ADAPTIVE MULTI-PERIOD RESOURCE ALLOCATION

ZHAO ZHENGYI

(M.Sci Singapore MIT Alliance

M.Eng, B.Eng Tsinghua University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2016

Acknowledgements

I am full of gratitude to all my supervisors, Professor Shuzhi Sam Ge, Professor Lau Hoong Chuin and Professor Lee Tong Heng.

Professor Ge is the person always exciting me with his way. He teaches me the basic principle for PhD research. He leads me to fight with wisdom. He drives me to always struggle for the best.

Professor Lau is the person leading me into the world of scheduling, from basic shop scheduling to complicated resource constrained project scheduling. When I just started my PhD, I was luckily chosen by him to work as a research assistant in Singapore Management University. He gave much flexibility to my learning & working spaces. He further encouraged me to link scheduling and adaptive control and to exploit combinatorial areas.

I will give my special thanks to Professor Lee (NUS), Professor Sun Jie (NUS) Professor Dipti(NUS) and Professor Thin Yin (PSA-SMU). Special thanks to Dr. Kong Wei, Dr. Luo Ming and Dr. Zhang JingBing(SimTech), Michael (SMU) and Park JongHan, Prof. J.K. Lee (KAIST), Dr. Xiao Fei, Fu Na, Li Jia and Dr. Chih Fen. Special thanks to Dr. Fua Cheng-heng, Dr. S.H. Ling Steve, S.C.Lau, Ireneus and Kim JiYong.

I appreciate the patience of my family. I appreciate the encouragement from my father, who is deaf, but an outstanding professor in Harbin Engineering University. He ages over 70, but still teaching in his university. Finally solute to my grandfather (a great doctor) and my grandmother (my primary school teacher). Hope they rest in the heaven.

Contents

List of Tables

List of Figures

For the classical shop scheduling, we apply Pritsker's approach to get a 0-1 IP(Integer Programming) formulation. This formulation can be adjusted to problems with constraints either no-wait or infinite-wait-buffer. This formulation is further extended to handle (1) multi-machine problems (job-shop or flow-shop), (2)multi-period problems, (3) COS(Critical Operational Sequencing) constraints, (4) resource allocation problems.

For general multi-machine shop scheduling problem, a continuous time method is proposed to construct the machine utilization. This method is fundamental for the whole thesis, it is used in the following aspects:

- to compare the 2 optimal criterion for multi-machine job-shop problems;
- to build machine capacity relaxation-based heuristics (CH:Constructive Heuristic and RH:Repair Heuristic) for flow-shop, bi-directional flow-shop with or without COS constraints;
- to construct the auction's approach in resource allocation among several scheduling agents;

For the multi-machine job shop problem with infinite wait buffer, we proposed a heuristic method to get a fast feasible solution. The solution quality is compared with CPLEX solution with IP formulation. This heuristic is extendable to handle dynamic batch-job arrival.

For the multi-machine Bi-directional flow shop with COS constraints, 2 heuristics (CH and RH)are proposed and compared with existing greedy method and CPLEX solution. Although CH and RH give better solution than Greedy, the computational time is substantially longer. But if compared with the ScheGen-IP solution, the run time for all heuristic methods are relatively much shorter. The heuristic method works

fine for pure **Forward Flow Jobs** with small variance in processing time, when it could achieve real optimal solutions.

The COS constraints not only has meaningful applications, but also provides a new point theoretically. Even the job-sequence has been fixed, the heuristic solutions still make a lot of difference from the optimal one. This is a counter-example for such point by the researchers of genetic algorithm — optimal sequence will result in optimal solution.

The integrated resource allocation is done among several flow-shop scheduling agents, in multi-machine multi-period environment. Our contribution is an adaptive tatonnement auction scheme that comprises two key ideas for price adjustment: the concept of utility pricing and variable step size, along with an algorithm for performing bid generation. Combined with the pre-processing and post-processing steps, the power of the entire system is demonstrated in solving large-scale decentralized scheduling problems.

On utility pricing, we demonstrated both analytically and experimentally that it has a better convergence property than conventional price adjustment schemes with bidding price only. What's more, bidding with utility price does not depend on initial prices, although the cost is an increased communication overhead between auctioneer and bidders. On variable step size, we show experimentally that it performs better than fixed step size with respect to both solution quality and speed, and it is insensitive to the underlying speed function parameters.

Chapter 1

Introduction

The title of this thesis is "Adaptive Multi-period Resource Allocation". More exactly, it is to allocate resources among several scheduling agents. The resources are discrete machine-hours. Each agent is responsible to schedule for a job-list, which is rooted from classical shop scheduling.

The scheduling problem has a long history while it is still in the state of the art. The art of scheduling is how to make best utilization of time and resources. Although time is evenly given to every one, the best scheduler can make fully utilization of time and resources.

As said by Benjamin Franklin: *Dost thou love life? Then do not squander time, for that the stuff life is made of.* There is another well saying by Mark Twain: *Time and tide wait for no man.*

Among all types of scheduling problems, the focus of this thesis is shop scheduling (including job-shop and flow-shop). From the broadest viewpoint, we are interested in the following questions:

1. **How do we model resource utilization through the time horizon?** What

primitives should we utilize in these models? Can we develop models which are flexible as well as computationally tractable?

2. **How do we decompose a large problem into several smaller problems for the computational tractability?** Especially in the real-time operational environment, how do we allocate resources among several competing scheduling agents?

In this thesis, the resources are some discrete machines. There are multiple exchangeable machines in a production plant, transportation port or etc. The time can be either discrete or continuous in the horizon, while we find the model in continuous time is more flexible and useful in the multi-agent systems s.t. different agents may have different time units.

1.1 Layout and Contribution of this Thesis

The layout of this thesis is shown in Fig. 1.1.

Basically, this thesis studies multi-machine non-preemptive scheduling and multi-period resource allocation. Chapter 2 (*Chap*-2) studies job-shop scheduling, flow-shop scheduling, while bi-directional flow-shop and Critical Operational Sequencing (COS) is a model abstracted from the operations in a container terminal port. In this chapter, a classical 0-1 integer formulation is used for CPLEX solution to benchmark the proposed heuristic solutions. The general 0-1 formulation considers **Non-Preemptive** constraint, **Precedence** constraint and **Machine Capacity** constraint, while some special constraints as COS will be discussed specifically.

