a family have the same due date. Simulated annealing that utilizes job rearrangement techniques is used to generate neighborhood solutions. Kim *et al*. (2003) test four heuristics for unrelated parallel batch machines: (1) the earliest weighted due date, (2) the shortest weighted process time, (3) the two level batch scheduling heuristic, and (4) the simulated annealing method. Their test shows that simulated annealing outperforms other heuristics to minimize total weighted tardiness. For the same problem, Eom *et al*. (2002) propose a three stages method where the last phase is tabu search.

To find the optimal solution, Chen and Powell (2003) propose a branch and bound algorithm to minimize total weighted completion time on identical parallel batch machines. Computational analysis shows that it is capable to optimally solve medium size problems within reasonable computation time.

Table 4 shows the research related to the parallel batch machines scheduling with sequence dependent setup time

Table 4. Parallel batch machines with sequence dependent setup time

Objective funtions	**References**	**Machines type**	**Approach**	**Others**
1. Meta-heuristic				
Minimize the total tardiness	Kim et al (2002)	Unrelated parallel batch machine	Simulated Anealing	
Minimize the total weighted atrdiness	Eco et al (2002)	Indentical parallel batch machine	A three stages method.	ATCS+ Simulated Anealing
Minimize the total weighted atrdiness	Kim et al (2003)	Unrelated parallel batch machine	Test four heuristics	Simulated anealing outperforms than others.
2. Others				
Minimize total weighted completion time	Chen et al (2003)	Identical parallel batch machine	Brach and bound	Solve medium size problem optimally

2.3 Look-ahead control procedures

This section reviews look-ahead control heuristics of the machine scheduling. Related heuristics are grouped into Look-ahead non-batching heuristics and Look-ahead batching heuristics.

2.3.1 Look-ahead non-batching heuristics

Christos and Milton (1988) reduce interference for one operator who operates parallel machines. The schedule made by their heuristic yields high machine utilization and high operator utilization simultaneously. Mao *et al.* (1994) explore a one step

look-ahead heuristic to improve the performance of on-line heuristics. They minimize the total completion time and study the performance of the worst case. Jihene *et al.* (2002) look ahead machine preventive maintenance and compare four single machine heuristics to minimize total tardiness. Jang *et al.* (2001) propose a heuristic to minimize flow time or tardiness on the parallel machine. Each part has different processing times on different machines and there is no local buffer. Once part arrived, the destination machine is decided for the operation at once. Different from other look-ahead heuristic, the considered part is selected from the machine perspective. The assigned parts on the machine are processed by first in first out (FIFO) rule. Chang *et al.* (2004) propose a one step look-ahead heuristic which targets to minimize the total weighted tardiness with sequence dependent setup and unequal ready time. They iteratively select a job to the machine, so that the created partial schedule yields the smallest incensement of the total weighted tardiness. Once all jobs are scheduled, pairwise exchange is used to further reduce the total weighted tardiness. Their heuristic is proved as an efficient method when problem size is small.

In the more complicated scheduling environments, such as flow shop and job shop, Smith *et al.* (1996) explore the influence of changing part input sequence, part mix ratio and look-ahead strategy to the machine utilization in a flexible flow shop. The tested look-ahead strategy is a one-step look-ahead that guarantees that the machine used for the next operation is the one with the earliest available time. Experiment shows that the theoretical maximum utilization can be achieved with lower WIP level when balanced part mix ratio incorporates look-ahead strategy. Ginzburg *et al*. (1997) propose a heuristic

that combines pairwise comparison with the look-ahead concept to select the next job for the idle machine. Competitions are carried out among available jobs. The look-ahead horizon is decided by the processing time of the winner. The future arrivals, whose estimated finish time falls in this horizon, are considered. The selected available job is then competed with considered future jobs. If the winner is still the available job, dispatch the job now, otherwise, wait for the winning future arrival. Other look-ahead heuristics see works of Holthaus and Ziegler. (1997) and Tunali (1997).

2.3.2 Look-ahead batching heuristics

In the batching production where several parts are produced together by one time, full batch always has higher priority to process than partial batch. When a full batch can not be formed, scholars generate look-ahead heuristics to decide the batch should be processed now or delayed to a future arrival.

To minimize flow time, Glassey and Weng (1991) propose dynamic batch heuristic (DBH) heuristic. DBH computes a net value for each considerable future arrival during one processing time of the batch machine from the decision time. The next batch's loading time is the future arrival with the largest net value. Fowler *et al.* (1992) propose next arrival cost heuristic (NACH) heuristic which only considers the next future arrival. Other look-ahead heuristics for the same problem see, Guilher *et al*. (2000), Fowler *et al*. (2002), Cigoloni *et al.* (2002), and Gupta *et al*. (2004).

To minimize tardiness, Gupta and Sivakumar (2006) propose a look-ahead heuristic for Just-In-Time production. They look ahead one processing time of the batch machine from the decision time. Two earliness and tardiness measures are considered: the mean of the absolute sum of the earliness and tardiness, and the mean of their squared sum. The scenario with smallest root mean square value of earliness and tardiness decides the loading time of the next batch. In their later work (Gupta and Sivakumar. 2007), each lot's slack time is considered (Slack time = d_i-p_i-t_0), where t_0 is the current time. The best scenario is defined as the batch with the minimum value of $\left(\frac{1}{n}\sum_{i=1}^{n}(\max(0, d_i - p_i - t_0))^2\right)^{1/2}$, where n is the number of lots in the scenario.

Weng and Leachman. (1993) propose minimum cost rate (MCR) heuristic. MCR considers max $\{0, k\text{-}q_0\}$ future arrivals at the decision time. k and q_0 are the machine capacity and the number of available parts, respectively. Different from DBH and NACH, MCR's look-ahead horizon is not fixed but equals to one process time of the batch machine plus the waiting time of the considered future arrival. MCR calculates a cost ratio which equals to the total holding costs over the length of the look-ahead horizon to make decision. The loading time of the batch is decided by the considered lot with the smallest cost ratio. Other look-ahead heuristics using cost ratio to make decision see, Robinson and Fowler (1995), Van et al. (1997) and Van (2002).

3. NEW ATC-BASED DISPATCHING RULES FOR THE SINGLE MACHINE SCHEDULING

This section first mentions the problem statement and assumptions, and then introduces two new ATC-based dispatching rules. The sequence dependent setup is classified into two categories: continuous sequence dependent setup and separable sequence dependent setup. The former is the conventional type considered by most research, while the latter does not need a job or a part on a machine to setup.

3.1 Problem statement and assumptions

This research considers the single machine scheduling problem, $1|r_j,s_{ij}|\sum W_jT_j$, which is stated as: there are n jobs arriving to the single machines at different times. Each job j has its ready time (r_j), processing time (p_j), due date (d_j), and job weight (w_j). The setup (s_{ij}) of each pair of jobs i and j is sequence dependent. The objective is to minimize the total weighted tardiness of jobs, $\sum_{j=1}^{n} w_jT_j$, where T_j, $\max\{0,C_j-d_j\}$, is the tardiness of job j and C_j is the competition time of job j.

The considered single machine problem assumes the following:

- The job attributes (p_j, d_j, w_j, r_j, and s_{ij}) are known in advance.
- Machines can process at most one job at each time.
- Job preemption is not allowed.
- Interruption such as machine breakdown and order cancellation does not happen.

3.2 The proposed ATC-based dispatching rules

This section gives detailed information of the proposed ATC-based dispatching rules. We first make an analysis of the ATC-based dispatching rules and then generate two new rules, ATCRCS and ATCRSS.

3.2.1 Analysis of the ATC-based dispatching rules

This section analyzes the WSPT term, the slack term, and the ready time term of existing ATC-based rules. The basic format of ATC-based indexes is a product of several terms:

Index = Term A x Term B x (Term C) x (Term D) (1)

It is noticed that the index of ATC-based rules has, in literature, at least two terms or at most four terms. To select a job to process next on the considered machine, ATC-based rules compute the index value for each unprocessed job and select the job with the largest index value to process.

The index of ATCSR (Pfund *et al*. 2008) is given as an example of the related ATC-based rules:

$$I_{ATCSR}(t,i,j)=\frac{w_j}{p_j}\exp\left(-\frac{\max(d_j-p_j-\max(r_j,t),0)}{k_1\bar{p}}\right)\exp\left(-\frac{s_{ij}}{k_2\bar{s}}\right)\exp\left(-\frac{\max(r_j-t,0)}{k_3\bar{p}}\right) \quad (2)$$

where k_1, k_2, and k_3 are scaling parameters. The four terms in formula (2) are the WSPT term, the slack term, the setup term and the ready time term, respectively, from left to right.

(I) The WPST term

The denominator of the conventional WSPT term is p_j, which signifies the earliest possible completion time of job j from the current time (or the time length for the considered job to use the machine exclusively from the current time). If setup is sequence independent, the processing time can include the setup time. If setup is sequence dependent, and a job is available at time zero, its earliest job completion time is $p_j + s_{ij}$; Raman *et al*. (1997) propose $\frac{w_j}{p_j + s_{ij}}$ as the WSPT term. In their WSPT term, the sequence dependent setup, s_{ij}, is treated as a part of the processing time. On the other hand, if a job has future ready times and sequence dependent setup, its earliest possible completion time is $p_j + s_{ij} + \max(r_j - t, 0)$ for the continuous sequence dependent setup, and is $p_j + \max(s_{ij}, r_j - t)$ for the separable sequence dependent setup. In the above two formulas, $\max(r_j\text{-}t, 0)$ and $\max(s_{ij}, r_j\text{-}t)$ are possible machine idle times for the continuous setup and separable setup, respectively. These possible machine idle times influence the earliest completion times for the considered jobs, and it is reasonable to treat these possible machine idle times as a part of the processing time. Based on this analysis, we propose the following formulas as the new WSPT term:

$$\frac{w_j}{p_j + s_{ij} + \max(r_j - t, 0)} \quad \text{(Continuous sequence dependent setup)} \tag{3}$$

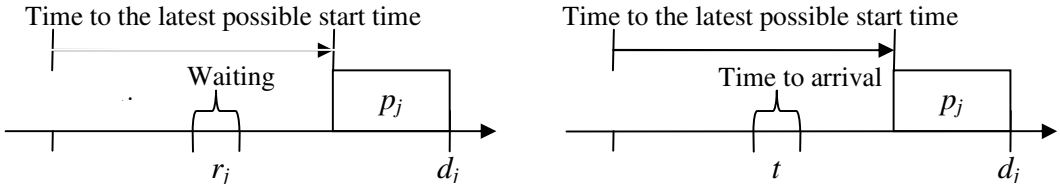
$$\frac{w_j}{p_j + \max(s_{ij}, r_j - t)} \qquad \text{(Separable sequence dependent setup)} \qquad (4)$$

(II) Exponent numerator of the slack term

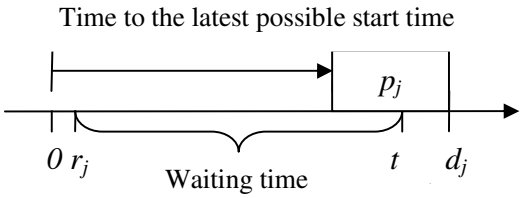
One of the important differences among ATC-based rules is the slack term. This section compares four popular exponent numerators of the slack term and proposes two new formulas. Specifically, both new formulas use the sequence dependent setup time, s_{ij}, which is seldom used in the existing formulas.

(1) $-\max(d_j - p_j + r_j - t, 0)$

This formula in BATCS (Mason *et al.* 2002), $-\max(d_j - p_j + r_j - t, 0)$, assigns a lower priority to a job with a larger latest possible start time, $d_j - p_j$, while meeting the specific due date. It also considers the waiting time of a job, $t - r_j$. If a job has been ready for a longer time, its priority becomes higher, when only consider t and r_j (figure 1(a)). On the other hand, a more future job is assigned a lower priority (figure 1(b)). If a ready job has been waiting longer than the time length (time to the latest possible start time), it gets the highest priority score of the slack term (e^0=1) (figure 1(c)).



(a) Ready job ($r_j \leqq t$) (b) Future job ($r_j > t$)



(c) Late job (d_j-p_j<t) with long waiting time (d_j-p_j<t-r_j)

Figure 1. Slack term of BATCS

(2) $-\max(d_j - p_j + \max(r_j - t),0)$

This formula is used by BATCSmod (Pfund *et al.* 2008). Different from BATCS, all ready jobs ($r_j \leq t$) are assigned the same highest priority when we only consider r_j and t. These ready jobs' priorities are decided by the value of ($d_j - p_j$) regardless of the length of their respective waiting times (figure 2). For a future job ($r_j > t$), this formula is the same as that of BATCS (figure 1(b, c)).

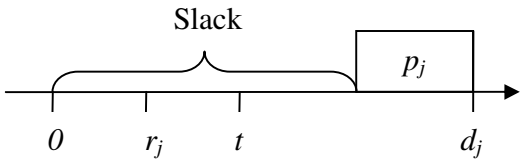
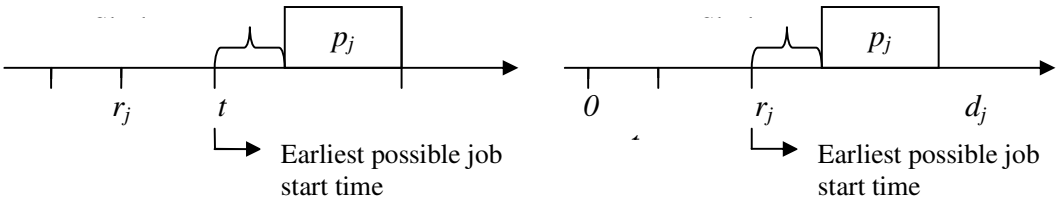


Figure 2. Slack term of BATCSmod: ready job ($r_j \leq t$)

(3) $-\max(d_j - p_j - \max(r_j, t), 0)$

This formula, $-\max(d_j - p_j - \max(r_j, t), 0)$, is used in ATCSR (Pfund *et al*. 2008). It uses the current decision time (t) or the ready time (r_j) to calculate the slacks of a ready job or a future job (figure 3). The slack in ATCSR measures the maximum time length of postponing the start of a job from its earliest possible start time, $\max(r_j, t)$, while still meeting its due date. When this slack is negative, $d_j - p_j - \max(r_j, t) < 0$, the slack term gets the highest priority score (e^0=1). This term does not consider the sequence dependent setup time; even when the slack for this term is positive, the due date may not be met because of the required setup time.



(a) Ready job ($r_j \leqq t$) (b) Future job ($r_j > t$)

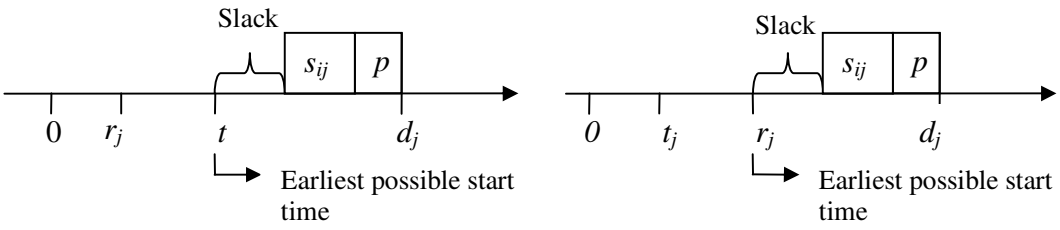
Figure 3. Slack term of ATCSR

(4) $-\max(d_j - p_j - s_{ij} - t, 0)$

If we do not consider the effect of ready time but consider that of the setup time in the formula in 4.2.3 (ATCSR), the formula becomes "$-\max(d_j - p_j - s_{ij} - t)$", which is used by Raman *et al*. (1989).

(5) $-\max(d_j - p_j - s_{ij} - \max(r_j, t), 0)$

We propose a new exponent numerator for the slack term. This new formula includes the sequence dependent setup time in the formula of ATCSR. The slack measures the maximum time for postponing the job start from its earliest possible start time, $\max(r_j, t)$, while still meeting its due date (figure 4).

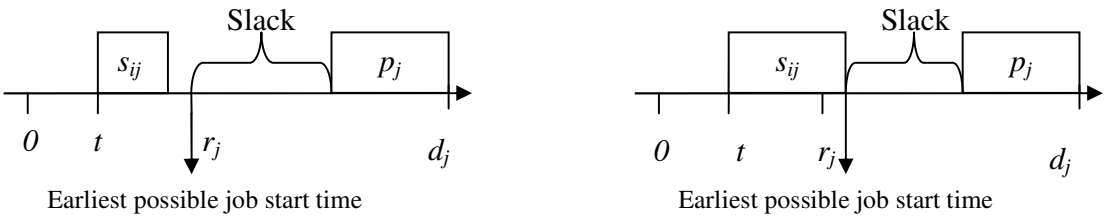


(a) Ready job ($r_j \leqq t$) (b) future job ($r_j > t$)

Figure 4. Slack term of ATCRCS

(6) $-\max(d_j - p_j - \max(r_j, t + s_{ij}), 0)$

In this new formula, the slack is the maximum time in delaying the start of processing a job (but not necessarily the start of setup) from its earliest possible start time while still meeting its due date (figure 5).