Chapter 3 (*Chap*-3) further studies the dynamic issues, a list of dynamics are specified

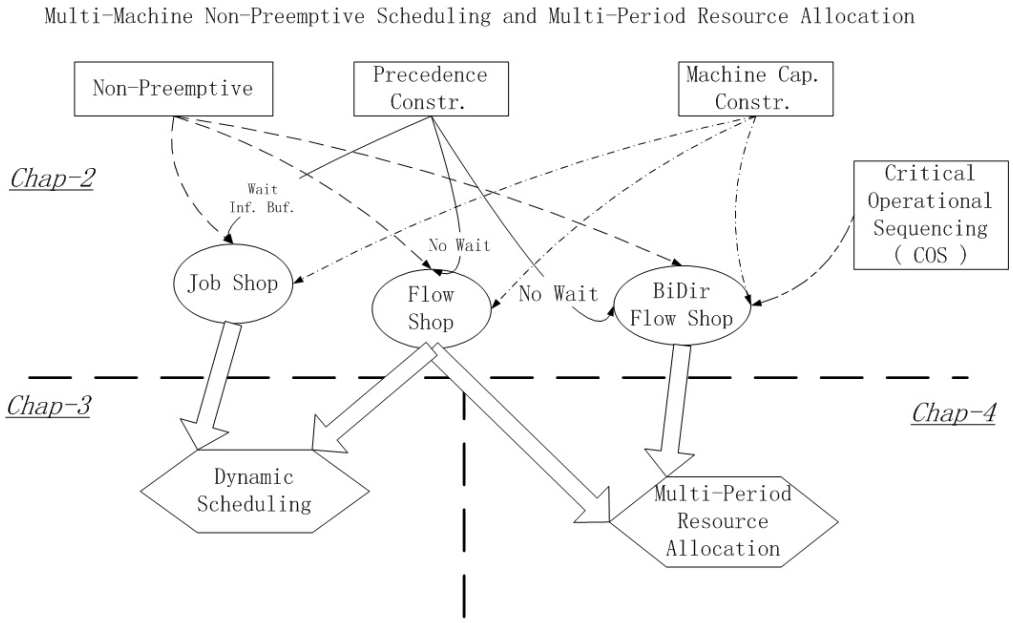


Fig. 1.1: Layout of this thesis

and we extend the algorithms in Chapter 2 to handle some dynamics.

In Chapter 4 (*Chap*-4), we study multi-period resource allocation problem. The resource is defined as some quantity of machine-hour and allocation is among several agents, each of whom is in charge of a job-list of flow-shop and bi-directional flow-shop.

Some contribution points are high-lighted as follows.

- Chapter 2 studies scheduling problems for both job-shop and flow-shop. Our contribution is a model in the continuous time domain and the related heuristic methods. This model uses the δ function for a close-form formulation. Some heuristic methods are proposed to solve job-shop and flow-shop scheduling problems. RH(Repair Heuristic) and CH(Constructive Heuristic) are heuristics for bi-directional multi-machine no-wait flow-shop scheduling. IMM(Iterative Minimization Micro-model) is for multi-machine job-shop scheduling with infinite wait buffer.

- In Chapter 3, we actually extend the above heuristics to handle some kinds of

dynamics. The extended IMM heuristic is to handle batch job arrival. We highlight the contribution of the circular charts, which are studied in comparison with the rectangular charts. We also extend the RH and CH to handle machine de-gradation / up-gradation problems. The implementation is explained with consideration of fast computation.

- Chapter 4 studies static resource allocation problem. Our contributions are the concept of utility price and an adaptive price adjustment strategy. The utility price is studied both in theory and by numerical experiments. The fast bid-generation algorithm is based on previous study of CH and greedy heuristics. Combined with the pre-processing and post-processing steps, the power of the entire system (Adaptive Multi-period Resource Allocation) is demonstrated in solving large-scale decentralized scheduling problems.

In one word, this thesis is trying to solve the scheduling and resource allocation problems with view point of control engineer and by ways of adaptive control methodology.

1.2 Literature Review

This thesis will cover static scheduling, dynamic scheduling and resource allocation among several scheduling agents. This section will present preliminary works in the following aspects,

- static and dynamic shop scheduling, and
- auction and Lagrangian relaxation.

1.2.1 Preliminary for shop scheduling

The shop scheduling problem was originated from the operational scheduling in a workshop. Now, the similar model has been widely used in many areas including production, transportation, construction [14], computer network, hospital operations and etc. In 1910s, Henry Gantt developed Gantt charts [24] for major infrastructure projects including the Hoover Dam and Interstate highway system. In 1960s, Muth and Thompson proposed the famous MT10 problem [110], one problem instance is so hard as just to be solved after 20 years [90].

This kind of problem is usually NP-hard to solve [100]. It belongs to the category of combinatorial optimization [83].

Solutions for the classical job shop (flow shop) problems may be classified into the following categories:

1. Methods based on local search or heuristics, i.e. branch-and-bound [61, 66], beam-search [64], tabu-search [3, 28], simulated-annealing [49, 53, 90],

2. Methods based on global search genetic algorithms [13, 18, 26, 51], and

3. Integer programming and Lagrangian relaxation or auction approach [17, 42].

The flow shop scheduling problem has been addressed since early 1970s researchers, [100, 108] and etc. The extended models with no wait in operation (or No Intermediate Storage for the machine, called Flow Job with No Intermediate Storage FSNIS) have been studied by Panwalkar [96], who proposed a heuristic solution for a special kind of FSNIS problems with ordered property in processing time. The general problem of FSNIS is NP-hard to solve [98]. The n-job FSNIS can be transformed to $(n + 1)$ city traveling salesman problem(TSP) [104].

The early works before 1990s do not handle the problem with multiple machine constraints. Multi-machine capacity constraints may be studied in one of the following cases:

Case 1: There are multiple identical machines C_k for each machine type k, and each specific machine could perform no more than one operation at any time; and

Case 2: There is exactly one machine for each type k, and there can be as many as C_k operations done on each machine at any time, and $C_k \geq 1$, and

Case 3: Combination of **Case 1** and **Case 2**.

Case 1 is a usual assumption in discrete manufacturing systems. One sample of **Case 2** is multi-task handling of an operation system in any personal computer, while **Case 3** can be found in more flexible systems. In my research, we study the problems associated with **Case 1**, i.e. a semi-conductor assembly line and modern container terminal port.

Note that there is a significant difference for problems with $C_k = 1$ and with $C_k \geq 2$. The former can be formulated as disjunctive graph [102], which could be easily mapped to mixed integer programming model, while the latter has so far not yet nicely formulated in my opinion, except that in mid-2000s, Kutanoglu [42] applied the Pritsker's formulation [109] and solved a MIP problem with Lagrangian relaxation approaches. And actually the author still focused on the problems with $C_k = 1$, although Pritsker's formulation could cover $C_k \geq 2$. Pritsker's formulation is huge in problem size because in discrete time domain, each time slot of every operation needs one 0-1 variable to represent. In this thesis, a formulation in continuous time domain is proposed, focusing on implementation of machine capacity functions, and compared with the Pritsker's formulation in the discrete time domain.

Besides the features of multi-machine, we also studied multi-period problems, which

means that machine resources are dividable into time-slots.

Given that job scheduling is computationally hard to solve, it is even harder to consider resource allocation and operational scheduling jointly [44]. Almost no literature gives an integrated model. By partitioning machine resources to Multi-Period (each has fixed number of time-slots pre-defined as problem input), this thesis extended Pritsker's 0-1 formulation [109] to handle resource allocation among multiple agents, each of whom is handling a list of shop-scheduling problem, which is both multi-machine and multi-period. The formulation is used by CPLEX solver and bench-marked with the solution by auction.

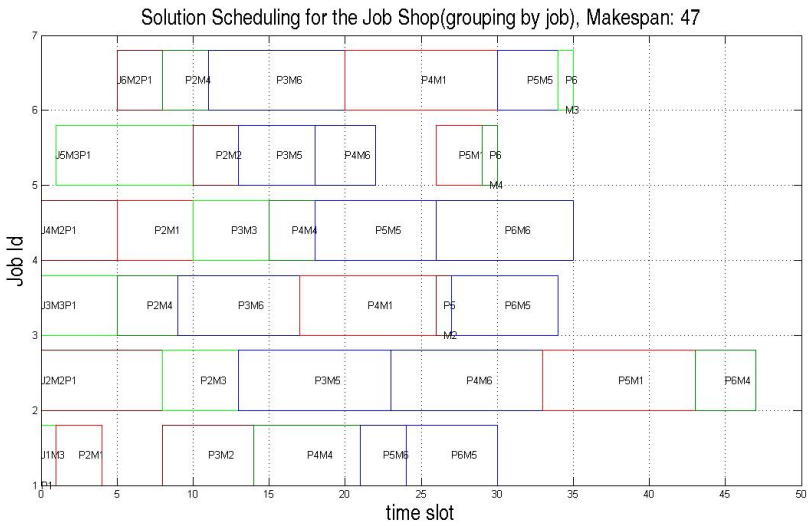
1.2.2 Briefing for Shop Scheduling by Gantt Chart

In the literature, Gantt chart is usually used to represent a schedule solution.

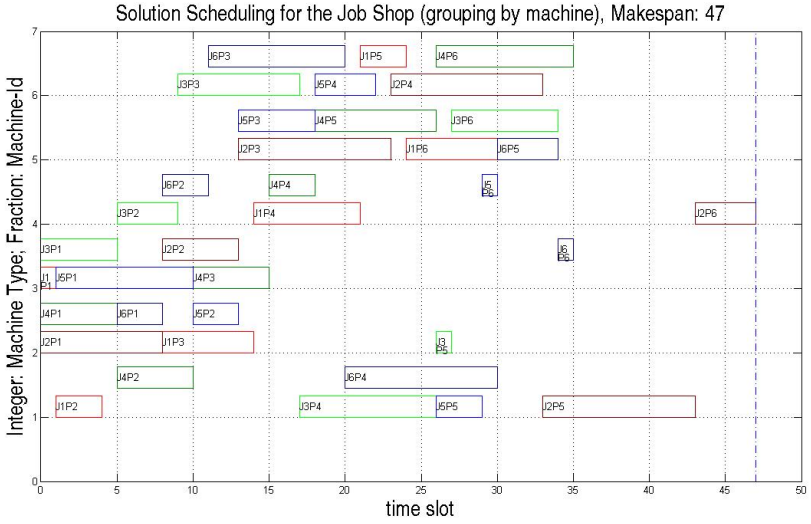
Fig. 1.2 shows a classical job-shop scheduling. From Fig. 1.2-(a), there are 6 jobs, whose id is represented by y-axis. Each job has exactly 6 tasks by 6 different types of machines. The job-task-machine configuration is, in fact, just the same as **MT6** problem [110].

A different feature in this thesis is that, there are 2 exchangeable machines for each type. Let us see Fig. 1.2-(b), which is machine schedule. The integer part of y-axis is machine-type, while the fractional part is machine-id.

Fig. 1.3 shows typical charts for a multi-machine bi-direction flow-shop scheduling. Fig. 1.3-(a) is machine schedule and Fig. 1.3-(b) is job schedule. From machine-schedule in Fig. 1.3-(a), please notice that it is, again, a multi-machine problem. Machine type-1 has 1, machine type-2 has 4 items and machine type-3 has 2 items. From job-schedule in Fig. 1.3-(b), there are no more than 3 processes for each job. There is no wait between

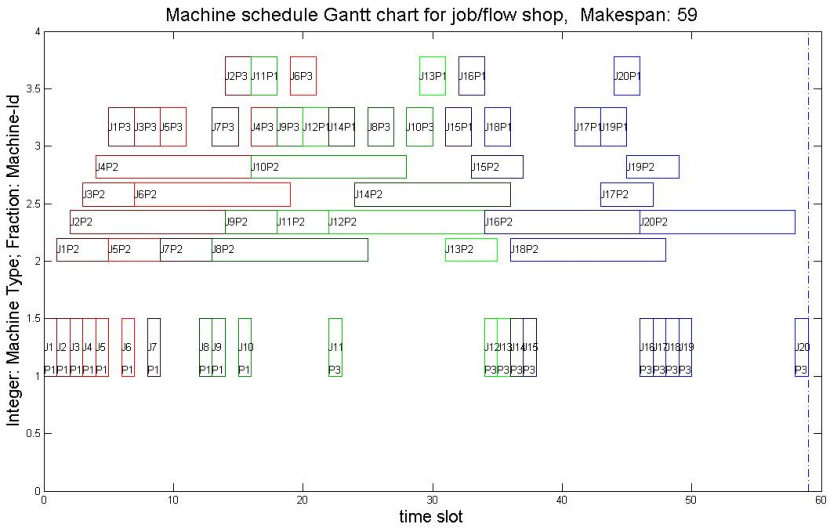


(a) schedule (by job)

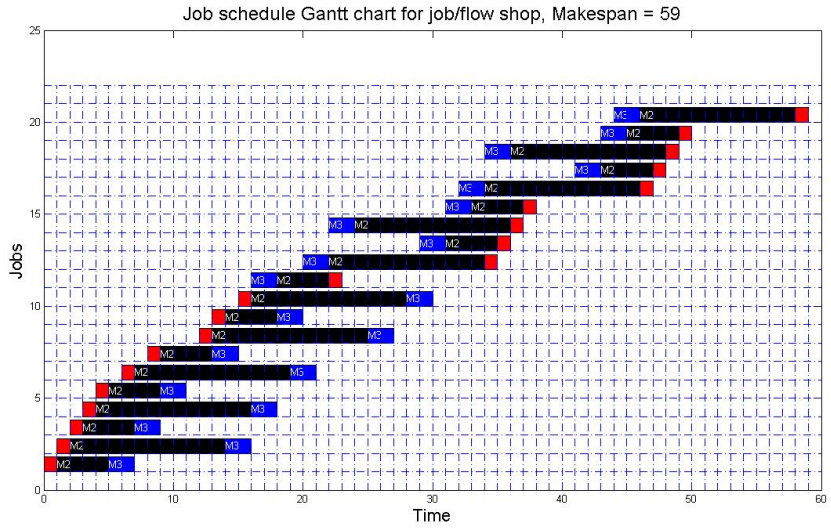


(b) schedule (by machine type (integer) and ID(fraction))

Fig. 1.2: Typical Gantt chart for multi-machine job-shop schedule



(a) schedule (by machine type (integer) and ID(fraction))



(b) schedule (by job)

Fig. 1.3: Typical Gantt chart for multi-machine bi-directional flow-shop scheduling

consecutive processes, i.e. once the previous process completed, the following will start immediately thereafter.

With the help of charts in Fig. 1.2, we give the definitions for job-shop and flow-shop in this thesis.

- There is a list of jobs and each job has a number of tasks, which are carried out in a pre-defined sequence.
- Tasks are defined according to the machine-type in use. The same task uses the same type of machine, while different tasks use different types.
- All the tasks are non-preemptive. Once it is started, it should be carried out to the end without interruption.
- One machine can handle no more than one task at any time. For each type of machine, there are multiple and exchangeable items.
- For **Job-shop** scheduling in the thesis, the task sequence in different jobs may not be the same. And a period of wait time is allowed between consecutive tasks.
- For **Flow-shop** scheduling in the thesis, all the jobs have exactly the same task sequence. There is no wait allowed between consecutive tasks.
- What's more for this thesis, we define the Bi-Direction Flow-shop. The task sequence for all the jobs is either **forward** or **reverse** but no others.