(a) setup is done before ready time ($t+s_{ij} \geq r_j$) (b) setup is done after ready time ($t+s_{ij} < r_j$)

Figure 5. Slack term of ATCRSS

(III) Exponent denominator of the slack term

The exponent denominators of the slack terms of existing rules use only the mean processing time ($\overline{p}$) to normalize the numerator; it does not consider the mean setup time ($\overline{s}$), which is important when the setup time is large. This section proposes a new formula as the exponent denominator of the slack term:

$$k_1(\overline{p}+\overline{s}) \tag{5}$$

where k_1 is the scaling parameter for the slack term.

(IV) Exponent denominator of the ready time term

In the exponent denominator of the ready time term, existing ATC-based rules use only the mean processing time ($\overline{p}$) to normalize the numerator of the exponent. This section proposes a new formula as the exponent denominator of the ready time term:

$$k_3(\overline{p}+\overline{s}) \tag{6}$$

where k_3 is the scaling parameter for the ready time term.

3.2.2 The proposed ATC-based rules

Bases on the above analysis, this section proposes new ATC-based rules, ATCRCS and ATCRSS, to minimize the total weighted tardiness with the sequence dependent setup and future time.

(I) Data generation and performance criteria

The test data sets are generated by Pfund's procedure. In the experiment, we use all of their factors except for the job machine factor μ, which is meaningful only for parallel machine problems. For the rest of the five factors, the same levels of each factor are used. The experiment is a 3^5 experiment, which has 243 scenarios. In each scenario, seven problems are randomly generated. A total of 1701 (243 x 7) problems are considered. Each problem has 40 jobs to be scheduled. Table 5 shows the factors and levels of the experiment. Factor and levels in this experiment is given in table 5.

Table 5. Factors and levels of the performance test

Factor	Notation	Factor name	Low level	Center level	High level
1	η	Setup severity factor	0.02	1.01	2
2	τ	Due date tightness factor	0.3	0.6	0.9
3	R	Due date range factor	0.25	0.63	1
4	J_a	Job availability factor	0.2	0.5	0.8
5	r_τ	Ready time factor	1	5.5	10

To compare ATC-based rules, the experiment uses the following two measures: *the best of the best measure* and *the territory measure*.

The best of the best measure

By using a given rule, the grid method generates multiple schedules of each problem (one schedule for each grid) and selects the best one as the final schedule. To evaluate different rules, this test compares the selected best schedules (one from each rule) to determine which rule gives the best solution. We call this measure the best of the best measure.

The territory measure

For a pair of ATC-based rules, the territory measure compares two schedules in a given problem at each grid. It then finds the percentage of grids in which one method performs better than, equal to, or worse than the other method on all the grids. This measure is helpful when choosing a rule that does not have a good procedure for finding a good grid, because the rule with a larger favorable territory is more likely to give a better schedule. On the other hand, the best of the best measure is superior when we have a good procedure to find the best grid for an ATC-based rule.

(II) The proposed new ATC-based rules: ATCRCS and ATCRSS

This section evaluates the effects of the new formulas introduced in section 3.2.1 and proposes two new ATC-based rules for $1|r_j,s_{ij},con|\sum w_jT_j$, and $1|r_j,s_{ij},sep|\sum w_jT_j$, respectively. In the latter part of this section, we compare performances of the two proposed rules with those of ATCSR, one of the best ATC-based rules in literature (ATCSR outperforms other ATC-based rules, such as BATCS, BATCmod, and X-Rmod, to minimize the total weighted tardiness of a problem which considers the sequence dependent setup and future ready time, Pfund *et al.* 2008).

(a) The effect of the modified WSPT term

To evaluate the modified WSPT terms that are introduced in section 3.2.1, $\frac{w_j}{p_j + s_{ij} + \max(r_j - t,0)}$ and $\frac{w_j}{p_j + \max(s_{ij}, r_j - t)}$, we compare the original ATCSR with the modified ATCSR that includes the new WSPT terms. The results are summarized in table 6.

The three numbers in each cell are the number of cases, the percentage of cases, and the average reduction of tardiness, respectively (*e.g.*, the tardiness reduction is 3.7% on average when only the 770 better cases are considered). The table clearly shows the modified WSPT terms improve the performance of ATCSR. In the following section, we will use the modified WSPT terms in further performance tests.

Table 6. Effect of the modified WSPT terms

	Continuous setup ATCSR(new WSPT term) vs. ATCSR	**Separable setup ARCSR(new WSPT term) vs. ATCSR***
Better	770, 45% , 3.7%	785, 46% , 3.8%
Equal	466, 28% , 0%	457, 27%, 0%
Worse	465, 27% , -2.8%	459, 27% , -3.1%

* the setup of a future job starts at the decision time instead of the ready time

(b) The new introduced ATC-based rules, ATCRCS and ATCRSS

Several new formulas are introduced for the slack term and the ready time term in sections 3.2.1. In order to determine the indexes of the new ATC-based rules, we evaluate the effect of these formulas. The considered ATC-based indexes in the experiment have the following format:

$$I(t,i,j) = A\exp(-\frac{B}{C})\exp(-\frac{s_{ij}}{k_2\bar{s}})\exp(-\frac{\max(r_j - t,0)}{D}) \quad (7)$$

where A: modified WSPT term from section 4.2.1.

B: numerator of the exponent of the slack term in section 4.2.2.

C: denominator of the exponent of the slack term in section 4.2.3.

D: denominator of the exponent of the ready time term in section 4.2.4.

Tables 7 and 8 show the results of the best of the best measure and the territory measure for the continuous setup case, respectively. Tables 9 and 10 are for the separable setup case. The three numbers in each cell of tables 7 and 9 are the number of problems in which the corresponding index gives tardiness values that are better than, equal to, or worse than those of the benchmarking index (ATCSR with the modified WSPT term). The three percentage values in each cell of tables 8 and 10 are the average percentage of better, equal, or worse grids in the territory measure of the 1701 problems, when the corresponding index is compared to the benchmarking index (ATCSR with the modified WSPT term).

The formulas of column 3 of tables 7 to 10 are newly proposed in section 3.2.1, while the formulas of column 2, column 4, and column 5 are used in ATCSR, ATC and a modification of ATC (Raman et al. 1997), respectively. Tables 9 and 10 are similar to tables 7 and 8 respectively, except for the separable sequence dependent setup. The rows of the tables consider combinations of the exponent denominator of the slack term and the exponent denominator of the ready time term. The formulas in rows 3, 4, and 5 have newly proposed formulas.

Tables 7 and 10 show the following for the continuous setup cases:

• When comparing the second and fourth rows and the third and fifth rows, both show that the average setup time in the exponent denominator of the slack term (when given in these forms) significantly improves the performance.

• When comparing the second and third rows and the fourth and fifth rows, both show that the average setup time in the exponent denominator of the ready time term (when given in these forms) does not affect the results significantly.

• When comparing the second and fourth columns and the third and fifth columns, both show that the ready time in the exponent numerator of the slack term (when given in these forms) does not make a significant difference to the results.

• When comparing the second and third columns and the fourth and fifth columns, both show that s_{ij} in the exponent numerator of the slack term (when given in these forms) increases the number of better cases (table 7) and the average percentage of grids with tardiness reduction (table 8); however it also increases the number of worse cases and the average percentage of grids with worse solutions for 1701 problems.

Table 9 and table 10 show comparable results to table 7 and table 8 for the separable setup cases. The new rules are proposed based on the above test results, and we select terms based on the performance of the test (the performances are not significantly different) and their simplicity. Because $d_j - p_j - t$ yields both fewer worse cases in the best-of-the-best measure and lower percentage of average worse grids in the territory measure, and also because the formula is simpler than others, we select it as the numerator of the exponent of our new rules. In deciding formulas based on the results of rows, we note the tables show similar performances in the fourth and fifth rows, and we

select the formula in the fourth rows for its simplicity. The resultant formulas are given below:

The index of ATCRCS (for continuous setup):

$$I_{ATCRCS}(t,i,j)=\frac{W_j}{p_j+s_{ij}+\max(r_j-t,0)}\exp(-\frac{\max(d_j-p_j-t,0)}{k_1(\bar{p}+\bar{s})})\exp(-\frac{s_{ij}}{k_2\bar{s}})\exp(-\frac{\max(r_j-t,0)}{k_3\bar{p}}) \quad (8)$$

The index of ATCRSS (for separable setup):

$$I_{ATCRSS}(t,i,j)=\frac{W_j}{p_j+\max(s_{ij},\, r_j-t)}\exp(-\frac{\max(d_j-p_j-t,0)}{k_1(\bar{p}+\bar{s})})\exp(-\frac{s_{ij}}{k_2\bar{s}})\exp(-\frac{\max(r_j-t,0)}{k_3\bar{p}}) \quad (9)$$

Notably, the above choices of the slack term for (8) and (9) could be changed depending on the objective of the scheduling. For example, if we are aggressive, we can maximize the possibility of getting better schedules (at the cost of increased chance of getting worse schedules) by using the formulas of the third or fifth columns and the formula of the fifth row (instead of the fourth row). After deciding the indexes of the new rules, we compare the new ATC-based rules with ATCSR in the following section.

Table 7. The best-of-the-best test (for the continuous sequence dependent setup)

B / C, D	d_j-p_j-max(r_j,t)	d_j-p_j-s_{ij}-max(r_j,t)	d_j-p_j-t	d_j-p_j-s_{ij}-t
$k_1\bar{p}, k_3\bar{p}$	0, 1701, 0	218, 1221, 262	24, 1636, 41	224, 1196, 281
$k_1\bar{p}, k_3(\bar{p}+\bar{s})$	72, 1557, 72	250, 1160, 291	73, 1562, 66	245, 1170, 286
$k_1(\bar{p}+\bar{s}), k_3(\bar{p})$	450, 1139, 112	511, 970, 220	462, 1111, 128	516, 953, 232
$k_1(\bar{p}+\bar{s}), k_3(\bar{p}+\bar{s})$	480, 1077, 144	531, 938, 232	484, 1078, 139	534, 933, 234

Table 8. The territory test (% for the continuous sequence dependent setup)

B, / C, D	d_j-p_j-max(r_j,t)	d_j-p_j-s_{ij}-max(r_j,t)	d_j-p_j-t	d_j-p_j-s_{ij}-t
$k_1\bar{p}, k_3\bar{p}$	0, 100, 0	15.6, 61.5, 22.9	8.7, 89.4, 1.9	21, 56.2, 22.8
$k_1\bar{p}, k_3(\bar{p}+\bar{s})$	5.1, 81.7, 13.2	15.5, 58.8, 28.8	10, 82.8, 7.2	20.7, 53.5, 25.8
$k_1(\bar{p}+\bar{s}), k_3(\bar{p})$	40.3, 43.5, 16.1	40.2, 41.6, 18.2	43.9, 39.9, 16.2	43.5, 38.2, 18.3
$k_1(\bar{p}+\bar{s}), k_3(\bar{p}+\bar{s})$	39.2, 43.1, 17.7	39.3, 41.1, 19.6	43.8, 39.3, 16.9	43.4, 37.7, 19

Table 9. The best-of-the-best test (for the separable sequence dependent setup)

B, / C, D	d_j-p_j-max(r_j,t)	d_j-p_j-max(r_j,t+s_{ij})	d_j-p_j-t	d_j-p_j-s_{ij}-t
$k_1\bar{p}, k_3\bar{p}$	0, 1701, 0	196, 1223, 282	34, 1606, 61	222, 1190, 289
$k_1\bar{p}, k_3(\bar{p}+\bar{s})$	123, 1493, 85	269, 1122, 310	120, 1506, 75	288, 1126, 287
$k_1(\bar{p}+\bar{s}), k_3(\bar{p})$	455, 1133, 113	497, 985, 219	461, 1092, 148	509, 962, 230
$k_1(\bar{p}+\bar{s}), k_3(\bar{p}+\bar{s})$	513, 1039, 149	549, 926, 226	515, 1042, 144	561, 921, 219

Table 10. The territory test (% for the separable sequence dependent setup)

B, / C, D	d_j-p_j-max(r_j,t)	d_j-p_j-max(r_j,t+s_{ij})	d_j-p_j-t	d_j-p_j-s_{ij}-t
$k_1\bar{p}, k_3\bar{p}$	0, 100, 0	15.5, 61.5, 23.1	8.9, 89.9, 2.3	20.8, 56.3, 22.9
$k_1\bar{p}, k_3(\bar{p}+\bar{s})$	6.6, 80.3, 13.2	16.3, 55.2, 28.5	11.3, 81.5, 7.2	21.6, 53, 25.4
$k_1(\bar{p}+\bar{s}), k_3(\bar{p})$	39.9, 44, 16.1	39.7, 42, 18.4	43.3, 40.3, 16.3	42.9, 38.6, 18.5
$k_1(\bar{p}+\bar{s}), k_3(\bar{p}+\bar{s})$	39.4, 43.2, 17.4	39.4, 41.2, 19.4	43.9, 39.5, 16.6	43.5, 37.8, 18.7

(III) The performance of the new proposed ATC-based rules

This section evaluates the new proposed ATC-based rules over ATCSR on the single machine with the continuous setup and separable setup, respectively.

(a) ATCRCS vs. ATCSR

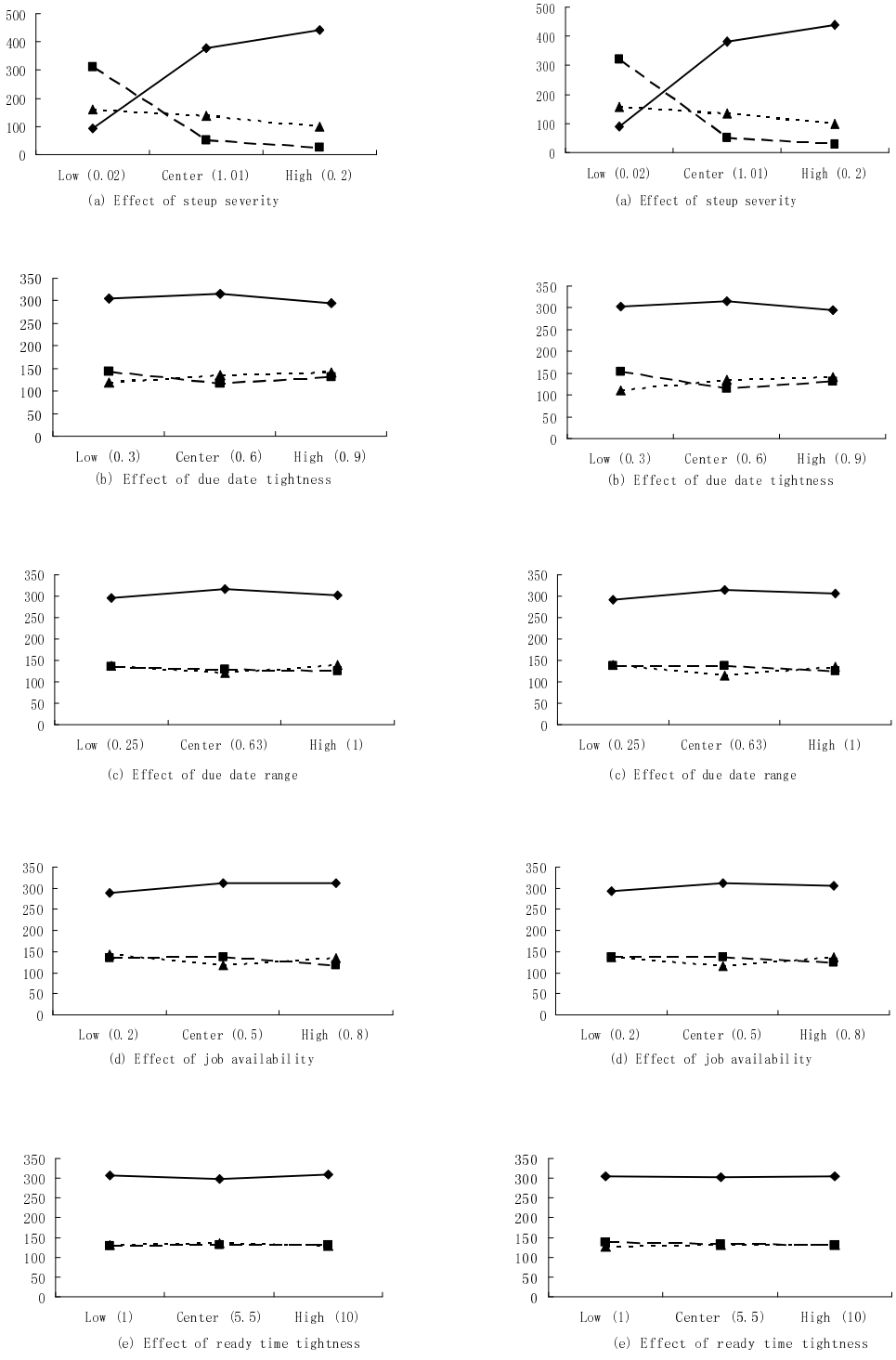
We compare ATCRCS with ATCSR for continuous setup under the best of the best measure and the territory measure.

The best of the best measure

Among the considered 1701 problems, ATCRCS performs better than, equal to, and worse than ATCSR in 915 problems (54%), 390 problems (23%) and 396 problems (23%), respectively. When ATCRCS outperforms ATCSR, the average reduction of tardiness is 5.1%. In the opposite case, ATCRCS gives 2.4% higher weighted tardiness than ATCSR.

Based on this measure, we also study the effect of each factor. The test results in figure 6(a) show ATCRCS outperforms ATCSR at all levels of all factors except at the low level of factor 1, setup severity. This figure shows that ATCSR performs better than ATCRCS only when the average setup time is short (2% of the average processing time). When the setup severity level is set at 30% and 60%, which are not parts of Pfund's level of factor 1, additional tests show that ATCSR performs better than ATCSR. When setup severity is 30%, ATCRCS gives 283 better problems, 104 equal problems, and 180 worse problems in 567 newly generated problems (also generated by Pfund's method, but at the

level value of 30% and 60% of factor 1). When the value of setup severity is increased to 60%, the results are 359, 75, and 133 problems respectively.

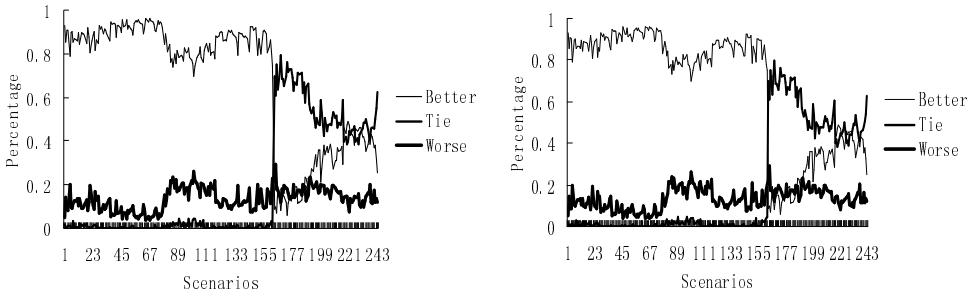


(a) ATCRCS vs. ATCSR (b) ATCRSS vs. ATCSR

Figure 6. Effect of factors

The territory measure

The results of a territory test are shown in figure 7(a) by scenarios (statistics of seven problems are summarized in each scenario of the figure). Figure 7(a) shows when setup severity is at high or center levels (scenarios 1 to 81 and scenarios 82 to 161, respectively), ATCRCS significantly outperforms ATCSR. When setup severity level is low (scenarios 163 to 243), many grid points yield equal performance for ATCRCS and ATCSR. We also note when the setup severity level is low and due date range tightness is central or low (scenarios 190 to 243), ATCRCS outperforms ATCSR; only when the setup severity level is low and the due date tightness level is high (scenarios 163 to 189), ATCSR slightly outperforms ATCRCS. We also note the curve of the worse percentage is relatively stable; it is not as sensitive to scenarios as the other two curves.



(a) ATCRCS vs. ATCSR (b) ATCRSS vs. ATCSR

Figure 7. The territory test

(b) ATCRSS vs. ATCSR

This section performs the same tests for the separable setup type. Because ATCRSS is the only rule to solve a problem with separable sequence dependent setup, ATCSR is

modified so that a setup can start at the decision time instead of the ready time of a job. The results of the test for ATCRSS are similar to that of ATCRCS.

The best of the best measure

Among the 1701 problems considered, ATCRSS performs better than, equal to, and worse than ATCSR in 912 problems (54%), 400 problems (23%) and 389 problems (23%), respectively. When ATCRSS outperforms ATCSR, the average reduction of tardiness is 5.3%. In the opposite case, ATCRSS gives 2.5% higher weighted tardiness than ATCSR. The test results in figure 6(b) show ATCRSS outperforms ATCSR at all levels of all factors except at the low level of factor 1, setup severity. When setup severity level is set at 30% and 60%, which are not parts of Pfund's level of factor 1, additional tests show that ATCRSS performs better than ATCSR. When the level of setup severity is 30%, ATCRSS gives 291 better problems, 121 equal problems, and 155 worse problems in the newly generated 567 problems mentioned in section 7.4.2. When the level of setup severity is increased to 60%, the results are 370, 75, and 122 problems, respectively.

The territory measure

For the separable setup, figure 7(b) shows a pattern similar to figure 7(a). When setup severity is at high or center levels (scenarios 1 to 81 and scenarios 82 to 161, respectively), ATCRSS significantly outperforms ATCSR. When setup severity level is low (scenarios 163 to 243), many grid points yield equal performance for ATCRSS and ATCSR. We also note when the setup severity level is low and due date range tightness is central or low (scenarios 190 to 243), ATCRSS outperforms ATCSR; only when the

setup severity level is low and due date tightness level is high (scenarios 163 to 189), ATCSR slightly outperforms ATCRSS. We also note the curve of the worse results is relatively stable; it is not as sensitive to scenarios as the other two curves.

(IV) The results from the proposed new rules vs. the optimal solution

This section compares the performances of ATCRCS and ATCRSS with optimal solutions. Six scenarios (5 jobs, 6 jobs, 7 jobs, 8 jobs, 9 jobs and 10 jobs) are tested on a single machine. Each scenario contains five randomly generated problems. Table 11 and figure 8 show the average computation time needed to achieve the optimal solution, we used Lingo 9.0 on a desktop with 3.0 GHz processor and 1TB RAM. It shows that it takes more than half of a day to get the optimal solution for 10 jobs on a single machine. However, for the dispatching method, it spends less than 1 second to generate a schedule for all tested problems. Appendix A and B are the proposed non-linear IP model and its lingo program. Appendix C is an example to schedule 5 jobs on a single machine by the lingo program.

Table 11. Computation time to get optimal solution for different cases

Time	Continuous type						Separable type					
	5 jobs	6 jobs	7 jobs	8 jobs	9 jobs	10 jobs	5 jobs	6 jobs	7 jobs	8 jobs	9 jobs	10 jobs
Test 1	1	4	141	388	4725	51356	1	5	57	287	7979	52433
Test 2	1	2	28	229	1734	29491	1	2	12	257	1414	46709
Test 3	1	3	34	338	7899	105364	1	3	66	229	6534	62539
Test 4	1	4	12	132	2915	31433	1	2	13	147	1838	26330
Test 5	1	2	21	285	4145	37235	1	3	50	223	3621	31310
Average (S)	1	3	47.2	274.4	4283.6	50975.8	1	3	39.6	228.6	4277.2	43864.2
Average (H)	0.00028	0.00083	0.01311	0.07622	1.18989	14.1599	0.00028	0.00083	0.011	0.0635	1.18811	12.1845

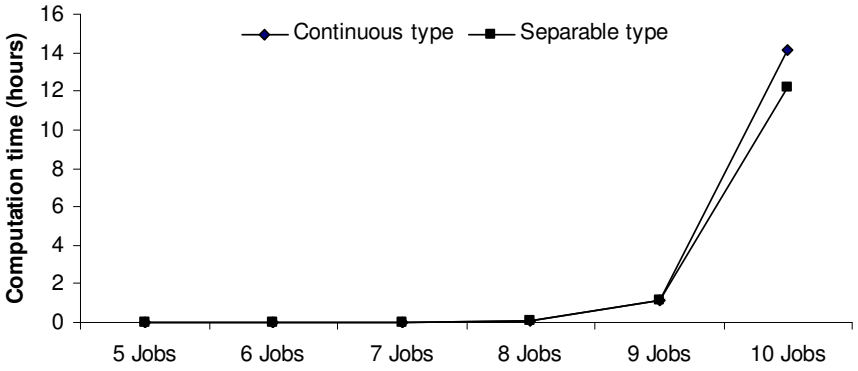


Figure 8. Average computation time to get optimal solution

Figure 9 shows the tardiness comparison (the percentage of the result from the proposed rules deviates from the optimal solution). Result shows among the 60 tested problems, the proposed ATC-based rules get an optimal solution 36 times. There are 13 problems to achieve a near optimal solution (deviation is within 5% of the optimal solution). The last 11 problems have a deviation over 10% of the optimal solution. The test shows that the proposed new rules generate a quality schedule, when the problem size is small.

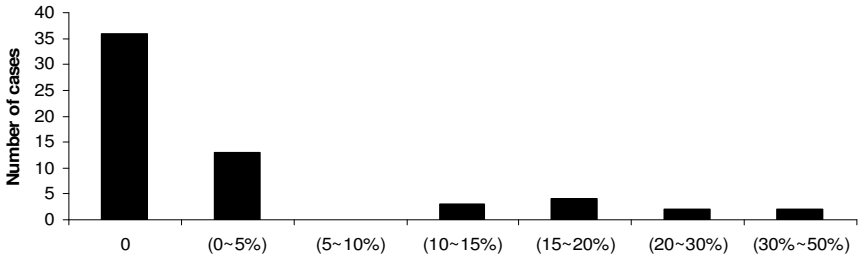


Figure 9. Deviation from optimal solution

In conclusion, the proposed new rules, ATCRCS and ATCRSS, are extensions of ATCSR (Pfund *et al*. 2008). All these three rules use the same terms but different formulas in some terms, for example, the proposed ATCRCS and ATCRSS use ready

time and sequence dependent setup in the WSPT term, while not for ATCSR. The logics or properties behind the WSPT term, the slack term, setup time term, and ready time term of these three rules are the same: a job has larger slack has a lower priority to be processed next; a job with a larger job weight or a shorter processing time has a higher priority to be processed next; a job with a shorter sequence dependent setup has a higher priority to be processed next; and a ready job has a higher priority to be processed next than a future job.

4. Performances of new rules on the identical parallel machines

This section evaluates the performance of the proposed ATC-based dispatching rules, ATCRCS and ATCRSS, on the identical parallel machines with other ATC-based dispatching rules.

4.1 Problem description and assumptions

This paper considers two problems: $Pm|r_j,s_{ij},con\ |\sum w_jT_j$ and $Pm|r_j,s_{ij},sep\ |\sum w_jT_j$. The first problem is stated as: there are n jobs arriving to m identical parallel machines at different times. Each job j has its ready time (r_j), processing time (p_j), due date (d_j), and job weight (w_j). The setup time (s_{ij}) of each pair of jobs i and j is sequence dependent and the continuous type, *con*. In general, s_{ij} is not equal to s_{ji}. The objective is to minimize the total weighted tardiness of jobs, $\sum_{j=1}^{n} w_jT_j$, where T_j is the tardiness of job j, $\max\{0,C_j-d_j\}$, and C_j is the competition time of job j. The second problem is the same as the first problem except that the setup is changed into the separable type, *sep*.

Both problems assume the following:

- The job attributes (p_j, d_j, w_j, r_j, s_{ij}) are known in advance.
- The machines are parallel identical.
- Each machine can process at most one job at each time.
- Job preemption is not allowed.

- Production interruptions such as machine breakdown and order cancellation do not happen.

4.2 Benchmark methods and design of experiment

Five ATC-based dispatching rules, BATCS, BATCSmod, ATCSR, ATCRCS and ATCRSS are studied in the experiments. When BATCS and BATCSmod are used as benchmark methods, their batch size is set at 1. ATCSR, ATCRCS and ATCRSS have three *k* values, while BATCS and BATCSmod have only two *k* values. These ATC-based indexes are given as below:

BATCS (Mason *et al*. 2002)

$$I_{BATCS}(t,i,j) = \frac{w_j}{p_j}\exp(-\frac{\max(d_j - p_j - r_j - t,0)}{k_1\overline{p}})\exp(-\frac{s_{ij}}{k_2\overline{s}}) \tag{10}$$

BATCSmod (Pfund *et al*. 2008)

$$I_{BATCSmod}(t,i,j) = \frac{w_j}{p_j}\exp(-\frac{\max(d_j - p_j + \max(r_j - t,0),0)}{k_1\overline{p}})\exp(-\frac{s_{ij}}{k_2\overline{s}}) \tag{11}$$

ATCSR (Pfund *et al*. 2008)

$$I_{ATCSR}(t,i,j) = \frac{w_j}{p_j}\exp(-\frac{\max(d_j - p_j - \max(r_j, t),0)}{k_1\overline{p}})\exp(-\frac{s_{ij}}{k_2\overline{s}})\exp(-\frac{\max(r_j - t,0)}{k_3\overline{p}}) \tag{12}$$

ATCRCS (Continuous sequence dependent setup)

$$I_{ATCRCS}(t,i,j) = \frac{w_j}{p_j + s_{ij} + \max(r_j - t,0)}\exp(-\frac{\max(d_j - p_j - s_{ij} - \max(r_j,t),0)}{k_1\overline{p}})\exp(-\frac{s_{ij}}{k_2\overline{s}})\exp(-\frac{\max(r_j - t,0)}{k_3\overline{p}}) \tag{13}$$

ATCRSS (Separable sequence dependent setup)

$$I_{ATCRSS}(t,i,j) = \frac{w_j}{p_j + \max(s_{ij}, r_j - t)} \exp\left(-\frac{\max(d_j - p_j - \max(r_j, t + s_{ij}), 0)}{k_1 \bar{p}}\right) \exp\left(-\frac{s_{ij}}{k_2 \bar{s}}\right) \exp\left(-\frac{\max(r_j - t - s_{ij}, 0)}{k_3 \bar{p}}\right) \quad (14)$$

Two experiments are performed to evaluate the performances of ATC-based rules on identical parallel machines scheduling. The first experiment completely repeats Pfund's 3^6 experimental design but considers more problems: Each of the six factors has three levels (low, center, and high). The number of machines is set at 5 (m=5). This experiment has 729 (3^6) scenarios. In each scenario, seven cases, or problems, are randomly generated. In total, 5103 (729 x 7) problems are tested in the first experiment. Table 12 shows factors and levels of Pfund *et al*. (2008). We evaluate the performance of ATC-based rules by two types of tests, *the best of the best test* and *the territory test*, which are explained in detail in sections 3.2.1

Table 12. Experiment of Pfund *et al.* (2008)

Factors	Pfund et al. (2008) 5 machines		
	Low	Center	High
Job machine factor	11	19	27
Setup severity factor	0.02	1.01	2
Due date tightness factor	0.3	0.6	0.9
Due dtae range factor	0.25	0.63	1
Job availability factor	0.2	0.5	0.8
Ready time factor	1	5.5	10

* The ready time rj is generated with uniform probability in the range [dj- r_τ*pj, dj], If (dj- r_τ*pj) is less than 0, a range of [0, dj] is used for ready time generation.

In the second experiment, we focus on the effect of the number of machines. This is a 3^7 experimental design extended from Pfund's 3^6 experiment. A new factor called machine number factor is added to the experiment of Pfund *et al*. (2008). The low, center, and high levels of this factor are set at 2, 4, and 6 respectively. In addition, the low, center, and high levels of the job machine factor, u, are set at 10, 20, and 30. The settings of the rest of the five factors (η, r, R, J_a and r_r) are exactly the same as those of Pfund. Table 13 contains nine sections, and shows the total number of jobs considered in each section. There are 2187 (3^7) scenarios in this experiment. Each scenario contains seven problems. In total, 15309 (2187 x 7) problems are considered in the second experiment.

Table 13. Number of jobs of the experiment

Job machine factor: μ	Number of machines		
	2 Machines	4 Machines	6 Machines
μ=10	20 jobs	40 jobs	60 jobs
μ=20	40 jobs	80 jobs	120 jobs
μ=30	60 jobs	120 jobs	180 jobs

4.3 Performance comparison of different ATC-based rules with CSDS

This section shows results of the two experiments mentioned in section 4.2 for the continuous sequence dependent setup cases and separable sequence dependent setup cases.

4.3.1 Performance comparison of different ATC-based rules with CSDS

(I) The best of the best test

For each scheduling problem, ATCRCS and ATCSR make 3146 (22 x 11 x 13) grids, and generate the same number of schedules. For BATCS and BATCSmod, 242

(22 x 11) grids and the same number of schedules are made. The schedule with the smallest total weighted tardiness among the 3146, or 242 schedules is selected as the corresponding rule's final schedule for the considered problem. We compare the best selected schedules, one from each scheduling method, and check which method gives the best schedule.

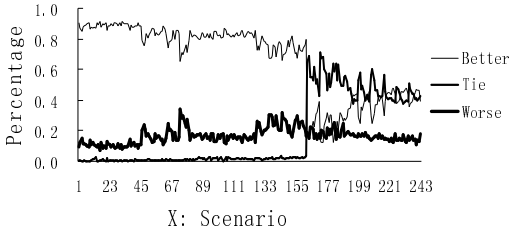
In this test, ATCRCS, ATCSR, BATCS and BATCSmod have 3407 (66.76%), 2157 (42.27%), 88 (1.7%) and 30 (0.59%) times to get the best solution out of the 5103 problems generated in the first experiment in section 4.2. The sum of percentage is more than 100% due to ties. This test shows that ATCRCS yields more number of the best cases than ATCSR, which significantly outperforms BATCS and BATCSmod.

Next, we compare the two best methods from the above experiment, ATCSR and ATCRCS, and study the effect of each factor. In the 1701 problems (1/3 of 5103) where the setup severity factor is low (average setup time is short, i.e., 2% of the average processing time), ATCSR slightly outperforms ATCRCS (39% to 32% of 1701 cases); they show the same tardiness level for the rest cases (29% of 1701 cases). Amongst the total of 5107 problems, there are 2908 (57%) problems where ATCRCS is better, 536 (11%) problems are tied, and 1659 (32%) problem where ATCRCS is worse. For problems where ATCRCS gives better results, the average of improvements over ATCSR is about 3%. For problems where ATCSR gives better results, ATCRCS yields worse results at an average regression of about 2.1% over ATCSR.