In our definition of shop-scheduling, the task and process mean the same.

The **Job-shop** model is found to be useful in semiconductor production-line. Each job is a kind of chip to be produced. The different chips have different processes so the task-sequences for producing different chips may be different. A period of wait time

is allowed between consecutive tasks. During this wait period, the chips are usually temporarily stored in the so-called "magazines", and keeping such magazines doesnot cost so much.

For the **Bi-Direction Flow-shop**, It is a model abstracted from a typical operation in a container terminal port, where machine-1 is quay crane, machine-2 is truck (or mobile engines, robots and etc.), machine-3 is yard-crane. For this particular chart, there is so-called Critical Operational Sequencing (COS) constraint, which will be further explained in later Section 2.3. From job-schedule in Fig. 1.3-(b), there are 20 jobs. First 10 jobs are called forward-jobs, their operations are carried out on machine-1, machine-2 and machine-3 consecutively. The following 10 jobs are reverse-jobs for their operations are carried out on machine-3, machine-2 and machine-1.

1.2.3 From static scheduling to dynamic scheduling

Research on dynamic scheduling started ever since late 1970s [101], when Nelson proposed a centralized scheduling heuristic for a dynamic job shop. The dynamics only cover the processing time variation. In 1990s, this topic is growing more interesting as more publications can be found. Li [80] suggested more dynamic factors for rescheduling. He proposed a rescheduling method based on binary tree construction. Sabuncuoglu [45, 58, 64] proposed the term reactive scheduling on comparison with predictive scheduling.

Predictive scheduling is to generate schedule which determines planned start and completion times of job's operations. Reactive scheduling is for dynamic monitoring execution of the schedule and to cope with unexpected events [58].

Generally speaking, people studied static and dynamic scheduling for 3 generations

till now, each with different focuses;

- Theoretical study, analysis on computational complexity, one machine problem. See publications until early 1980s i.e. [97, 101, 106];
- Software-based simulation and algorithm comparison, since late 1980s to 1990s. See [65, 75, 76, 79–82, 84, 85, 89, 94];
- System design and hardware implementation, problems with multiple machines (robots, automated guided vehicles i.e. AGV). Some publications from 1990s till now are following [12, 23, 25, 31, 32, 34, 36, 37, 40, 41, 44, 45, 48].

Following Sabuncuoglu's classification, we further add more recent works into the tables in Fig.1.1 - 1.4. The original Sabuncuoglu's concept in 1990s is for FMS(Flexible Manufacturing Systems). It is adaptive to the following aspects in 2000s,

- RCPSP(Resource Constrained Project Scheduling Problem) in [21, 30];
- transportation problems, i.e. railway time-table in [29], container-terminal in [22, 34];
- robotic cell in manufacturing in [32]; and
- actual or conceptual mobile robots or AGVs, [12, 23, 36, 37].

We have seen the trends of rescheduling to more flexible environment, as technology keeps developing and more tools are available, i.e. MSN(Mobile Sensor Network) in [23], vision, recognition and etc. There will be more topologies, i.e. negotiations among agents [31, 50]. There will be even more dynamics, i.e. complete and incomplete communications, complete and incomplete information (or knowledge) in [16].

Table 1.1: Dynamic scheduling research in 1970s to early 1990s

Year	Author	Environment			Predictive Schedule generation		Reactive schedule	
		Shop Floor	Job arrival	Stochasticity	Method	Objective function (default to minimize)	When	How schedule
1974	Holloway Nelson	Job shop	Static	Process time variation (var.)	HSP	Tardiness rel. perform. M.	Initial once	Initial Full
1977	Nelson et al.	Job shop	Dynamic	Process time var.	HSP	Tardiness rel. perform. M.	Periodic	Full new schedule
1979	Farn and Muhleman	Single machine	Dynamic	No	DR & Heuristics based on TSP	Changeover time	Periodic	Full new
1982	Muhleman et al.	Job shop	Dynamic	Machine break., Process time var.	Dispatch rules	FT, MT, PL, CMT	Periodic	Full new
1985	Yamamoto and Nof	Job shop	Static	Machine breakdown	Branch and Branch (B&B)	Makespan	Event driven (MB)	Full new
1989	Wu and Wysk	FMS	Dynamic	No	Dispatching rules	Mean tardiness, Mean Flow time	Periodic	Partial of simulation window
1990	Dutta	FMS (Flexible Manufacturing Systems)	Static	Breakdown, New jobs, Change in job priority	KB	Mean completion time, Mean machine utilization (max)	Event driven	Rerouting, Preemption, etc.
1991	Kiran et al.	FMS	Static - dynamic	No No	Multipass heuristic	Tardiness rel. perform. M.	None periodic	Full new
1991	Bean et al.	Multiple resource	Static	Mach. break., Unavail. Tool	MUSA	Weighted total tard.	Event driven	Repair
1991	Nof and Grant	Small cell	Static	Machine break. Unexpected order arrival	Several	Performance based,	Periodic	Full new, right shift, rerouting to alter. mac.
1992	Church and Uzsoy	Single machine	Dynamic	No	EDD	L max	Periodic Event driven	Full new (urgent jobs)

Table 1.2: Dynamic scheduling research in 1990s

Year	Author	Environment			Predictive Schedule generation		Reactive schedule	
		Shop Floor	Job arrival	Stochasticity	Method	Objective function (default to minimize)	When	How schedule
1993	Matsuura et al.	Job shop	Semi-dynamic	Mach. Break, Rush jobs, Specif. change,	B&B, FCFS, SPT	Makespan	After first disruption	Full Job selection
1993	Li Rong-Kwei et al.	Job shop	Dynamic	Machine break, Process time variation	Binary tree construction	Makespan	Event-driven	Partial reschedule
1994	Ovacik and Uzsoy	Single machine	Dynamic	No	Algorithm based on B&B	L max	After scheduling some jobs	Partial
1994	Kim and Kim	FMS	Semi-dynamic	Machine break , Urgent job	Dispatch rules	mean FT T combination	Periodic Event driven	Full new
1994	Bengu	Flowline	Dynamic	Machine breakdown	ATC	Mean weighted tardiness	No	Job selection
1995	Wu and Li	job shop	Dynamic	mach.break. New jobs	graph construction	Makespan	Event-driven	Partial
1997	Sabuncuoglu and Karabuk	FMS	Static	Machine break. Process time variation	Beam search and dispatch rule	Mean tardiness and makespan	Periodic	reschedule Full new
1997	Lawrence and Sewell	Job shop	Static	Processing times	Optimum and heuristic methods	Makespan	Operation completion	Full new
1998	Akturk and Gorgulu	Modified flow line	Static	Machine break.	RHSA	Earliness and tardiness	Event driven	Repair
1998	Mehta and Uzsoy	job shop flow shop	Dynamic	machine breakdown	Shift bottle-neck	Lateness Earliness	Periodic	Full new
1998 1998	Byeon, Wu and Storer	job shop	static	No	ATC heuristic weighted	tardiness	Event-driven	Partial
1999	Bierwirth and Mattfeld	job shop	Dynamic	new job,	genetic algorithm	makespan	Event-driven	Full new