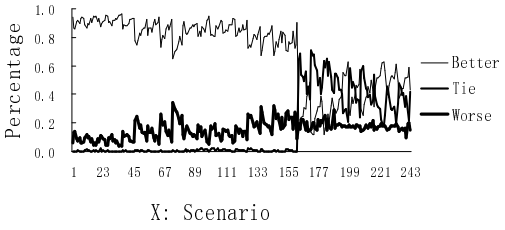
(II) The territory test

This section performs *the territory test* as follows: for each of the 5103 problems generated in 4.2, the results from ATCRCS and ATCSR are compared at each of all 3146 grid points and the percentages of grid points at which ATCRCS performs better than, equal to and worse than ATCSR are recorded. The test uses the average values of the percentages of the seven problems of each scenario. This analysis is important in relation to the regression method because the k values estimated by the regression method are somewhat away from the best grid point in a random fashion. When the best k values are not estimated accurately, the scheduling method with a larger favorable territory is likely to give a better schedule. (On the other hand, when the regression equation gives the best k values accurately, *the best of the best test* introduced in the previous section is more relevant because it compares the best points selected by ATC-based rules.)

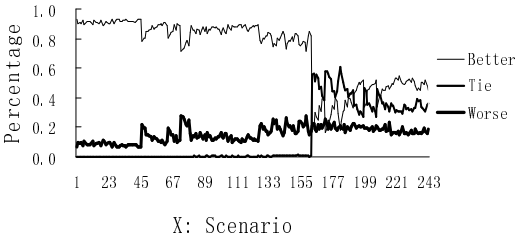
Figure 10 (a-i, a-ii, and a-iii) shows that when setup severity is at the high or center levels (scenarios 1 to 81 or scenarios 82 to 162, respectively) with 55, 95, and 135 jobs on 5 machines, respectively, ATCRCS performs significantly better than ATCSR. It also shows that the number of jobs does not affect the test result much. When setup severity level is low (scenarios 163 to 243), many grid points yield the same total weighted tardiness for these two methods. The percentage of ties decreases when the number of jobs increases. It can also be noticed that when the setup severity level is low and due date range tightness is central or low (scenarios 190 to 243), ATCRCS outperforms ATCSR again. Only when the setup severity level is low and due date tightness level is high, ATCSR is found to have less chance to outperform ATCRCS in the territory test (scenarios 163 to 189).



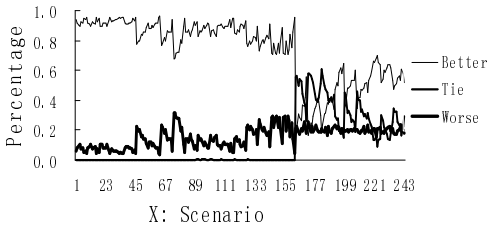
(i) 55 jobs on 5 machines



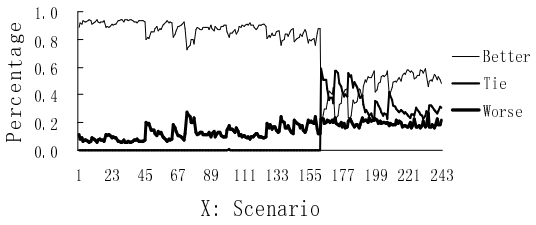
(i) 55 jobs on 5 machines



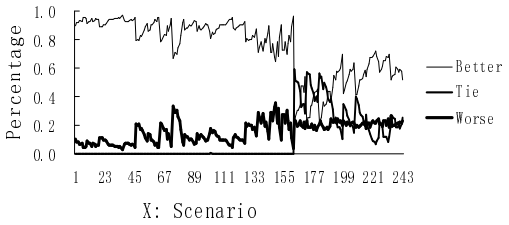
(ii) 95 jobs on 5 machines



(ii) 95 jobs on 5 machines



(iii) 135 jobs on 5 machines



(iii) 135 jobs on 5 machines

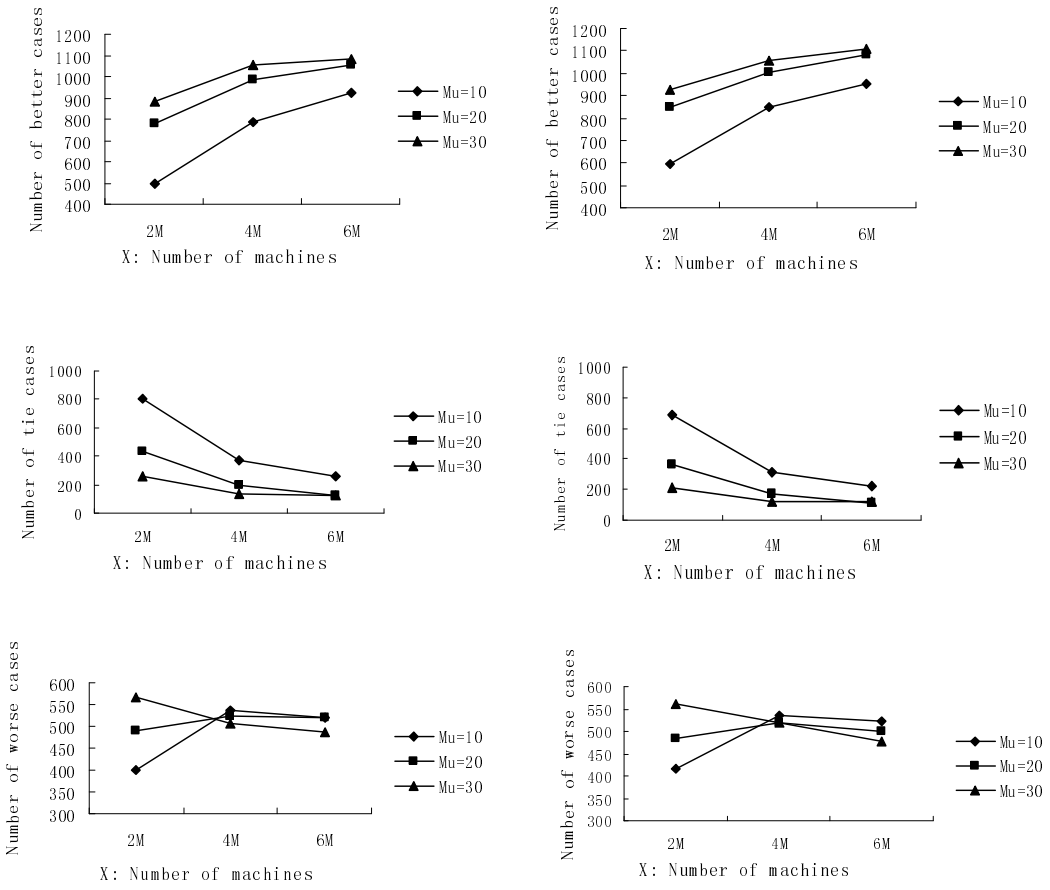
(a) ATCRCS vs. ATCSR (b) ATCRSS vs. ATCSR

Figure 10. Territory test

(III) Effect of the number of machines

Figure 11(a) shows the effect of the number of parallel machines. It shows that when the level of the machine number factor μ increases, ATCRCS outperforms

ATCSR more. Additionally, for a given value of job machine factor, μ, using more machines yields more number of better cases.



(a) ATCRCS vs. ATCSR (b) ATCRSS vs. ATCSR

Figure 11. The effect of the number of machines

4.3.2 Performance comparison of different ATC-based rules with SSDS

This section shows the results of the two experiments explained in section 4.2 for the separable sequence dependent setup cases. Due to the lack of benchmark methods for the separable sequence dependent setup case, BATCS, BATCSmod, and ATCSR are slightly modified as follows and compared with ATCRSS: if a future job is selected to process next, its setup is allowed to start as soon as the last scheduled job is finished.

(I) The best of the best test

In this test, ATCRSS, ATCSR, BATCS, and BATCSmod have 3517 (68.92%), 1874 (36.72%), 101 (1.98%), and 45 times (0.88%) to get the best solution for the 5103 problems generated in the experiment in section 4.2. The sum of percentage is more than 100% because of ties. This test shows that ATCSSR yields more number of the best cases than ATCSR, which significantly outperforms BATCS and BATCSmod.

Next, we compare the two best methods, ATCSR and ATCRSS, in more detail. The result shows ATCRSS outperforms ATCSR at all levels on all factors, even when the setup severity is low (average setup time is short, i.e., 2% of the average processing time). Amongst the total of 5107 problems, there are 3189 (62%) problems where ATCSSR is better, 385 (8%) problems are tied, and 1529 (30%) problem where ATCRSS is worse. For problems where ATCRSS gives better results, the average of improvements over ATCSR is about 3.2%. For problems where ATCSR gives better results, ATCRSS yields worse results at an average regression of about 2.0% over ATCSR.

(II) The territory test

Figure 10(b-i, b-ii, and b-iii) shows, when setup severity is at the high or center levels (scenarios 1 to 81 or scenarios 82 to 162, respectively) for 55 jobs, 95 jobs, and 135 jobs on 5 machines, ATCRSS performs significantly better than ATCSR, and the number of jobs does not affect the result significantly. When setup severity level is low (scenarios 163 to 243), many grid points yield the same total weighted tardiness.

It can be noticed that when the setup severity level is low, and due date tightness level is central or low (scenarios 190 to 243), ATCRSS outperforms ATCSR again. Also, under this condition, the rate of ties decreases with increasing number of jobs. Only when the setup severity factor level is low and due date tightness level is high, ATCSR has less chance to outperform ATCRSS in the territory test (scenarios 163 to 189).

(III) Effect of the number of machines

Figure 11(b) shows the effect of the number of machines. It tells that when the value of the machine number factor, μ, increases, ATCRSS outperforms ATCSR more. Additionally, for a given value of job machine factor, μ, using more machines yields more number of better cases.

4.4 Computation time

A larger search range and smaller grid size increase computation time while improving the quality of the final schedule; the computation time and quality of schedule need to be balanced in the application of the scheduling procedure. Table 14 shows the computation time of scheduling. All 5301 problems generated in section 4.2 are tested on a 32 bit notebook with Pentium (1.86GHz) processor and 1GB RAM. We observe that using more scaling parameters improves the quality of results, but needs little bite more computation time. Table 14 also shows the grid method is computationally fast enough for most real applications; however, a regression method will still be helpful when scheduling is needed soon. In appendix D, we modify the single machine model to get a mathematic model for the parallel machines cases. Appendix E discusses the application of the math model proposed in appendix D.

Table 14. The computation time

5 machines	Methods with 2 scaling parameters (BATCS and BATCSmod)	Methods with 3 scaling parameters (ATCSR, ATCRCS, and ATCRSS)
55 jobs	Less than 1 second/problem	Less than 1 second/problem
95 jobs	Less than 1 second/problem	About 1.5 second/problem
135 jobs	Less than 1 second/problem	About 4.5 seconds/problem

5. THE PROPOSED LOOK_AHEAD HEURISTIC (LAIPM)

The section introduces the proposed look-ahead heuristic and evaluates its performances on the identical parallel machines. The considered setup is the continuous type.

5.1 Introduction and potential application

The proposed heuristic, LAIPM, is different from other look-ahead heuristics in the following ways. First, the proposed LAIPM is a search-based heuristic: it generates multiple schedules and selects the schedule with the smallest total weighted tardiness as the final schedule (these generated schedules are created by different combinations of scaling parameters). Second, the concept of look-ahead has two twofold meanings: (1) to select the next job on a machine, only the available jobs and some near future jobs are considered; (2) among considered jobs at the decision time, at most two jobs are selected by ATCRCS rule and at most one job is kept. This is done by comparing the total weighted tardiness of these two jobs to that of the reversed sequence. Finally, LAIPM uses a job switching heuristic to generate another possible iteration schedule, which allows selected jobs to be switched on all machines.

The potential customers of the proposed LAIPM are from both manufacturing industry and service industry. In the manufacturing industry, an example of the parallel machines scheduling with the sequence dependent setup is the print shop which has parallel print machines. The color change from a dark color to a lighter color takes longer time than in the opposite case. In this example, the color changes are deemed as the sequence dependent setups. Press die change is example at the machine level, and assembly line setup is an example at the production line level.

In the service industry, an example is to generate schedules for maintenance workers. In this case, maintenance staffs are the parallel machines. Failures or repair requests are comparable to jobs to be processed on the parallel machines. The traveling times between failures is deemed as sequence dependent setup. The maintenance time is comparable to the job's processing time. Another example in the service industry is the handicapped senior riding service. Handicapped persons call to request a wheel-chair lift vehicle riding. The service agents know the departing and destination location for each request. With limited resource (vehicles), they also encounter the parallel machines scheduling problems with sequence dependent setup. In this example, the distance between two calls is comparable to the sequence dependent setup time.

5.2 Logic and flow chart of the LAIPM heuristic

To get the final schedule, LAIMP uses bellowing steps:

For a certain scaling parameters combination,

Step 1. Select the initial job on each machine. Twofold one-step look-ahead is used to select the initial job for each machine (one-step means consider available jobs and the nearest future job).

Step 2. Check the number of unscheduled jobs:

(a) if no unscheduled jobs, go to step 4, pairwise exchange.

(b) if there is one unscheduled jobs, assign it to the machine with the smallest finish time and go to step 3, job switching heuristic.

(c) if there are two or more unscheduled jobs, identify the critical machine which has the smallest finish time and use twofold one-step look-ahead to select a job (the critical job) on it. Reduce the number of unscheduled jobs by one and compute the look-ahead thresh which is the sum of the finish time of the critical job and the average of setup time.

2.1. For other non critical machines, we consider them by the increasing order of the finish time: the non critical machine with the smallest finish time is considered first for its next job selection. After all machines finish selecting their next job, a possible iteration schedule is created (Not all non machines must have the next selected job in the iteration), go to step 3, job switching heuristic. To decide the next job on the considered non critical machine, we start to count the number of unscheduled jobs.

(a) if all jobs are scheduled, go to step 3, job switching heuristic.

(b) if one job is unscheduled, assign it to the considered non critical machine. Reduce the number of unscheduled jobs by one and go to step 3, job switching heuristic.

(c) if two or more jobs are unscheduled, we only consider jobs whose ready time is smaller or equal to the look-ahead thresh. These jobs are called qualified jobs. Twofold look-ahead is used to select at most two qualified jobs.

(1) If no qualified jobs, go to 2.1 (consider the job selection on the next non-critical machine).

(2) If only one qualified job is found, assign this job to the considered non critical machine. Reduce the number of unscheduled jobs by one and go to 2.1 (consider the job selection on the next non-critical machine).

(3) If more than two qualified jobs are found, use twofold look-ahead to assign one of them to the considered non critical machine. Reduce the number of unscheduled job by one and go to 2.1 (consider the job selection on the next non-critical machine).

Step 3. Job switching heuristic. The selected jobs on machines are used as inputs of job switching heuristic. These selected jobs are switched among all machines to determine another possible iteration schedule which has a small total weighted tardiness value. This possible iteration schedule is compared with the possible iteration schedule which is created in step 2. If the schedule from job switching heuristic has a smaller total weighted tardiness and has a smaller sum of the machine finish time, using the schedule generated by the job switching as the schedule of the considered iteration, otherwise, using the schedule generated before the job switching heuristic as the schedule of the considered iteration. More detailed information of job switching heuristic is illustrated in section 5.3. Count the number of unscheduled job and go to step 2.

Step 4. Pairwise exchange is used to reduce the total weighted tardiness for the created schedule of each machine. More detailed information about pairwise exchange is given in section 5.4. Until now, we get a schedule for a combination of scaling parameters.

Creating schedules by other combinations of scaling parameters, among generated multiple schedules, the schedule with the smallest total weighted tardiness is the final schedule. The flow chart of the proposed heuristic is shown in figure 12.