Table 1.3: Dynamic scheduling in 2000s, classical concept

Year	Author	Environment			Predictive Schedule generation		Reactive schedule	
		Shop Floor	Job arrival	Stochasticity	Method	Objective function (default to minimize)	When	How schedule
2000a	Vieira, Herrmann and Lin	job shop	Dynamic	inter-job time, job-size, setup time	FIFO	aver. flow time, MAX mach. Utilization	Periodic, hybrid event-driven	Full new
2000b	Vieira, Herrmann and Lin	job shop parallel mach.	Dynamic	job arrival, machine breakdown,	FIFO	MAX machine utilization	Periodic event-driven	Right-shift
2003	Sabuncuoglu	FMS	Dynamic	6 factors (Q, SF,RF, TF, PV, e)	3-layer system (simulator, controller, scheduler)	mean flow time	multiple	full and partial
2004	Shabtay,D. Kaspi,M	single machine	Static	processing time	IP	total weighted flowtime	initial once	5 algorithms
2006	Shabtay, D. Kaspi, M.	identical parallel machines, nonrenewable resource	staic	Controllable processing time	IP: assign jobs sequencing , allocate the resource	makespan or the sum of completion times	initial once	
2007	Shabtay, D.	no-wait 2-machine flow shop	staic	Resource dependent time processing time	IP-¿ TSP with permuted Monge matrix	convex resource consumption function	every cycle	4 heuristics 3 algorithms
2008	Safia Kedad-S. and etc.	parallel machine	static	due dates, earliness tardiness costs	Assignment and time-indexed IP	earliness-tardiness	initial once	Lagrangian relaxation + local search
2008	Ruslan Sadykov	job shop single machine	static	no	brounch and cut	weighted number of late jobs	initial once	check feasibility
2008	Heidemarie B. and etc.	preemptive open shop	static	job number ¿ = ¡ machine number,	heuristic	mean flow time	initial once	4 heuristics
2009	A. Turkcan1, M. S. Akturk and R. H. Storer	parallel-machine	static,	stability measure for dynamic Controllable processing time	time-indexed IP model	total weighted processing time earliness + tardiness	initial once once measure	Linear programming relaxation

Table 1.4: Dynamic scheduling in 2000s, new domain and technology

Year	Author	Environment			Predictive Schedule generation		Reactive schedule	
		Shop Floor	Job arrival	Stochasticity	Method	Objective function (default to minimize)	When	How schedule
2003	Bish and etc.	multiple cranes in port	static	number of jobs	List Scheduling (greedy)	makespan	initial once	LS
2005	Ada Che, C. Chu	2 robots cyclic flowshop	static in one cycle	no	heuristic	cycle time (r jobs)	every cycle	full new
2005	Fua S.S.GE	Multiple robots	dynamic	Robot malfunction task uncertainty	IP: assignment Greedy	number of robots for the task, MAX total suitability	event driven	COBOS (Greedy)
2006	P. Vansteenwegen D. V. Oudheusden	railway timetables	hourly daily cyclic	delay and passenger variation	Linear programming	weighted sum of waiting times and late arrivals	initial once	no
2006	Wong, T.N.	FMS	Dynamic	parts arrival among part-agents and machine-agents	negotiation	makespan and flowtime	event driven	Re-negotiation
2007	Fua, S.S.GE and KW Lim	Multiple Robots	static and dynamic	delay and task failure	greedy	roam space population	event driven	greedy
2007	Fua, S.S.GE and KW Lim	Multiple robots	dynamic	formation move , and change in local communication	Lyapunov function	Formation error	event driven	Q-formation scheme
2007	S. V. D. Vonder and etc.	resource-constrained project scheduling	static and dynamic	delivery dates	compare 3 methods	earliness and tardiness	event driven	partial and full compare 4 methods

The Table 1.1 presents research inclusively before early 1990s.

The Table 1.2 presents related publications from late 1990s to early 2000s.

The Table 1.3 presents related publications in 2000s, which belongs to conventional shop-scheduling domain.

The Table 1.4 presents related publications in 2000s, while either scheduling is in new domains or new techniques are applied.

Especially for the robotic application, we would like to address conceptual difference in conventional reactive scheduling and feedback control. While both are used to take action when actual feedback (or schedule execution) departs from expect set-point, they are different such that

- Reactive scheduling will change the schedule for unexecuted tasks, as defined by Sabuncuoglu [58]; while
- Feedback control will apply action (force, power or etc.) to change the controllable object and maintain the set-point, as in Fua and Sam's work [16].

As the development of robots, there will be more and more conceptual changes in the conventional knowledge. The basic inter-relationship between human-being (man/woman) and robots (machine) keeps unchanged, while continuous efforts will be put into following research fields

- how robots can behave like human-being, w.r.t. intelligence and adaptiveness;
- how human-being can behave like robots, w.r.t. efficiency and power; and last but not least,
- how to inter-act among human-being and robots, including (1) inter-act among

human-beings, (2) inter-act among robots, (3) from human-beings to robots and (4) from robots to human-being.

1.2.4 Preliminary for resource allocation by auction and Lagrangian relaxation

The resource allocation, in this thesis, is defined to allocate machine-hours (i.e. vehicles, robots and etc) among several scheduling agents. This thesis focuses on static problems. A method, called *auction*, is used because of its high efficiency and the natural linkage with Lagrangian relaxation.

In the literature, auctions have been widely used as mechanisms for coordination among agents over multiple periods. Wellman [56] for example, introduced auction mechanisms which used prices derived through distributed bidding protocols to determine job schedules, and investigated the existence of equilibrium prices for some general classes of scheduling problems. In other works such as Gagliano's [74] and Kutanoglu's [63], similar ideas were proposed to solve other resource allocation problems. In these works, the price equilibrium is achieved by adjusting prices for all the resources iteratively as the auction proceeds, which is done through a *tatonnement* process that resolves resource conflicts by updating the price based on excess demands. Tatonnement was originally proposed in computational economics [112], and the major factor in the computational efficiency of an auction mechanism hinges on the speed of convergence of this process.

Auctions over multiple periods have been studied in other scheduling contexts. For instance, in electricity markets, the pricing model in Toczywski's work [2] is designed for a centralized pool-based auction which involves solving the unit commitment optimization problem under competition that balances production offers (sell bids) with demand over

horizon of several periods of time. Recently, Confessore [15] considers decentralized multi-project scheduling problems, where different projects are managed by local decision makers and a coordination mechanism is proposed to resolve shared resource allocation conflicts between different projects.

The computational efficiency issue surrounding auctions has been studied ([68, 71, 95]), and in particular, Fisher [93] proposed price adjustment process based on the sub-gradient search method.

Unfortunately, in all above works, the prices are often determined primarily and simplistically by the supply-demand gap, or by the profit gap between buyer and seller. In my research, careful study shows how the standard scheme can be augmented in two ways.

Firstly, if the bidders will be given the opportunity to also provide signal to the auctioneer) their *marginal utility* values, and together with the market (i.e. aggregate demand and supply), faster convergence can be achieved with better quality solutions. Intuitively, the faster price equilibrium is because the auctioneer is now able to adjust prices not only by observing the aggregate demand and supply discrepancies, but also the *individual* marginal utilities information signalled by the agents which give a sense of the sensitivity of individual objectives (and therefore demands) with respect to price changes. It is assumed that agents truthfully report the marginal utility values. So, the proposed price adjustment strategy is called price adjustment with utility pricing (termed as PAU), which is compared with the conventional price adjustment strategy based on bid prices alone (PAB).