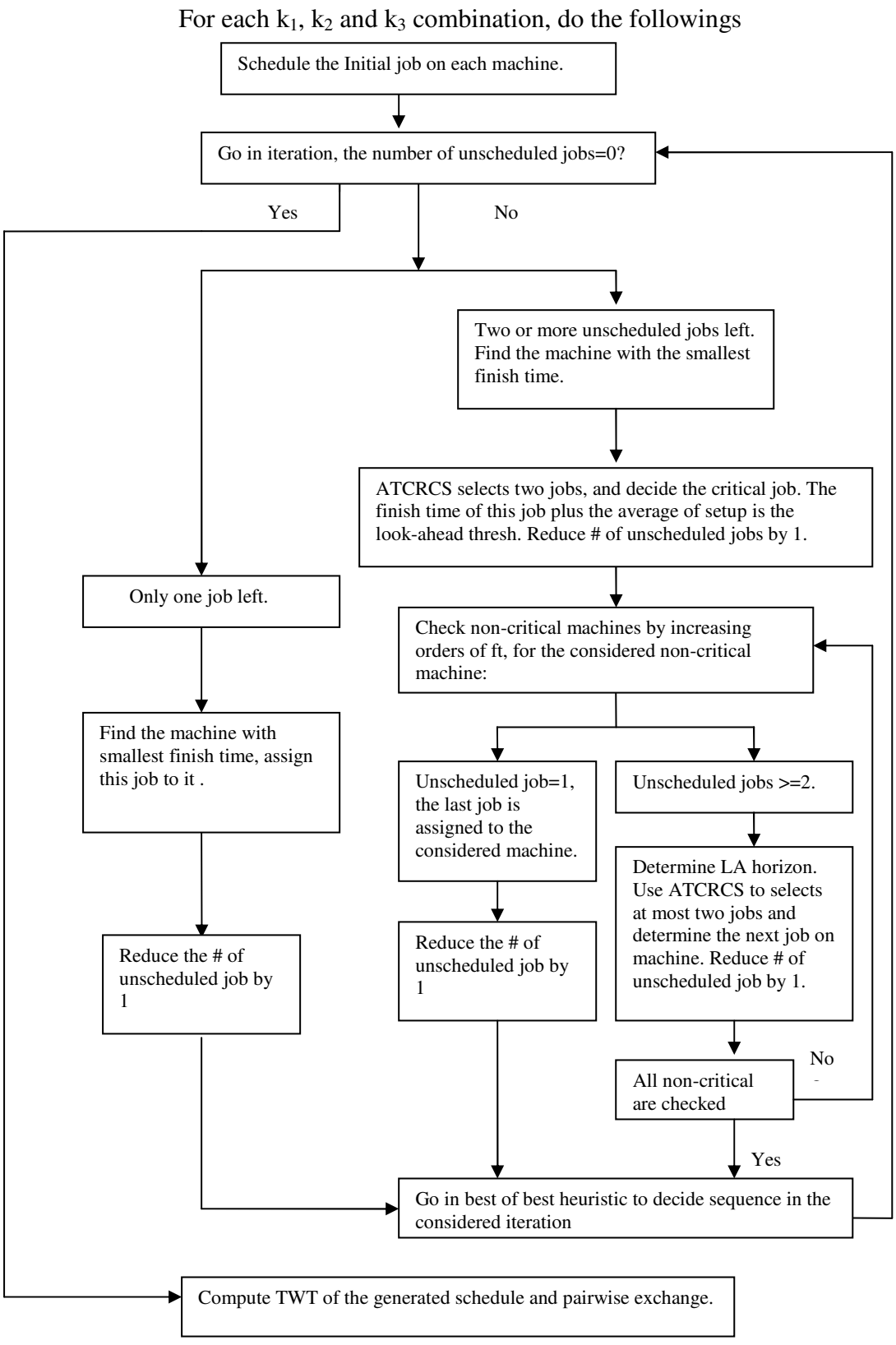


Figure 12. Flow Chart of the proposed heuristic Phase

5.3 The job switching heuristic

To allow jobs switching on machine and among machines, we proposed *job switching heuristic*. This heuristic is carried out after all machines finish considering their next job selection. Information, like the selected jobs, is used as inputs of the job switching heuristic.

Consider a given iteration in the 3 machines' scheduling, machines 1 and 2 have assigned job, F and C (Machine 3 do not get assigned job) in the considered iteration. At the beginning of this iteration, the last jobs on these three machines are A, E and B, respectively. The input information is summarized in table 15:

Table 15. Input of smaller in smaller heuristic

Machine #	Last job on machine	assigned job on machine
1*	A	F
2	E	C
3	B	

* The critical machine

Job switching heuristic considers job switching among all machines: each selected job have chance to be scheduled before the last job of each machine or after the last scheduled job on each machine. In this example, job sequence A-F, F-A, A-C, and C-A are considered on machine 1; job sequence E-F, F-E, E-C, and C-E are considered on machine 2; and job sequence B-F, F-B, C-B, and BC are considered for machine 3. Let $TWT_{m,AB}$ donate the total weighted tardiness of the schedule on machine *m* where the last two jobs of the schedule are job A and job B. The total weighted tardiness values of the considered sequences on machine *m* are computed. Suppose these total weighted tardiness values are computed and given in table 16.

Table 16. Total weighted tardiness of considered sequence

Machine #	Value of $TWT_{m,AB}$			
1*	$TWT_{3,AF}$=10	$TWT_{3,FA}$=8	$TWT_{3,AC}$=6	$TWT_{3,CA}$=3
2	$TWT_{5,EF}$=5	$TWT_{5,FE}$=2	$TWT_{5,EC}$=9	$TWT_{5,CE}$=2
3	$TWT_{5,BF}$=8	$TWT_{5,FB}$=4	$TWT_{5,BC}$=12	$TWT_{5,CB}$=4

* The critical machine

In table 16, we want to decide the position of selected jobs (job F and job C) so that the three parallel machines yields small total weighted tardiness. For the general cases, table 16 has $2n * m$ values, where m is the number of machines that have selected job in the considered iteration, n is the number of machines. The positions of job F and job C can be determined by solving a linear binary programming with several constrains. The mathematical model of above example is given as:

$$\text{Min} \sum_{m} \sum_{seq} TWT_{m,seq} * X_{m,seq} \quad \text{(seq=AF, FA, AC, CA, EF, FE, EC, CE, BF, FB, BC and CB; m=1,2,3)} \tag{15}$$

Subject to:

$$X_{1,AF} + X_{1,FA} + X_{1,AC} + X_{1,CA} \leqslant 1 \tag{16}$$

$$X_{2,EF} + X_{2,FE} + X_{2,EC} + X_{2,CE} \leqslant 1 \tag{17}$$

$$X_{3,BF} + X_{3,FB} + X_{3,BC} + X_{3,CB} = 1 \tag{18}$$

$$X_{1,AF} + X_{1,FA} + X_{2,EF} + X_{2,FE} + X_{3,BF} + X_{3,FB} = 1 \tag{19}$$

$$X_{1,AC} + X_{1,CA} + X_{2,EC} + X_{2,CE} + X_{3,BC} + X_{3,CB} = 1 \tag{20}$$

where $X_{m,AB}$ is the binary solution. $X_{m,AB}$=1 represents the schedule where the last two jobs are A and B is on machine m.

Constrains 16 to 20 can be divided into two groups: (1) row constrains or the machine constrains (16, 17 and 18). Each machine can get at most one selected job. (2) The column constrains or the selected job constrains (19 and 20). Each selected job must be only scheduled by one time. Also, on each machine, each selected job has two possible positions, either before the last scheduled job or after the last scheduled job.

To quickly decide the positions of the selected jobs, job F and job C, we horizontally combine every two TWT values in table 17 into one value, and put these values in table 17.

Table 17. Combined TWT table

Machine #	Modified TWT value	
1*	18 (10+8)	9 (6+3)
2	7 (5+2)	11 (9+2)
3	12 (8+4)	16 (12+4)

* The critical machine

Table 17 only contains 6 values (in general cases, $n*m$ values). The problem becomes simpler (the considered problem is changed into the assignment problem): it is similar to find a value from each column from table 17 and satisfied with a constraint that the number of selected value of each row is smaller or equal to one. In this example, it is clear that 7 and 9 give the smaller sum value than others. The value 7 and 9 in table 17 are then tracked back to get solution in table 16: 7 is the sum of 5 and 2. 9 is the sum of 6 and 3 in table 16. To decide the sequence or the position of the selected job C and F, we select the smaller value between 5 and 2 and the smaller value between 6 and 3 as solutions. In this example, the schedule whose last two jobs on machine 1 is C-A. The schedule whose last two jobs on machine 2 is F-E. Once

this sub-problem is solved, we have two schedules, one is from the job switching heuristic and the other is generated before the job switching heuristic. These two possible schedules are shown in table 18 and 19:

Table 18. Possible schedule (Before job switching heuristic)

Machine	Last job	Assigned job
1*	A	F
2	E	C
3	B	

Table 19. Possible schedule (from job switching heuristic)

Machine	Last job	Assigned job
1*	C	A
2	F	E
3	B	

To determine the schedule of the considered iteration, we compare the total weighted tardiness of these two possible iteration schedules. The schedule of the considered iteration is decided by the value of TWT and TWT_{switch}, where TWT is the total weighted tardiness of the schedule before job switching heuristic, while TWT_{switch} is the total weighted tardiness of the schedule from the job switching heuristic. If TWT_{switch} is smaller and the sum of machine finish time becomes smaller, the schedule generated by the job switching heuristic is the schedule of the considered iteration; otherwise, the schedule generated before the job switching heuristic is the schedule of the considered iteration.

5.4 Pairwise exchange

Pairwise exchange is an improvement technique by switching two selected jobs' position in predefined orders. It is used as soon as all jobs are assigned to machines.

Suppose a sequence on a machine has four jobs, A-B-C-D. The pairwise exchange orderly considers following changes:

B-A-C-D; (switch A and B)

C-B-A-D; (switch A and C)

D-B-C-A; (switch A and D)

A-C-B-D; (switch B and C)

A-D-C-B; (switch B and D)

A-B-D-C; (switch C and D)

The final sequence is the one with the smallest total weighted tardiness among all considered schedules. In literature, this pairwise exchange technique is also suggested in the improvement phase of the look-ahead heuristic of Chang *et al*. (2004).

5.5 An example (8 jobs on 6 machines)

This section uses the proposed look-ahead heuristic, LAIPM, to schedule 8 jobs on 6 machines. The grid settings of scaling parameters, k_1, k_2, and k_3, are that of Rene and Lars (2009):

k_1: 0.2, 1, 1.6, 2.4, 3.6, 4.8, 6

k_2: 0.1, 0.7, 1.3, 1.9

k_3: 0.001, 0.005, 0.05, 0.6, 1.2

The processing time, due date, sequence dependent setup time, job weight and ready time of these 8 jobs are given in table 20 and table 21.

Table 20. Processing time, due date, job weight and ready time

	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8
Processing time	8	2	6	9	8	5	3	2
Due date	10	5	13	12	10	8	7	8
Job weight	4	2	8	3	1	6	4	3
Ready time	4	0	5	0	7	0	0	5

Table 21. Sequence dependent setup time

	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8
Initial setup	3	2	4	5	1	3	4	2
J1	0	1	1	3	2	2	2	3
J2	3	0	3	4	1	5	1	4
J3	1	5	0	2	4	1	4	1
J4	5	2	1	0	4	3	3	5
J5	2	3	5	2	0	2	1	2
J6	4	2	3	5	1	0	2	2
J7	5	2	4	3	3	3	0	2
J8	2	4	5	3	2	1	1	0

*IST: Initial Setup Time

To use the proposed look ahead heuristic generate schedule, we first consider the combination of scaling parameters (k_1=0.2, k_2=0.1, k_3=0.001).

Step 1. Decided initial job on each machine

The initial jobs on all machines (machine 1 to machine 6) are J6, J7, J4, J1, J8 and J3, respectively. In this step, machine 1 is considered first, J6 is selected by the one step look-ahead ATCRCS.

Step 2. Job selection on the critical machine

After step 1, the number of unscheduled jobs is 2. We next identify the critical machine. In this example, the critical machine is machine 2 whose finish time of job 7 is 7. To decide the critical job on the critical machine (machine 2), only two unscheduled jobs (J2 and J5) left to be considered. The total weighted tardiness of sequence J7-J2-J5 is compared with that of sequence J7-J5-J2. Because sequence J7-J2-J5 has smaller total weighted tardiness (value is 22) than sequence J7-J5-J2 (value is 44). J2 is assigned on machine 2 as the critical job. The average setup time is 3. The look-ahead thresh is 14, which is the sum of the finish time of J2 (value is 11) and the average setup time (value is 3).

Step 3. Job selection for non-critical machine

The non-critical machines are machines besides of machine 2. Machine 1 is considered first because of its smallest finish time (value is 8). The last job, J5, is assigned on machine 1. Now, all jobs are assigned, the considered iteration is the last iteration. We have a possible schedule before the job switching heuristic. This schedule is given as below:

Machine 1: J6-J5;

Machine 2; J7-J2;

Machine 3: J4;

Machine 4: J1;

Machine 5: J8;

Machine 6: J3;

The total weighted tardiness on all machines is 64.

Job switching heuristic uses the selected jobs, J2 and J5, as inputs. These two jobs are switched on all machines. In this example, we consider following sequences:

J6-J2, J2-J6, J6-J5, and J5-J6 on machine 1;

J7-J2, J2-J7, J7-J5, and J5-J7 on machine 2;

J4-J2, J2-J4, J4-J5, and J5-J4 on machine 3;

J1-J2, J2-J1, J1-J5, and J5-J1 on machine 4;

J8-J2, J2-J8, J8-J5, and J5-J8 on machine 5;

J3-J2, J2-J3, J3-J5, and J5-J3 on machine 6;

To determine the position of the selected jobs (J2 and J5), the total weighted tardiness of all above considered sequences is computed and put in table 22.

Table 22. TWT of considered sequences on machines

Machine 1	TWT_{62}=14	TWT_{26}=36	TWT_{65}=7	TWT_{56}=96
Machine 2	TWT_{72}=18	**TWT_{27}=4**	TWT_{75}=13	TWT_{57}=58
Machine 3	TWT_{42}=32	TWT_{24}=15	TWT_{45}=22	TWT_{54}=51
Machine 4	TWT_{12}=46	TWT_{21}=20	TWT_{15}=35	TWT_{51}=70
Machine 5	TWT_{82}=23	TWT_{28}=9	**TWT_{85}=12**	TWT_{58}=42
Machine 6	TWT_{32}=50	TWT_{23}=8	TWT_{35}=33	TWT_{53}=118

After computing the total weighted tardiness of above sequences, every two cells in table 22 are combined into one cell horizontally to get table 23.

Table 23. Combined TWT table.

Machine 1	50 (14+36)	103 (7+96)
Machine 2	**22** (18+4)	71 (13+58)
Machine 3	47 (32+15)	73 (22+51)
Machine 4	66 (46+20)	105 (35+70)
Machine 5	32 (23+9)	**54** (12+42)
Machine 6	58 (50+8)	151 (33+118)

In table 23, we want to find a value from each column and satisfied the constraint that the number of selected values in each row is smaller or equal to 1. In this example, the sum of 22 and 54 gives smaller value than other combinations (these two values are bold in table 15). Also, from table 23, we know that 22 is the sum of 18 and 4 on machine 2, while 54 is the sum of 12 and 42 on machine 5. We then go back to table 22 to find the selected jobs' position. Table 22 shows that the total weighted tardiness of sequence J2-J7 on machine 2 is 4, and the total weighted tardiness of sequence J8-J5 is 12. The possible schedule generated by the job switching heuristic is:

Machine 1: J6;

Machine 2; J2-J7;

Machine 3: J4;

Machine 4: J1;

Machine 5; J8-J5;

Machine 6: J3;

Until now, we have two possible iteration schedules. One is generated before job switching heuristic and the other is generated by the job switching heuristic. Table 24 shows these two schedules and their total weighted tardiness.

Table 24. Two possible schedules and their tardiness.

Schedule	Before job switching heuristic	By job switching heuristic
Machine 1	J6-J5	J6
Machine 2	J7-J2	J2-J7
Machine 3	J4	J4
Machine 4	J1	J1
Machine 5	J8	J8-J5
Machine 6	J3	J3
TWT	TWT=64	TWT_{switch} =58

The total weighted tardiness TWT_{switch} (by job switching heuristic) on all machines is 58. It is smaller than TWT before the job switching heuristic (64). Because all jobs are scheduled, we use the schedule generated by the job switching heuristic as the schedule before pairwise exchange.

Step 5. Pairwise exchange

In this example, pairwise exchange does not further reduce the total weighted tardiness on machines. For the considered combination of scaling parameters (k_1=0.2, k_2=0.1, k_3=0.001), the schedule on each machine after pairwise exchange is the same as the schedule generated by the job switching heuristic:

Machine 1: J6;

Machine 2; J2-J7;

Machine 3: J4;

Machine 4: J1;

Machine 5; J8-J5;

Machine 6: J3;

In similar, we generate schedules by other combinations of scaling parameters. However, their total weighted tardiness is not as good as the schedule generated by using k_1=0.2, k_2=0.1, and k_3=0.001. The above schedule is selected as the final schedule for this 8 jobs, 6 machines example.

5.6 Experiment design and benchmark methods

We use the experiment of Xi and Jang (2012) to evaluate the proposed look-ahead heuristic. The experiment of Yue and Jang is an extension of Pfund *et al*. (2008) for the identical parallel machines scheduling. The differences of these two experiments are shown in table 25. The considered experiment has 729 (3^6) scenarios. In each scenario, seven cases, or problems, are randomly generated. In total, 5103 (729 x 7) problems are tested on 6 identical machines.

Table 25. Comparison of experiment in this research and that of Pfund *et al*. (2008)

Factors	Pfund et al.(2008): 5 machines			Yue and Jang (2012): 6 machines		
	Low	Center	High	Low	Center	High
Job machine factor	11	19	27	10	20	30
Setup severity factor	0.02	1.01	2	0.02	1.01	2
Due date tightness factor	0.3	0.6	0.9	0.3	0.6	0.9
Due dtae range factor	0.25	0.63	1	0.25	0.63	1
Job availability factor	0.2	0.5	0.8	0.2	0.5	0.8
Ready time factor	1	5.5	10	1	5.5	10

★The ready time r_j is generated with uniform probability in the range [d_j- r_τ*p_j, d_j], If (d_j- r_τ*p_j) is less than 0, a range of [0, d_j] is used for ready time generation.