Secondly, an adaptive scheme is proposed such that the step size is adjusted from one iteration to the next. This will improve the speed of obtaining the first feasible solution.

The adaptivity of the step-size follows closely the concept of control theory applied in robotics motion control. These two extensions give rise to what is called an "adaptive auction" mechanism.

Before proceeding to detailed issues in scheduling, we would like to list the terminologies in Tables from 1.5 to 1.6.

Table 1.5: List of notations in shop scheduling problem

Symbol	Description
Constraints	
f_{PREEM}	Preemptive constraints
f_{OPRECE}	Equality constraints for operation precedence
$g_{MACAP}(X)$	Machine capacity constraints in discrete time formulation
$g_{MACAP}(t)$	Machine capacity constraints in continuous time formulation
g_{COS}	Job dependency constraints
Notation continuous time and discrete time constraints	
Symbol	Description
Notation for Machines	
$M_k(t)$	Machine utilization function in continuous time Each k indicates a specific machine type
$\tilde{T}$	Total time slot in discrete time formulation
M_{kt}	Machine utilization function in discrete time Each k indicates a specific machine type Each t indicates a positive time slot $k \in \{1, ..., K\}, t \in \{1, 2, ..., \tilde{T}\}$
Decision Variable	
X_{ijt}	Discrete decision variable for i^{th} job, j^{th} operation at time slot t
T^s_{ij}	Start time of operation j in job i
T^e_{ij}	Completion time of operation j in job i
T_{MS}	Makespan completion time of last critical operation
Agents	
L	Total number of agents/job-lists
$\mathcal{T}$	Total number of time periods
R^l	Release time of job-list l
D^l	Due time of job-list l
W^l_d	Delay penalty of job list l
W^l_m	Makespan price of job list l
MTC	Sum of weighted makespan and tardiness costs
$\mathcal{SC}$	Sum of weighted costs including makespan, tardiness and machine costs
$\mathcal{RC}^l$	Resource cost for agent-l

Table 1.6: List of notations in resource allocation

Symbol	Description
Resources	
K	Total number of machine types
k	Machine type index, $k \in \{1, ..., K\}$
τ	time period index, $\tau \in \{1, ..., \mathcal{T}\}$
B_k^l	Base-line allocation of machine type k for agent l
$U_{k,\tau}^l$	Utilization by agent l for machine type k at period τ
$\mathbf{U}^l$	Bid by agent l expressed as $\{U_{1,1}^l, \cdots, U_{K,1}^l, \cdots U_{1,\mathcal{T}}^l, \cdots, U_{K,\mathcal{T}}^l\}$
$\mathcal{U}^l$	Bid space for agent l, where $\mathbf{U}^l \in \mathcal{U}^l \subseteq \mathcal{N}^{K\mathcal{T}}$
$\mathcal{M}(\mathbf{U}^l)$	Makespan for job-list l achieved using resources given in bid $\mathbf{U}^l$
$\mathcal{M}(\mathcal{U}^l)$	Makespan matrix for job-list l through its bid space $\mathcal{U}^l$
$S_{k\tau}$	Overall supply resource k at period τ
C_k^M	or C_k Capacity of machine type k, $C_k^M = max_\tau S_{k\tau}$,
$D_{k\tau}$	Overall demand for resource k at period τ
Price	
$u_{k\tau}^l$	Utility price by agent l for machine k at time period τ
$p_{k\tau}^r$	Bidding price of machine k for time period τ at auction iteration r
Job-list Information	
N^l	Total number of jobs in job-list l
o_i^l	Total number of operations for job i in job-list l
t_{ij}^l	Processing time of job i operation j in job-list l $i \in \{1, ..., N^l\}$ and $j \in \{1, ..., o_i^l\}$
m_{ij}	Mapping from job i and operation j to machine type
IP Formulation: Variables and Constraints	
X_{ijt}^l	discrete decision variable for job i operation j at time slot t
$X_{0,t}^l$	Dummy variable for job list starting
$X_{N+1,t}^l$	Dummy variable for job list completion
Symbol in Micro-Model in IMM Heuristic	
t_m^r	time point for a particular machine m to be ready to start a new task.
t_l^a	time point for task l to be ready to start.
$p_l, 1 \leq l \leq L_k(R)$	processing time for task l
$L_k(R)$	total num. of schedule-able task for machine type k in R's iteration
$mi(l)$	1-to-1 mapping from the task-id $l \in \{1, 2, ..., L_k(R)\}$ to the machine-id $m = mi(l) \in \{1, 2, ..., C_k\}$ for machine type k
R	iteration number

1.3 Some Basic Issues in Scheduling Problems

Although Scheduling problems are studied for many decades, they are still in the state of the art as there are so many issues. They are basic, though complicated for their inter-dependency.

Scheduling problems deal with time and time depends on a lot of factors.

- Processing time depends on machine capability, the different kinds of machines (i.e. different brands or manufacturers) may differ so great in the processing speed;
- Even for the exactly same machine, processing time may depend on the running status, i.e. machine setting and parameters;
- Processing time may also change when machine environment variate, i.e. variation in temperature, humidity, power supply and etc.;
- Processing time also depends on task or mission requirement.

For the manufacturing, a job means making a part, an equipment or etc. The job may need several fabrications in a fixed procedure. The requirement on the final part means a list of industrial specifications, i.e. packaging, operational conditions, failure rate and etc. The different specifications may correspond to different process or different procedures and so there will be different processing time.

For the transportation, a job means delivering some goods. Processing time may depends on the routing and traffic conditions.

One big difference between time scheduling and space planning is that scheduling doesnot have triangular inequality which is a basic constraints in space and geometry problems.

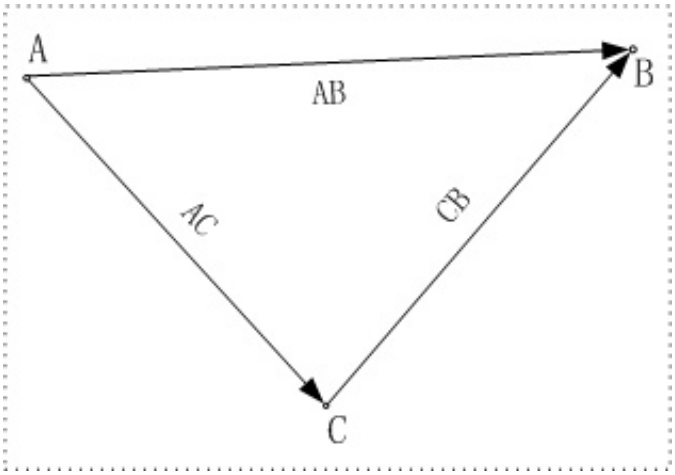


Fig. 1.4: Demo. Triangular Inequality

As shown in Fig. 1.4, there is a graph with 3 vertexes {A, B, C}and 3 edges {AB, BC, AC}. The graph may represent a geometric or space planning problem. If the length of edge is just representing the distance of 3 places, there will be the triangular inequality

$$||AC| - |CB|| \leq |AB| \leq |AC| + |CB|$$

The graph may also illustrate a scheduling problem, i.e. to deliver some good from A to B directly or to deliver it via point C. If the length of each edge represents its time, above triangular inequality may not exist. For example, traveling from A directly to B may take more time than traveling via C, only because there is a high-way from C to B.