The benchmark methods are divided into two groups: (1) Look-ahead scheduling heuristics, and (2) Non-look ahead heuristic.

(1) Look-ahead scheduling heuristic

Four look-ahead heuristics, heuristic of Chang *et al*. (2004) and three modified heuristics by Yoon et al. (2011), are studied in the experiment. Chang's heuristic targets to minimize the total weighted tardiness and proved to be effective when the problem size is small. Without considering sequence dependent setup, Yoon *et al*. (2011) propose three look-ahead heuristics to minimize the total weighted tardiness with equal ready time. Their heuristics exclude non-urgent jobs from the candidates for the next job selection. Jobs are divided into urgent jobs and non-urgent jobs by a computed thresh value. In order to compare these heuristics with other look-ahead heuristics that consider sequence dependent setup, like Chang's and LAIPM, we make a modification so that these three heuristics can solve the total weighted tardiness problem with future ready time and sequence dependent setup. Below shows how these three heuristics (by Yoon *et al*. 2011) are modified: the idea of the modification is that the sequence dependent setup and the possible machine idle time is deemed as a part of the processing time, in the formula, we substitute p_j from $(r_j\text{-}t,0)+s_{ij}+p_j$

Modified heuristic 1:

Step 1: At decision time t, compute T= $\frac{\sum_{j\in u}\max(\max(r_j-t,0)+s_{ij}+p_j,d_j)}{|u|}$, and construct set G as $\{j|\, j\in u, d_j\le T\}$, where u is unscheduled job set and $|u|$ is the number of non-scheduled jobs.

Step 2: If G ≠ Ø, then select job j in set G with the minimum value of $\max\{\max(r_j - t,0) + s_{ij} + p_j, d_j - \max(r_j,t)\}/w_j$.
Step 3: If G = Ø, then select job j with the minimum value of $\{d_j - \max(r_j,t)\}/w_j$.

Modified heuristic 2:

Step 1: At decision time t, compute T= $\frac{\alpha + B}{2}$, where α is $\max_{j\in u}(\max(r_j - t,0) + s_{ij} + p_j, d_j)$, β is $\min_{j\in u}(\max(r_j - t,0) + s_{ij} + p_j, d_j)$, and u is unscheduled job set. Set G includes jobs $\{j | j \in u, d_j \le T\}$.

Step 2: If G ≠ Ø, then select job j in set G with the minimum value of $\max\{\max(r_j - t,0) + s_{ij} + p_j, d_j - \max(r_j,t)\}/w_j$.
Step 3: If G = Ø, then select job j with the minimum value of $\{d_j - \max(r_j,t)\}/w_j$.

Modified heuristic 3:

Step 1: Among unscheduled jobs, identify the job with the smallest Weighted Modified Due Date (WMDD) index value: $WMDD_j$ = $\frac{\max\{\max(r_j - t,0) + s_{ij} + p_j, d_j - t\}}{w_j}$. This job is noted as j*.

Step 2: At decision time t, compute T= $\max\{\max(r_{j*} - t,0) + s_{ij*} + p_{j*}, d_{j*}\}$. Set G includes jobs $\{j | j \in u, d_j \le T\}$, where u is unscheduled job set.

Step 3: If G ≠ Ø, then select job j in set G with the minimum value of $\max\{\max(r_j - t,0) + s_{ij} + p_j, d_j - \max(r_j,t)\}/w_j$.
Step 4: If G = Ø, then select job j with the minimum value of $\{d_j - \max(r_j,t)\}/w_j$.

(2) Non look-ahead heuristic

The proposed LAIPM is also compared with ATCRCS (Xi and Jang 2012) which is proved as an effective ATC-based heuristic on the identical parallel than ATCSR, ATCS, BATCS, and BATCSmod.

5.7 Performance evaluation

This section compares the proposed LAIPM with benchmark heuristics in minimizing the total weighted tardiness on the identical parallel machines with the consideration of sequence dependent setup and future ready time.

5.7.1 LAIPM vs. look-ahead heuristics

This section evaluates the performance of the proposed look-ahead heuristic and selected look-ahead benchmark heuristics. The average rate of the total weighted tardiness obtained by each of the heuristics over the total tardiness obtained by the look-ahead heuristic of Chang et al. (2004) is shown in table 26. Results are sorted by factors. Table 26 shows that the proposed look-ahead heuristic outperforms the heuristic of Chang et al. (2004), which is better than all three modified heuristics of Yoon et al. (2011). Among modified three heuristics of Yoon et al. (2011), modified heuristic two is better than the others. (This part is not needed.)

Table 27 shows the comparison of the proposed look-ahead heuristic to the one-step look-ahead heuristic of Chang et al. (2004). Each cell of table 27 contains three values, for example, 1701, 100%, 44% means that in the considered 1701 problems (60 jobs on 6 machines), there are 1701 (100% of 1701 problems) problems where the proposed look-ahead heuristic gives better solution than the heuristic of Chang *et al*. (2004). The average improvement of these 1701 problems is 44%.

We also compare all heuristic and study the effect of each factor at different levels. Figure 13 shows the related trends: (1) The proposed look-ahead heuristic becomes more effective when the number of jobs is increased; (2) The proposed

look-ahead heuristic becomes more effective when the average of the setup time is longer; (3) The proposed look-ahead heuristic tends to give better solutions when a few jobs are available at time zero. (4) The proposed look-ahead heuristic performs better than Chang's heuristic when the ready time is relatively further to the due date. ($r-\tau$ is larger).

Both the proposed look-ahead heuristic and Chang's one step look-ahead heuristic use pairwise change technique to reduce the total weighted tardiness for the generated schedule on each machine. For Chang's method, if the three phase, pairwise exchange, is removed, the average result will increase 22.46% compared to the results of Change's three phase method (pairwise exchanged is used). In other words, the improvement of pairwise exchange in Chang's work is 22.46%. While in the proposed look-ahead heuristic (LAIPM), without using the pairwise exchange technique, the average of total weighted tardiness increases about 7.63% than using the pairwise exchange technique. From the experiment, we tell that if the phase of pairwise exchange is removed for both methods, LAIPM still yields better solution than Chang's method.

Table 26. Performance of look-ahead heuristics

Factor	LA(Chang)	H1(Yoon)	H2(Yoon)	H3(Yoon)	LAIPM
Jobs/mchine					
μ=10	1	5.55	4.23	4.57	0.55
μ=20	1	6.54	4.69	4.86	0.4
μ=30	1	6.32	4.53	4.66	0.33
Setup severity					
η=0.02	1	12. 72	6. 88	6. 96	0. 53
η=1.01	1	3. 58	3. 48	3. 79	0. 39
η=2	1	2. 10	3. 09	3. 33	0. 36
Due date tightness					
τ=0.3	1	12. 72	8. 83	9. 30	0. 30
τ=0.6	1	3. 58	2. 63	2. 79	0. 29
τ=0.9	1	2. 11	1. 99	2. 00	0. 69
Due date range					
R=0.25	1	3. 91	3. 87	3. 82	0. 43
R=0.63	1	6. 87	5. 00	5. 65	0. 41
R=1	1	7. 62	4. 59	4. 63	0. 45
Jobs availability					
Ja=0.2	1	6. 18	4. 40	4. 74	0. 53
Ja=0.5	1	7. 38	5. 34	5. 62	0. 42
Ja=0.8	1	4. 85	3. 72	3. 74	0. 34
Ready time tightness					
r-τ=1	1	5. 24	3. 92	4. 10	0. 52
r-τ=5.5	1	6. 50	4. 76	5. 00	0. 40
r-τ=10	1	6. 67	4. 77	4. 99	0. 37
Average	**1**	**6.14**	**4.48**	**4.70**	**0.43**

Table 27. Comparison (the proposed look-ahead heuristic to that of Chang et al. 2004)

6 machines	60 jobs	120 jobs	180 jobs
Better	1701,100%, 44.7%	1701,100%,60.27%	1701,100%,66.61%
Tie	0, 0%, 0%	0, 0%, 0%	0, 0%, 0%
Worse	0, 0%, 0%	0, 0%, 0%	0, 0%, 0%

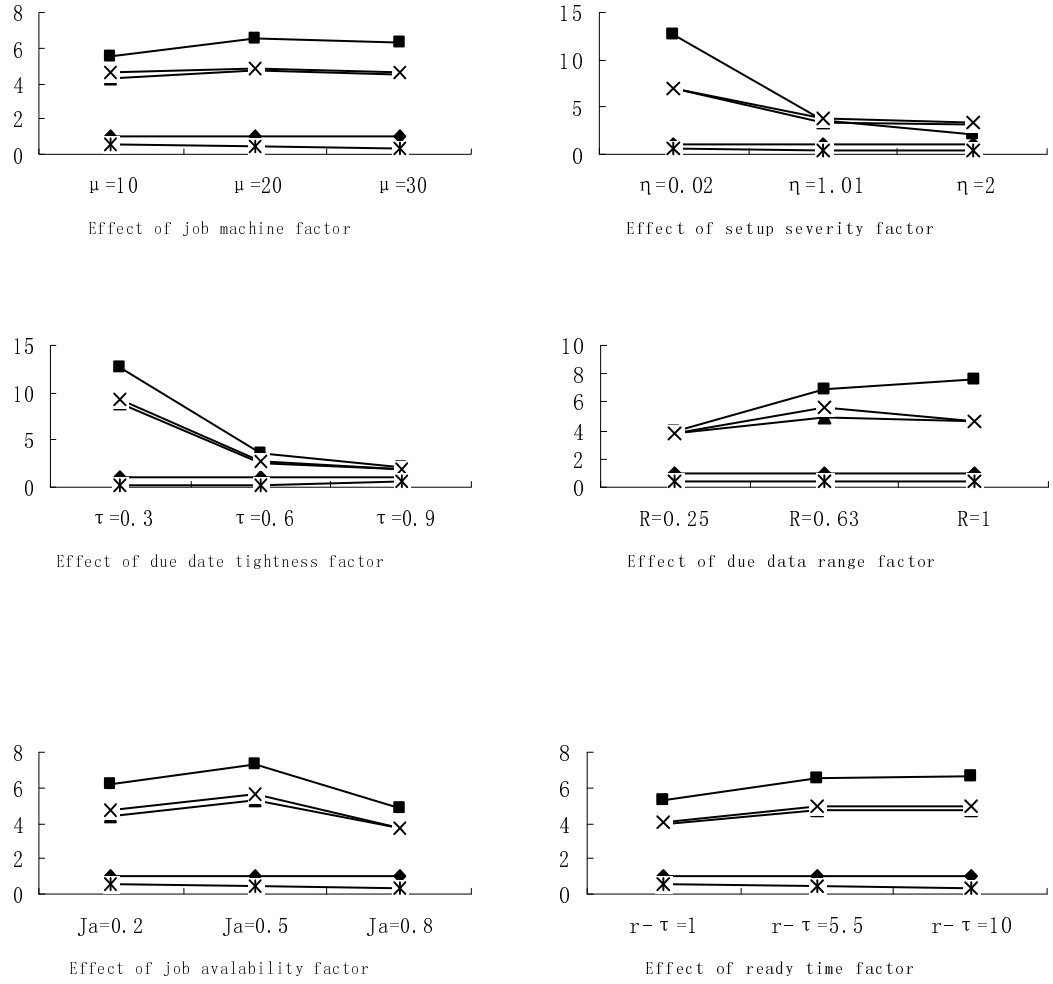


Figure 13. Comparison of factor effect of heuristics

5.7.2 LAIPM vs. non look-ahead heuristic (ATCRSR)

This section evaluates the performances of LAIMP and ATCTCR (the newly proposed ATC-based rule) over the generated 5103 problems in section 5.1. Each cell in table 28 contains three values: the number of better cases gained by LAIMP, the percentage of better cases over 1701 generated problems, and the average of tardiness reduction of the better cases (compared to the solutions of ATCRCS). It shows that

LAIPM (look-ahead) gives more numbers of better schedules than ATCRCS (none look-ahead), when the value of job machine factor increases. Also when the number of considered jobs is increased, the difference between these two methods becomes smaller and LAIPM gives more number of better cases with an increased tardiness reduction : $-(TWT_{LAIPM}-TWT_{ATCRCS})/TWT_{ATCRCS}$.

Table 28. Performance comparison (LAIMP vs. ATCRCS)

6 machines	60 jobs	120 jobs	180 Jobs
Better	827,48.62%,9.06%	944,55.5%,11.01%	1026,60.31%,12.29%
tie	7,0.41%,0%	4,0.23%,0%	0, 0%, 0%
Worse	867,50.97%.7.66%,	753,44.27%,5.2%	675,39.69%,4.47%

Table 28 shows the effect of the job machine factor. Table 29 to 33 shows the effect of the rest five factors: Setup severity, due date tightness, due date range, job availability, and ready time tightness. Form these tables, we notice that the number of better cases gained by LAIMP increases with increasing the number of considered jobs. This is the same at all levels of all factors, beside of the low level of due date tightness. At each level of the due date tightness, the number of better cases gained by LAIPM for 60 job, 120 jobs, and 180 jobs to ATCRCS is very close (288, 290, and 287 better cases respectively). It shows that due date tightness does not have significant effect for LAIPM to gain more number of better cases, compared to ATCRCS.

We also find that the setup severity factor has a significant effect to the results (LAIMP is compared with ATCRCS). Table 29 shows that LAIMP tends to give more number of better solutions than that of ATCRCS, when the value of setup severity factor is set at middle level (101% of the average processing time) and low level (2% of the average processing time). In the real life, the length of the setup is usually about

10% to 40% of the average processing time. This setup length falls in the range of the tested level: low setup level (2%) and the middle setup level (101%). Under this condition, we suggest using LAIMP for the identical parallel machines scheduling with sequence dependent setup and ready time, because it considers less number of jobs.

Table 29. Effect of setup severity (LAIMP vs. ATCRCS).

6m	60 jobs	120 jobs	180 Jobs
Low level	440,77.6%,8.02%	486,85.71%,11.11%	503,88.71%,13.28%
Middle level	234,41.27%,11.97%	255,44.97%,12.69%	315, 55.56%,12.28%
High level	153,26.98%,7.6%	203,35.8%,8.66%	208,36.68%,9,93%

Table 30. Effect of due date tightness (LAIMP vs. ATCRCS)

6m	60 jobs	120 jobs	180 Jobs
Low level	288,50.79%,10.81%	290, 51.15%, 9.89%	287, 50.62%, 10.26%
Middle level	368, 64.9%, 11.63%	451,79.54%, 15.82%	479%, 84.48%, 18.71%
High level	171,30.16%, 0.92%	203,35.8%,1.93%	260, 45.86%, 2.9%

Table 31. Effect of due date range (LAIMP vs. ATCRCS)

6m	60 jobs	120 jobs	180 Jobs
Low level	253, 44.62%,7.17%	280, 49.38%, 7.9%	296, 52.22%, 8.94%
Middle level	272, 47.97%,11.38%	311, 54.85%, 13.99%	363, 64.02%, 14.5%
High level	302, 53.26%, 8.75%	353, 62.26%, 10.85	367, 64.73%, 12.95%

Table 32. Effect of Jab availability (LAIMP vs. ATCRCS)

6m	60 jobs	120 jobs	180 Jobs
Low level	206, 36.33%, 6.56%	254, 44.80%,8.64%	274,48.32%,9.83%
Middle level	289, 50.97%, 8.88%	324, 57.14%, 10.93%	353,62.26%,12.35%
High level	332, 58.55%, 10.95%	366, 64.55%, 12.72%	399, 70.37%, 14.06%

Table 33. Effect of ready time tightness (LAIMP vs. ATCRCS)

6m	60 jobs	120 jobs	180 Jobs
Low level	240, 42.33%, 4.72%	255, 44.97%, 5.88%	272, 47.97%, 7.38%
Middle level	289, 50.97%, 9.63%	325, 57.32%, 11.18%	346, 61.02%, 12.38%
High level	298,52.56%, 12.2%	364,64.2%, 14.46%	408, 71.96%, 15.6%

Figure 14 shows the number of better cases gained by LAIPM at all levels of setup severity factor, due date tightness factor, due date range factor, job availability factor and ready time factor.