Most scheduling and resource allocation problems are non-linear and time-variant.

If only consider the machine resource as input and the production-rate as output, the scheduling system is not a linear system. Because of some special constraints like operational precedence and etc., adding the number of machines does not mean adding the speed of processing. Figure 2.1 is an example showing that adding resource after some level cannot further reduce the makespan.

For in-house fabrication and assembly, the temperature and power supply is relatively fixed and the processing time can be time-invariant. However, in cases of transportation, construction and port-operations, the whether may affect processing time greatly.

Although scheduling problem can be much more complex, in this thesis we only consider some constraints and dynamics. Some fundamental assumptions for this thesis are listed as below:

- Non-Preemptive tasks: one task is done by one type of machine, once the task is started, it will carry on to the end;
- Multiple exchangeable machines for every type;
- Task processing time is pre-defined as problem input;
- Resource means machine-hour, i.e. machine availability in multiple periods; and
- Machine setup time are not considered, i.e. it can be ignored compared with processing time.

Resource allocation means allocating resources among several agents, each is to schedule a job-list.

Chapter 2

Multi-Machine Non-Preemptive Shop Scheduling

A job (or a project) is made up of several processes (or tasks), which have pre-defined sequences. Each process (or task) is done by a kind of machine (or resource), and the capacity of machine is limited.

This chapter studies scheduling problems for both job-shop and flow-shop. Our contribution is a model in the continuous time domain. This model uses the δ function for a close-form formulation, which is detailed in Section 2.2.1. Some heuristic methods are proposed to solve job-shop and flow-shop scheduling problems. RH(Repair Heuristic) and CH(Constructive Heuristic), in Section 2.3, are heuristics for bi-direction multi-machine no-wait flow-shop scheduling. IMM(Iterative Minimization Micro-model), in Section 2.5, is for multi-machine job-shop scheduling with infinite wait buffer.

2.1 Mathematical Formulation in Scheduling Problems

We will study two kinds of formulations, one in discrete time domain and the other in continuous time domain. The formulation in discrete time domain was originally proposed by Pritsker [109] and recently studied by Kutanoglu [42]. Such formulation needs a proper time unit $U_T \in R$, with which the continuous time domain is discretized to totally $\tilde{T} \in N$ time slots. This estimation is introduced [63], and the total time period $U_T\tilde{T}$ could be available by heuristic solution.

In such application as multiple agents scheduling and resource allocation, different agents may have different time units. This will lead to the formulation in continuous time domain.

2.1.1 Decision variables

In discrete time domain, $t \in \{1, 2, ..., \tilde{T}\}$ is used to represent each time slot. A binary decision variable is X_{ijt}, where

$$X_{ijt} = \begin{cases} 1 & \text{if Operation j in job i starts by time t inclusively;} \\ 0 & \text{if Operation j in job i has not yet started time at t .} \end{cases}$$

In continuous time domain, the following variables are used for decision.

T^s_{ij}: start time of operation j in job i;

T^e_{ij}: completion time of operation j in job i;

One presumption is that all tasks are non-preemptive. In discrete time domain, it is formulated as follows (2.1).

$$X_{ijt} - X_{i,j,t+1} \leq 0, \forall i, j, t \in \{1, 2, ..., \tilde{T} - 1\}. \tag{2.1}$$

$$X_{i,j,t} - X_{i,j-1,t-p_{i,j-1}} \left\{ \begin{array}{ll} = 0 & \text{no wait} \\ \leq 0 & \text{with wait} \end{array} \right\} \forall i, j \in \{2, ..., o_i\}. \tag{2.3}$$

$$T^e_{i,j-1} - T^s_{ij} \left\{ \begin{array}{ll} = 0 & \text{no wait} \\ \leq 0 & \text{with wait} \end{array} \right\}, \forall i, j \geq 2. \tag{2.4}$$

While the formulation in continuous time domain is (2.2).

$$T^s_{ij} - T^e_{ij} + p_{ij} \quad = \quad 0, \forall i, j. \tag{2.2}$$

2.1.2 Constraints

For most scheduling problems, there are the following constraints to be considered,

- Task sequence constraint, and
- Machine capacity constraint.

For specific applications, there may be other constraints, i.e. project start time, job-sequence constraints and etc.

In the task sequence constraint, there are 2 scenarios, there is wait buffer or no wait buffer between consecutive processes. It turns out that the solution methods are greatly different between these 2 scenarios. But in mathematical formulation, they only differ as equality sign and inequality sign. The formulation in equation (2.3) is discrete time domain while corresponding continuous time formulation is in (2.4).

In the machine capacity constraints, we start with a simple case which assumes that the machine capacity is a constant throughout the scheduling horizon. That is C_k for machine type k. Correspondingly, formulation (2.5) is in discrete time and (2.6) is in continuous time.

$$g_{MACAP}(X) = \sum_{i,j:m_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > p_{ij} & (X_{ijt} - X_{i,j,t-p_{ij}}) \\ \text{if } t \leq p_{ij} & X_{ijt} \end{array} \right\} - C_k \leq 0, \forall k \in \{1, 2, ..., K\} \quad (2.5)$$

$$g_{MACAP}(t) = M_k(t) - C_k \quad \leq \quad 0, \forall t > 0, k \in \{1, 2, ..., K\} \quad (2.6)$$

The machine capacity constraint in continuous time (2.6) has a so-called **Machine Utilization Function**, which is implemented with the δ function as follows.

$$M_k(t) = \sum_{m_{ij}=k} \int_0^t [\delta(t - T_{ij}^s) - \delta(t - T_{ij}^e)] \, dt \quad (2.7)$$

In continuous time domain $t \in R$, $\delta(t - T)$ is a positive infinite pulse at time point defined by parameter T:

$$\delta(t - T) = \left\{ \begin{array}{ll} 0 & \forall t \in (-\infty, T) \cup (T, \infty) \\ \infty & \text{if } t = T \end{array} \right.$$

And the energy (or area) of the function $\delta(t-T)$ with axis t is unit step up at time point $t = T$.

$$\int_{-\infty}^{t} \delta(t - T) \, dt = \left\{ \begin{array}{ll} 1 & \text{if } t > T, \\ 0 & \text{if } t < T. \end{array} \right.$$

Note that this function is not continuous in nature, which makes the machine utilization function still theoretically unsolved yet. However, a look-up table based method will be proposed later to implement the above machine utilization function, which is sufficient to generate the machine capacity constraint relaxation to solve the problem by heuristic methods.

In the following sections, if not specially noted, $\forall i, j, t$ will stands for $\forall i \in \{1, 2, ..., N\}, j \in \{1, 2, ..., o_i\}, t \in \{1, 2, ..., \tilde{T}\}$

2.1.3 Objective functions

There are usually two types of criterions *CRI-1* and *CRI-2*:

CRI-1: The objective is to minimize the makespan, which by nature, is to minimize the time length from the starting time of very first task to completion time of the very last task.

CRI-2: The objective is to minimize the sum of weighted tardiness, which actually, is to minimize some areas.