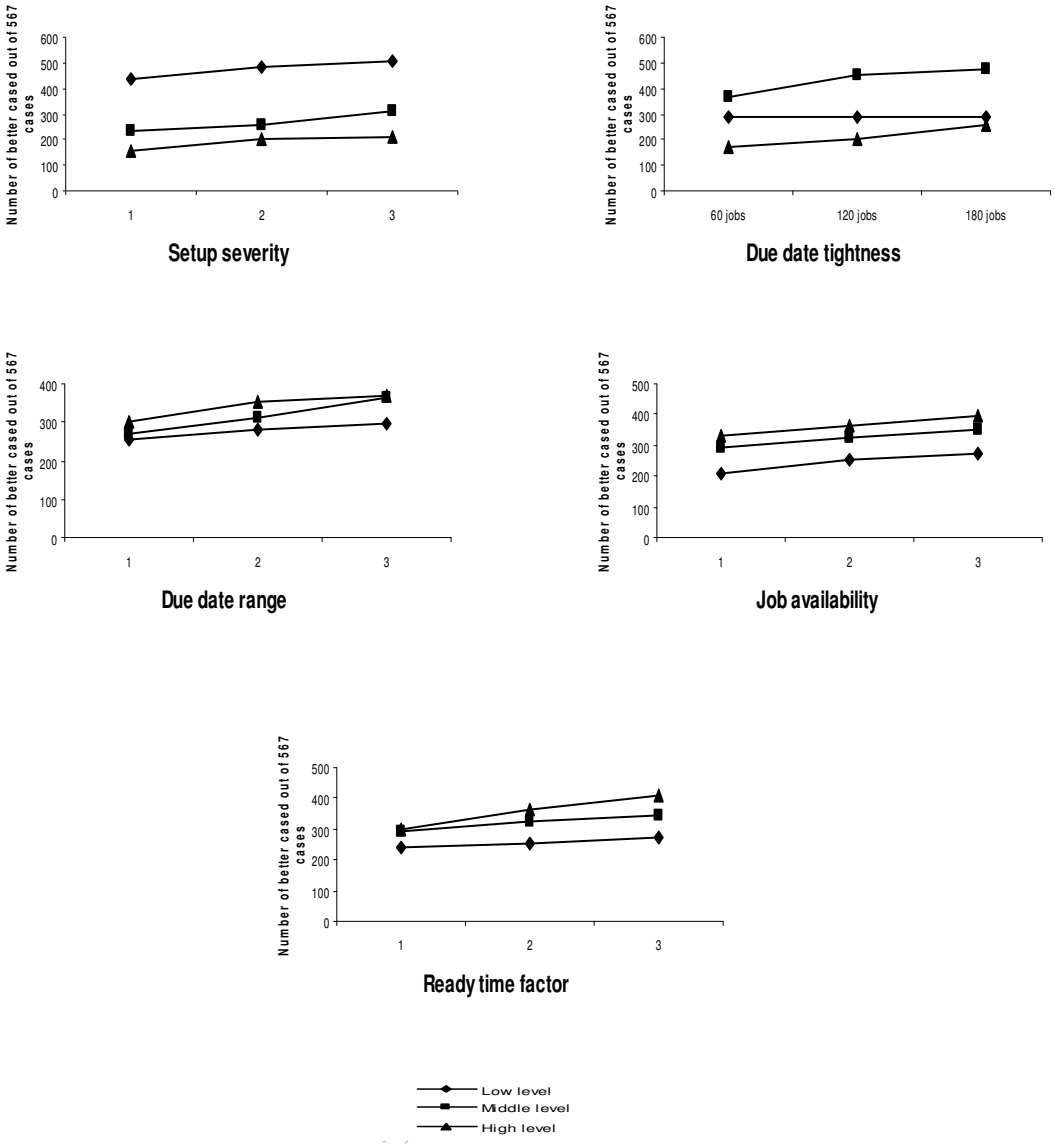


Figure 14. Number of better cases gained by LAIPM at different levels

6. CONCLUSIONS AND FUTURERESEARCH

This study is focused on minimizing the total weighted tardiness on the single machine and identical parallel machines. For the single machine scheduling, we analyze ATC-based dispatching rules and propose two new ATC-based rules, ATCRCS and ATCRSS. The performances of these two new rules are evaluated on the single machine and identical parallel machines. Experiments show that these new rules outperform other ATC-based rules in minimizing the total weighted tardiness in both single machine scheduling and parallel machines scheduling.

This paper also proposes a look-ahead heuristic (LAIPM) for the identical parallel machines scheduling with sequence dependent setup and future ready time. Twofold look-ahead is used to choose the next selected jobs on each machine. After all machines finish considering their next job selection, a possible iteration schedule is created. After that, the selected jobs in the considered iteration trigger the job switching heuristic to generate another possible iteration schedule. The schedule of the considered iteration is the better one.

Different with other look-ahead heuristic, like Change's one step look-ahead method, the job switching heuristic looks one-step back: each selected job have chance to be scheduled before the last scheduled job on each machines.

Experiment shows the proposed heuristic not only outperforms look-ahead heuristics by Chang *et al*. (2004) and modified heuristics by Yoon *et al*. (2011), but also gives better solution than ATCRCS which is a non look-ahead method. When

compared to ATCRCR, we notice that when setup severity is set at low level and middle level, LAIMP tends to gives better solution than ATCRCS.

For the future research, generating formulas to estimate values of scaling parameters in the index of ATCRCS is a good topic. It saves computation time and changes the search-based heuristic into an estimate-based heuristic. Besides of that, the proposed look-ahead heuristic may be modified and used in a more complex production environment where the parallel machines are unrelated or machine dependent.

7. REFERENCE

Allahverdi, A., and Mittenthal, J., (1994). Scheduling on M parallel machines subject to random breakdowns to minimize expected mean flow time. Naval Research Logistics, 41, 677-682

Anglani, A., Grieco, A., Guerriero, E., and Musmanno, R., (2005) Robust scheduling of parallel machines with sequence-dependent set-up cost. European journal of operational research, 161, 704-720

Anghinolfi, D., and Paolucci, M., (2008) A new ant colony optimization approach for the single machine total weighted tardiness scheduling problem. Int J Oper Res, 5: (1) 44–60

Arroyo, J. E. C., Nunes, G. V. P., and Kamke, E.H., (2009) Iterative local search heuristic for the single machine scheduling problem with sequence dependent setup times and due dates. Proceedings of the Ninth international conference on hybrid intelligent systems (Shenyang, China, Aug 2009), 104, 505-510

Arroyo, J. E. C., Rafael, D. S. O., Alcione, D. P. O., (2011) Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. Theo Comput Sci 281: 5-19

Asano, M., and Ohta, H., (1999) Scheduling with shutdowns and sequence dependent setup times. Int J Prod Res 37(7): 1661-1676

Azizoglu, M., and Webster, S., (1997) Scheduling job families about an unrestricted common due date on a single machine. Int J Prod Res 35: 1321–1330

Balakrishnan. N., John J. K., and Sri V. S,. (1999) Early/tardy scheduling with sequence dependent setups on uniform parallel machines. Computer & Operation Research 26 127-141

Chang T. Y., Chou F. D., and Lee C. E., (2004) A heuristic algorithm to minimize total weighted tardiness on a single machine with release dates and sequence-dependent setup times. Journal of the Chinese Institute of Industrial Engineers 21:289-300

Chen. J. F., (2009) Scheduling on unrelated parallel machines with sequence-andmachine dependent steup times and due date constraints. Int J Adv Manuf Technol 44:1204-1212

Chen Z. L., and Powell W. B., (2003) Exact algorithms for scheduling multiple families of jobs on parallel machines. Navel research logistics 50: 823-840

Christoph, H., Lars, M., and Jens, Z., (2007) Planning and scheduling Heuristic for parallel machines. Proceedings of the 2007 International Engineering Research Conference. Nashivill, Tenn, 1545-1550

Christos P. K., and Milton L. S., (1998) Look Ahead Scheduling for Minimizing Machine Interference. INT. J. PROD. RES, Vol. 26, No. 9, 1523-1533

Cicirello, V. A., and Smith, S. F., (2005) Enhancing stochastic search performance by value-biased randomization of heuristic. J Heuristics, 11: 5–34

Cigolini, R., Perona, N., Portioli, A., and Zambell, T., (2002) A New Dynamic Look-ahead Scheduling Procedure for Batching Machines. Journal of Scheduling 5: 185-204

Drieel, R., and Monch, L., (2009) Scheduling jobs on parallel machines with sequence dependent setup times, precedence constraints, and ready times using variable neighborhood search. IEEE, 273-278

Drieel, R., and Monch, L., (2011) Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constrains, and ready times. Computers and Industrial Enginnering 61,336-345

Eom, D. H., Shin, H., Kwun, I. H., Shim, J. K., and Kim, S. S., (2002) Scheduling jobs on parallel machines with dependent family set-up times. International journal of advanced manufacturing technology 19: 926-932

Felipe, R., (2005) Look-ahead constructive heuristic for the unrelated parallel machine problem with sequenced dependent setup time. IIE Annual Conference and Exposition 2005, 2005, IIE Annual Conference and Exposition

Feng, G., and Lau H. C., (2005) Efficient algorithm for machine scheduling problems with earliness and tardiness penalties. Proceedings of the 2nd multidisciplinary international conference on scheduling: theory and application, New York, USA, July 18-21, 196-211

Fowler, J. W., and Horng, S. M., (2003) A hybridized genetic algorithm to solve parallel machine scheduling problem with sequence dependent setup time. International journal of industrial engineering: theory application and practice 10, 232-243

Fowler, J. W., and Pfund, M., (2002) The Inclusion of Future Arrivals and Downstream Setups into Wafer Fabrication Batch processing Decision. Journal of Electronics Manufacturing, 11 (2) 149-159

Fowler, J. W., Phillips, D. T., and Hogg, G. L., (1992) Real-Time Control of Multiproduct Bulk-service Semiconductor Manufacturing Process. IEEE Transactions on Semiconductor Manufacturing. 5 (2)

Franca, P. M., Mendes, A., and Moscato. P., (2001) A memetic algorithm for the total tardiness single machine scheduling problem. Ero J Oper Res 132 (1): 224-242

Ginzburg, D. G., and Gonik. A., (1997) Using "Look Ahead" Techniques in Job-shop Scheduling with Random Operations. International. J. Production Economics (50) 13-22

Glassey, C. R., and Weng, W. W., (1991) Dynamic Batching Heuristic for Simultaneous Processing. IEEE Transactions on Semiconductor Manufacturing. 4 (2).

Guilher, M. E. V., Jeffrey, W. H., and Edward, L,. (2000) Analytical Models to Predict the Performance of a Single-Machine system Under Periodic and Event-Driven Rescheduling strategies. INT. J. PROD. RES., 2000, Vol.38, No.8, 1899-1915

Gupta, A. K., Ganesan, V. K., and Sivakumar, A. I., (2004) Hot Lot Management: Minimizing Cycle Time in Batch Process IEEE 0-7803-8519-5/04

Gupta, A. K., and Sivakumar, A. I., (2006) Optimization of Due-date Objectives in Scheduling Semiconductor Batch Manufacturing. International Journal of Machine tools & Manufacture 46 (2006) 1671-1679.

Gupta, A. K., and Sivakumar, A. I., (2007) Controlling Delivery Performance in Semiconductor Manufacturing Using Look Ahead Batching. International journal of Production Reseach, 45 (3): 591-613

Gupta, S. R., and Smith, J. S., (2006) Algorithms for single machine total tardiness scheduling with sequence dependent setups. Eru J Oper Res 175(2): 722-739

Heady, R. B., and Zhu Z., (1998) Minimizing the sum of job earliness and tardiness in a multi-machines system. Int J Prod Rese 1619-1632

Hepdogan, S., Moraga, R., Depuy, G. W., and Whitehouse, G. E., (2009) A Meta-Raps for the early/tardy single machine scheduling problem. Int J Prod Res 47(7): 1717-1732

Hirashi, K., Levner, E., and Vlach, M., (2002) Scheduling of parallel identical machines to maximize the weighted number of jobs of just-in tim jobs. Computers and operation research 29, 841-848

Ho, J. C., and Chang, Y. L., (1995) Minimizing the number of tardy jobs for m-parallel machines. European journal of operation research 84 343-355

Holthaus, O., and Ziegler, H., (1997) Improving Job shop Performance by Coordinating Dispatching Rule. INT. J. PPROD. RES., 35 (2): 539-549

Jang, J. J., Suh, J. D., Park, D and Liu, R., (2001) A Look-Ahead Routing Procedure for Machine Selection in a Highly Informative Manufacturing System. The International Journal of Flexible Manufacturing system 13: 287-308

Jihene, K., Chistophe, V., and Noureddine, Z., "Heuristics for Scheduling Maintenance and Production on a Single Machine"

Karp. R. M., (1972) Reducibility among combinational problems: complexty of computer computations. New York: Plenum press. P85-103.

Kim, D. W., Dong G., Na, F., and Chen. F., (2003) Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. Robotics and computer integrated manufacturing. 19: 173-181

Kim, D. W., Kim, K, H., Jang, W., and Chen, F., (2002) Unrelated parallel mchines scheduling with steup times using simulated annnealing. Robotics and computer-integrated manufacturing 18, 223-231

Kim, S. S., H. J., Eom, D. H., and Kim, C. O., (2002) A due date desity-based categorizing heuristic for parallel machines scheduling. International journal of advanced manufacturing technology 22, 753-760.

Kim, S. Y., Lee, Y. H., and Agnihotri, D., (1995) A hybrid approach to sequencing jobs using heuristic rules and neural network. Production Planning and Control, 6 (5), 445–454

Kirlik, G., and Oguz, C., (2012) A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. Comput Oper Res 39 (7): 1506 -1520

Kirlik,G., Kartal, Z., and Hasgul, S., (2012) A new crossover operator for single machine total weighted tardiness problem with sequence dependent setup times. Gazi University Journal of Science, 25 (1): 127-136

Kolahan, F., and Liang, M., (1998) An adaptive TS approach to JIT sequencing with variable processing times and sequence dependent setups. Eur J Oper Res 109(1):142-159

Krajewski, L. J., King, B.E., Ritzman, L.P., and Wong, D.S., (1987) Kaban, MRP and shaping the manufacturing environment. Management Science, 33, 39-57

Kurt, M. E., and Askin. (2001) Heuristic scheduling of parallel machines with sequence dependent set-up times. International journal of production research,39, 3747-3769

Lee, Y. H., Bhaskaran, K., Pinedo, M., “A heuristic to minimize the total weighted tardiness with sequence-dependent setups”1997, IIE Transaction, 29, 45-52.

Lee, Y. H., and Pinedo, M., “Scheduling jobs on parallel machines with sequence-dependent setup time” 1997, European journal of operational research 100, 464-474.

Liao, C. J., and Juan, H. C., (2007) An ant colony optimization for single machine tardiness scheduling with sequence dependent setups. Comput Oper Res, 34: 1899–1909

Liao, C., J., Tsou, H. H., Huang, K. L., (2012) Neighborhood search procedures for the single machine tardiness scheduling with sequence dependent setups. Ther Comput Sci 434: 45-52

Lin, S.W., and Ying, K.C., (2007) Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics. Int J Adv Manuf Tech, 34: 1183–1190

Lin, S. W, and Ying, K. C., (2008) A hybrid approach for single-machine tardiness problems with sequence-dependent setup times. Int J Adv Manuf Tech, 34: 1183-1190

Luo, X, Chu, C, and Wang, C (2006) Some dominance properties for single-machine tardiness problems with sequence-dependent setup. International Journal of Production and Research, 44 (17): 3367-3378

Mao, W., and Kincaid, R. K., (1994) A Look-Ahead Heuristic for scheduling Jobs with Release Date on a Single Machine. Computer Ops Res. 21 (10) 1041-1050

Mason, S. J, Fowler, J. W., and Carlyle, W. M., (2002) A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shop. Journal of Scheduling 5: 247-262

Mason, S. J, Fowler, J. W., and Carlyle, W. M., and Montgomery, D. C., (2005) Heuristics for minimizing total weighted tardiness in complex job shops. Int J Prod Res 43(10): 1943-1963

Morton, T. E, Rachamadugu, R.V., (1982) Myopic heuristics for the single machine weighted tardiness problem. http://www.ri.cmu.edu/pub_files/pub3/morton_thomas_f_1982 1/morton_thomas_f_1982_1.pdf

Morton, T.E, Pentico , D. W., (1993) Heuristic Scheduling Systems: With Applications to Production Systems and Project Management. Wiley, New York

Nekoiemehr, N., Moslehi, G., (2011) Minimizing the sum of maximum earliness and maximum tardiness in the single-machine scheduling problem with sequence-dependent setup time. J Oper Res Soc 62:1403-1412

Ovacik, I. M., Uzsoy, R., (1994) Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence dependent setup times. Int J Prod Res 32(6): 1243-1263

Panwalkar, SS, Dudek, RA, and Smith, ML (1973) Sequence research and the industrial scheduling problem. Proceedings of the Symposium on the theory of scheduling and it applications, 29-38

Park, Y., Kim,. and Lee. Y. H., (2000) Scheduling jobs on parallel machines applying neural network and heuristic rules. Computer & Industrial Engineering 38, 189-202

Pfund, M. E., Fowler, J. W., Gadkari, A., and Chen, Y., (2008) Scheduling jobs on parallel machines with setup times and ready time. Computers & Industrial Engineering, 54 (4), 764–782

Pinedo, M., (2002) Scheduling: Theory, Algorithm, and Systems, second ed., Prentice-Hall.

Rabadi, G., Mollaghasemi, M., and Anagnostopoulos, G. C., (2004) A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. Comput Oper Res 31(10): 1727-1751

Rachamadugu, R.V., and Morton, T. E., (1982) Myopic heuristics for the single machine weighted tardiness problem. Working Paper, Carnegie Mellon University, Pittsburgh, PA 30-82-83

Radhakrishnan and Ventura. (2000) Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. International Journal of production research. 38, 2233-2252

Raman, N., Rachamadugu, R. V., Talbot, F. B., (1989) Real time scheduling of an automated manufacturing center. Eur J Oper Res 40: 222-242

Robinson, J. K., and Fowler, J. W., (1995) The use of upstream and downstream information in scheduling semiconductor batch operations. INT. J. Prod. RES., 33 (7): 1849-1869

Sioud, A., Gravel, M., and Gagne, C., (2010) Constraint based scheduling in a genetic algorithm for the single machine scheduling problem with sequence-dependent setup times. In: Proceedings of the International Conference on Evolutionary Computation. 137-145

Smith, F. T. M., and Stecke, K. E., (1996) On the Robustness of using Balanced Part Mixed Ratios to Determine Cyclic Part input Sequences into Flexible Flow Systems. INT. J. PROD., 1996, Vol. 34, No. 10, 2925-2941.

Tan, K. C., and Narasimhan, R., (1997) Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach. Omega 25(6): 619-634

Tan, K. C., Narasimhan, R., Rubin, P. A., and Ragatz, G. L., (2000) A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times. Omega 28(3): 313-326

Tamimi, S. A., and Rajan, V. N., (1997) Reduction of total weighted tardiness on uniform machines with sequence dependent setups. Industrial engineering research-conference procedding, pp. 181-185.

Tunali, S., (1997) A Simulation Study Evaluating the Performance of a Look-ahead FMS Machine Scheduling Algorithm. Microcomputer Applications 16 (3)

Uzsoy, R., Lee, C. Y., and Martin-Vega, L. A., (1992) Scheduling semiconductor test operations. Minimizing maximum lateness and number of tardy jobs on a single machine. Nav Res Log 39 (3): 369-388

Van, D. Z., (2002) Adaptive Scheduling of Batch Server in Flow Shop, INT. J. PROD.RES, 40 (12): 2811-2833

Van, D. Z., Harten, A. V., and Schuur, P. C., (1997) Dynamic Job Heuristics for Multi-server Batch Operations-A Cost Based Approach. INT. J. PROD.RES., 35 (11): 3063-3093

Vepsalainen, A., and Morton, T. E., (1987) Priority rules for job shops with weighted tardiness costs. Management Science, 33, 1035–1047

Webster, S., Jog, P. D., and Gupta, A., (1998) A genetic algorithm for scheduling job families on a single machine with arbitrary earliness/tardiness penalties and an unrestricted common due date. Int J Prod Res 36. 2543–2551

Weng, M. X., Lu, J., and Ren, H. (2001) Unrelated parelllel machine scheduling with setup consideration and a total weighted completion time objective. International journal of production economics 70. 215-226

Weng, W. W., and Leachman, R. C., (1993) An Improved Methodology for Real-time Production Decision at Batch-Process Work Station. IEEE Transaction on Semiconductor Manufacturing. 6 (3)

Wodecki, M., (2008) A branch-and-bound parallel algorithm for single-machine total weighted tardiness problem. Int J Adv Manuf Tech 37: 996-1004

Xi, Y., and Jang, J. J., (2012) working paper "Minimizing Total Weighted Tardiness on the Single Machine with Sequence Dependent Setup Time and Future Ready Time" University of Wisconsin Milwaukee, IME department

Yoon, S. H, and Lee, I. S., (2011) New constructive heuristics for the total weighted tardiness problem. The Journal of the Operational Research Society 62 (1): 232-237

APPENDICES

Appendix A: A non-linear mathematic model (single machine)

C_j: Completion time of job j.

$p_{j,}$: Process time of job j.

$s_{i,j}$: Sequence dependent setup time to do job j after job i.

$s_{0,j}$: The initial setup time to do job j first on the machine.

$x_{i,j}$: 1, if job j is processed directly after job i, otherwise 0.

$x_{0,j}$: 1, if job j is the first job on the machine and 0 otherwise.

$x_{j,0}$: 1, if job j is the last job to be processed on the machine and 0 otherwise.

M: A large positive value.

d_j: Due date of job j.

T_j: Tardiness of job j.

w_j: Weight of job j.

r_j: Release time of job j.

Minimize

$$\sum_{j=1}^{n} w_j T_j \tag{1}$$

Subject to:

$$\sum_{i=0, i\neq j}^{n} x_{ij} = 1 \quad \forall \mathrm{j} = 1,...\mathrm{n} \tag{2}$$

$$\sum_{i=0, i\neq h}^{n} x_{i,h} - \sum_{j=0, j\neq h}^{n} x_{h,j} = 0 \quad \forall \mathrm{h} = 1,...,\mathrm{n} \tag{3}$$

$$C_j \geq \max(C_i, rj) + x_{ij}(S_{i,j} + P_j) + M(x_{ij} - 1) \qquad \forall \mathrm{i} = 0,...,\mathrm{n}, \forall \mathrm{j} = 1,...\mathrm{n}, \mathrm{i} \neq \mathrm{j} \tag{4}$$

$$\sum_{j=0}^{n} x_{0,j} = 1 \tag{5}$$

$$x_{i,j} \in \{0,1\} \quad \forall i = 0,...,n, \forall j = 0,...n, \forall k = 1,...,m \tag{6}$$

$$C_0 \geq 0 \tag{7}$$

$$C_j \geq 0 \qquad \forall j = 1,...n \tag{8}$$

$$T_j = \max(C_j - d_j, 0) \qquad \forall j = 1,...n \tag{9}$$

Formula (1) illustrates that the objective is to minimize total weighted tardiness. Constrain (2) ensures that each job is scheduled only once and processed on the machine. Constrain (3) makes sure that each job must not be proceeded or succeeded by more than one job. Constrain (4) deals with the relationship among completion time, release time and machine down time. This constraint also guarantees that no job can be preceded and succeeded the same one. Constraint (4) is a non linear constraint. Constrain (5) ensures that no more than one job can be scheduled first on the machine. Constrain (6) means that the decision variables are all binary. Constrain (7) is the completion time of the dummy job 0. Constrain (8) makes sure that all completion time must be positive values. Constrain (9) reflects the relationship of tardiness, completion time and due date for each job. It also guarantees that tardiness is a non-negative time. Above model can solve single machine total weighted tardiness problems with the sequence dependent setup time and unequal release times. This problem ($1|r_i,s_{ij}|\sum w_i T_i$) is also studied by Chang et al [17]. Compared to their mathematic model, our model uses less numbers of variables.

Appendix B: Lingo model for the proposed non-linear mathematic model (5 jobs)

```
sets:
jobs/0,1,2,3,4,5/:pt,due,ct,tar,weight,ti;
depstm(jobs,jobs):dstm;
links(jobs,jobs):x;
endsets

data:
dstm=
0    3    5    4    4    5
0    0    2    3    2    4
0    5    0    1    3    2
0    3    3    0    1    1
0    4    3    4    0    5
0    4    5    1    5    0;

pt= 0        5 8    4    2    8;

due=0         8     11   5    3    12
;
weight=0     3     7    4    4    7;

Ti=0      7     10   13   14   0;

ct=0,,,,,;

tar=0,,,,,;

enddata

min=@sum(jobs(i):tar(i)*weight(i));

@for(jobs(j)|j#gt#1:
        @sum(links(i,j)|i#ne#j:x(i,j))=1);

@for(jobs(h)|h#ge#2:
              @sum(links(i,h)|i#ne#h:x(i,h))-@sum(links(h,j)|j#ne#h:x(h,j))=0);

@for(jobs(i):
        @for(jobs(j)|j#ge#2#and#j#ne#i:
              ct(j)>=@if(ct(i)#ge#ti(j),ct(i),ti(j))+@sum(links(i,j):x(i,j)*
              (dstm(i,j)+pt(j)))+999*(@sum(links(i,j):x(i,j))-1)));

@sum(links(i,j)|i#eq#1#and#j#ne#i:x(i,j))=1;

@for(links:@bin(x));

@for(jobs(j)|j#ge#2:ct(j)>=0);
```

```
@for(jobs(i)|i#ge#2:
        tar(i)=@if(ct(i)#ge#due(i),ct(i)-due(i),0));
end
```

In the above model, dstm is the setup matrix which shows the setup between each pair of jobs. The row of pt, due, weight and Ti shows each job's processing time, due date, job weight and ready time. The optimal solution of above example is

J5-J3-J4-J2-J1.

The total weighted tardiness is 380.

Appendix C: A non-linear mathematic model (two parallel machines)

Minimize $\sum_{j=1}^{n} w_j T_j$

Subject to:

$$\sum_{i=0, i\neq j}^{n} \sum_{k=1}^{m} x_{i,j,k} = 1 \quad \forall \mathrm{j} = 1,...\mathrm{n} \qquad (2)$$

$$\sum_{i=0, i\neq h}^{n} x_{i,h,k} - \sum_{j=0, j\neq h}^{n} x_{h,j,k} = 0 \quad \forall \mathrm{h} = 1,...,\mathrm{n}, \forall \mathrm{k} = 1,...,\mathrm{m} \qquad (3)$$

$$C_j \geq Ci + \sum_{k=1}^{m} x_{i,j,k}(S_{i,j,k} + P_{j,k}) + M(\sum_{k=1}^{m} x_{i,j,k} - 1) \qquad \forall \mathrm{i} = 0,...,\mathrm{n}, \forall \mathrm{j} = 1,...\mathrm{n} \qquad (4)$$

$$\sum_{j=0}^{n} x_{0,j,k} = 1 \quad \forall \mathrm{k} = 1,...\mathrm{m} \qquad (5)$$

$$x_{i,j,k} \in \{0,1\} \quad \forall \mathrm{i} = 0,...,\mathrm{n}, \forall \mathrm{j} = 0,...\mathrm{n}, \forall \mathrm{k} = 1,...,\mathrm{m} \qquad (6)$$

$$C_0 = 0 \qquad (7)$$

$$C_j \geq 0 \qquad \forall \mathrm{j} = 1,...\mathrm{n} \qquad (8)$$

$$Tj \geq \max(Cj - dj, 0) \qquad \forall \mathrm{j} = 1,...\mathrm{n} \qquad (8)$$

Constrain (2) ensures that each job is scheduled only once and processed by one machine. Constrain (3) makes sure that each job must be neither be proceeded or succeeded by more than one job. Constrain (4) is used to calculate completion time and no job can precede and succeed the same job. Constrain (5) ensures that no more than one job can be scheduled first at each machine. Constrain (6) specifies that the decision variable is binary. Constrain (7) is the completion time of the dummy job 0. Constrain (8) makes sure that all completion time must be positive values. Constrain (9) ensures T_j is a non negative value.

Appendix D: Lingo model for the proposed non-linear mathematic model (5 jobs on 2 machines)

```
sets:
jobs/0,1,2,3,4,5/:pt,due,ct,tar,weight,ti;
machines/1,2/;
depstm(jobs,jobs):dstm1,dstm2;
links(jobs,jobs,machines):x;
endsets

data:
dstm1= 0 5 2 3 1 3
       0 0 3 8 6 2
       0 2 0 7 2 1
       0 5 3 0 5 3
       0 5 5 3 0 4
       0 6 2 1 7 0;

dstm2=  0 3 9 2 7 3
       0 0 2 4 5 2
       0 6 0 1 7 1
       0 3 5 0 4 3
       0 9 2 6 0 2
       0 3 1 4 6 0;

pt = 0 3 5 1 4 3;


due = 0 7 9 12 12 9;


weight = 0 2 4 8 5 2;


ti= 0 7 0 0 3 0;

ct=0,,,,,;

tar=0,,,,,;

enddata

!objective function;
min=@sum(jobs(i):tar(i)*weight(i));

!Constarint1: each job is scheduled once and on one machine;
@for(jobs(j)|j#gt#1:
      @sum(links(i,j,k)|i#ne#j:x(i,j,k))=1);

!constraint 2;
```

```
@for(jobs(h)|h#ge#2:
        @for(machines(k):
              @sum(links(i,h,k)|i#ne#h:x(i,h,k))-@sum(links(h,j,k)|j#ne#h:x(h,j,k))=0));


!constraint3. each job must be neither be proceeded or succeed by more than one job;
@for(jobs(i):
        @for(jobs(j)|j#ge#2#and#j#ne#i:
                ct(j)>=@if(ct(i)#ge#ti(j),ct(i),ti(j))+@sum(links(i,j,k)|k#eq#1:x(i,j,k)*(ds
                tm1(i,j)+pt(j)))+@sum(links(i,j,k)|k#eq#2:x(i,j,k)*(dstm2(i,j)+pt(j)))+99
                9*(@sum(links(i,j,k):x(i,j,k))-1)));

!constraint 4: only one job can be scheduled first;
@for(machines(k):
              @sum(links(i,j,k)|i#eq#1#and#j#ne#i:x(i,j,k))=1);



!constraint 5: binary constrain;
@for(links:@bin(x));


!constraint 6: cj is non-negative;

@for(jobs(j)|j#ge#2:ct(j)>=0);

!constraint 7: relation among tar, due and ct;
@for(jobs(i)|i#ge#2:
        tar(i)=@if(ct(i)#ge#due(i),ct(i)-due(i),0));


end
```

In the above model, dstm1 and dstm2 are the setup matrixes which shows the setup between each pair of jobs on machine 1 and machine 2. The row of pt, due, weight and ti shows each job's processing time, due date, job weight and ready time. The optimal solution of above example is:

M1: J2-J3

M2: J3-J5-J1

Total weighted tardiness is 21.

Appendix E: The application of proposed parallel machines model

In this section, we discuss the application of the proposed parallel machines model in Appendix C. With minor modifications, the proposed model in appendix C can solve other parallel machines scheduling problems.

Case 1: For the same example in appendix D, but using the below assumptions:

1. The processing time of a job is the same on all machines (same in appendix D).
2. All machines use the same setup time matrix (different in appendix D).

To solve this problem, we delete dstm1 matrix and change constraint 3 (the program in appendix 4) into:

```
@for(jobs(i):
    @for(jobs(j)|j#ge#2#and#j#ne#i:
        ct(j)>=@if(ct(i)#ge#ti(j),ct(i),ti(j))+@sum(links(i,j,k)|k#eq#1:x(i,j,k)*(dstm2
        (i,j)+pt(j)))+@sum(links(i,j,k)|k#eq#2:x(i,j,k)*(dstm2(i,j)+pt(j)))+999*(@su
        m(links(i,j,k):x(i,j,k))-1)));
```

The optimal solution for the considered problem is:

M1: J5-J2-J1

M2: J3-J4

The optimal solution (the total weighted tardiness is 40.

Case 2: For the same example in appendix D, but using the below assumptions:

1. The processing times of jobs are different on machines (different in appendix D).
2. Each machine has its own setup matrix (same in appendix D).

This case in literature is called “Machine dependency”. It also can be solved by the model in appendix C with minor modification:

1. Add another variable: pt2 (pt2 contains the processing times of all jobs on machine 2. The second sentence in the appendix D is changed into:

   jobs/0,1,2,3,4,5/:pt,due,ct,tar,weight,ti,**pt2**;

2. Add the processing times of all jobs. Put a new row: pt2 = 0 7 1 2 4 3; after pt = 0 3 5 1 4 3 in the program;

3. Change constraint 3 (the program in appendix 4 into:

```
@for(jobs(i):
    @for(jobs(j)|j#ge#2#and#j#ne#i:
        ct(j)>=@if(ct(i)#ge#ti(j),ct(i),ti(j))+@sum(links(i,j,k)|k#eq#1:x(i,j,k)*(dstm1
        (i,j)+pt(j)))+@sum(links(i,j,k)|k#eq#2:x(i,j,k)*(dstm2(i,j)+pt2(j)))+999*(@s
        um(links(i,j,k):x(i,j,k))-1)));
```

The optimal solution (total weighted tardiness) of the considered problem is:

M1: J4-J1

M2: J5-J2-J3

Total weighted tardiness is 18.

CURRICULUM VITAE

Name: Yue XI

Education:

B.A., Capital University of Economic & Business, June 2000
Major: Business Management

M.S., University of Wisconsin-Platteville, Dec 2004
Major: Industrial Technology Management

Ph.d., University of Wisconsin-Milwaukee, May 2013
Major: Industrial Manufacturing Engineering; Minor: Business Management

Dissertation Title: A look-ahead Heuristic to Minimize Total Weighted Tardiness on the Identical Parallel Machines with Sequence Dependent Setup and Future Ready Time.

Teaching & Research Experience:

Lab assistant, University of Wisconsin-Platteville, Jan-Dec 2004

Project assistant, University of Wisconsin-Milwaukee, Jan-May 2006

Teaching assistant, University of Wisconsin-Milwaukee, Sep 2007-May 2011

Publications (Selected):

Yue Xi and Jaejin Jang, 2012, Scheduling Jobs on Identical Parallel Machines with Sequence Dependent Setup Time and Future Ready Time: An Experimental Study", International Journal of Production and Economics, v 137, n 1, p 1-10.

Yue Xi and Jaejin Jang, 2013, Minimizing Total Weighted Tardiness on a Single Machine with Sequence Dependent Setup and Future Ready Time", Revision is accepted for publication on International Journal of Advance Manufacturing Technology on Dec 11th 2012