Their formulations are following:

CRI-1: For a general job-shop problem, two types of dummy variables $\{X_{S,t}, X_{E,t} : t = 1, 2, ..., \tilde{T}\}$ are introduced to formulate the *Start* time of very first task, and the *End* time of the very last task. See (2.8) and the inequality constraints (2.9) and (2.10).

$$\text{minimize} \sum_t (1 - X_{E,t}) \quad - \quad \sum_t (1 - X_{S,t}) \tag{2.8}$$

$$X_{i,j,t} - X_{S,t} \quad \leq \quad 0, \forall i, j, t \tag{2.9}$$

$$X_{E,t} - X_{i,j,t} \quad \leq \quad 0, \forall i, j, t \tag{2.10}$$

CRI-2: The most generalized formulation introduces two lists of parameters, $W_i : i = 1, 2, ..., N$ for each job's weight (or importance) and $d_i : i = 1, 2, ..., N$ for each job's due time. For a special kind of flow shop with some sequencing constraints like [22], by adjusting the parameter W_i, (2.36) represents either minimizing the makespan or minimizing the sum of weighted tardiness.

$$\text{minimize} \sum_{i} (W_i \sum_{t > d_i - p_{i,o_i}} (1 - X_{i,o_i,t})) \tag{2.11}$$

In this chapter, we study 2 different kinds of problems.

Prob-1: The 1st problem is multi-machine bi-directional no-wait flow-shop scheduling with COS constraints. This problem is actually abstracted from the operation of a modern container terminal port. The objective function of this problem is the combination of makespan and weighted tardiness.

Prob-2: The 2nd problem is multi-machine job-shop scheduling with infinite wait buffer. This problem is actually extended from classical one-machine job-shop scheduling, while it is found to be applicable for the operation of a semiconductor plant. The objective function of this problem is the sum of weighted tardiness. This objective function is used to compare our IMM heuristic with integer programming methods and solution by CPLEX. Besides, we make further comparison on the solution quality between IMM heuristic and ProModel (a commercial planning and scheduling software). The solution quality includes more specifications, s.t. {Mean Machine-Release Time, Mean Job-Completion Time, Mean Machine-Utilization, Total Number of Wait, Max Wait Time, Mean Wait Time}.

2.2 Machine Utilization Function and Capacity Formulation

It is a key issue to construct the machine utilization function both in discrete time and in continuous time, because the function will be used in heuristic methods as well as in

the Lagrangian relaxation procedure.

2.2.1 Utilization function in continuous time domain

The continuous-time formulation uses the $\delta(t)$ function, which is only available in close form formulas. It is equivalent to the following window functions.

$$
\begin{aligned}
M_k(t) &= \sum_{m_{ij}=k} w_{ij}(t) \\
w_{ij}(t) &= \begin{cases} 0 & \text{if } t \le T^s_{ij} \\ 1 & \text{if } T^s_{ij} < t \le T^e_{ij} \\ 0 & \text{if } t > T^e_{ij} \end{cases}
\end{aligned}
$$

In terms of implementation, a look-up table is generated given a set of schedule $\{(T^s_{ij}, T^e_{ij}) | i = 1, ..., N; j = 1, ..., o_i\}$. For each machine type k, first form a cell matrix of 2 by 2 for each job i and operation j whose machine type is k. Then for each type of machine, a set $\mathcal{TC}^{se}_k$ is constructed which contains all such cell matrix related with $m_{ij} = k$.

$$
\mathcal{TC}^{se}_k = \left\{ \left[\begin{pmatrix} T^s_{ij} \\ 1 \end{pmatrix}, \begin{pmatrix} T^e_{ij} \\ -1 \end{pmatrix} \right] : \right\}
1 \le i \le N, 1 \le j \le o_i \text{ and } m_{ij} = k \tag{2.12}
$$

It is called *Timing Cell* matrix with the following properties:

- It is a group of 2-by-2 matrix can also be viewed as a 2-by-2 mapping, from time to pulse;
- The start time T^s_{ij} maps to a unit positive pulse 1, while the end time T^e_{ij} maps to a unit negative pulse -1;

- One group is related on machine-type k, s.t. $m_{ij} = k$.

Considering all the machine types $k \in \{1, 2, ..., K\}$, the total number of above cell matrices is $\sum_{i\geq 1}^{i\leq N} o_i$.

Specially for flow shop, there is $o_i = K, \forall 1 \leq i \leq N$, and the total number of cell matrices for each type of machine is exactly N. For normal job shop problems sometimes with redundant machines or operations, one has to count $\sum_{m_{ij}=k} 1$ for each machine type k. There are $\left(2\sum_{m_{ij}=k} 1\right) = \left(\sum_{m_{ij}=k} 2\right)$ time points mapping to either 1 or -1.
Next, a mapping matrix of 2 by N is formed for flow shop problem, (or 2 by $\left(\sum_{m_{ij}=k} 2\right)$ in general job shop problem) by sequencing all cell matrices together.

$$\mathcal{TC}_k = \left[\begin{array}{cccc} \mathcal{TC}_k^{se}(1) & \mathcal{TC}_k^{se}(2) & \cdots & \mathcal{TC}_k^{se}(N) \end{array} \right] \tag{2.13}$$

Note that $\mathcal{TC}_k$ is ordered by job and operation. The lookup table should be sorted by time, which is simply done by sorting the matrix sequence in ascending order of the first row. Each element of the second row will follow the original mapping element in the first row. The following pseudo-code instruction will formulate this sorting process:

$$[T_k^{ord}, Index] = sort(\mathcal{TC}_k(1,:)) \tag{2.14}$$

$$\mathcal{TC}_k^{ord}(1,:) = T_k^{ord}. \tag{2.15}$$

$$\mathcal{TC}_k^{ord}(2,:) = \mathcal{TC}_k(2, Index) \tag{2.16}$$

Then,

$$M_k(t) = \sum_{j:\mathcal{TC}_k^{ord}(1,j)\leq t} \mathcal{TC}_k^{ord}(2,j) \tag{2.17}$$

Hence, (2.12) to (2.17) give the complete implementation of machine utilization function in continuous time domain. For on-line programming where numerical error might happens for improper rounding, an infinitesimal value ϵ could be added in (2.12), which will

be replace by (2.18).

$$\mathcal{TC}_k^{se} = \left\{ \left[\begin{pmatrix} T_{ij}^s + \epsilon \\ 1 \end{pmatrix}, \begin{pmatrix} T_{ij}^e \\ -1 \end{pmatrix} \right] : \right\} \quad 1 \le i \le N, 1 \le j \le o_i \text{ and } m_{ij} = k \tag{2.18}$$

2.2.2 Utilization function in discrete time domain

Given a set of binary variable $\{X_{ijt} \in \{0,1\} : 1 \le i \le N, 1 \le j \le o_i, 1 \le t \le \tilde{T}\}$ under the constraints by (2.1). The machine utilization function could be directly constructed by the first part of (2.5)

$$M_{kt}(X) = \sum_{i,j:m_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > t_{ij} & (X_{ijt} - X_{i,j,t-t_{ij}}) \\ \text{if } t \le t_{ij} & X_{ijt} \end{array} \right\}, \quad \forall k \in \{1,2,...,K\} \tag{2.19}$$

However, it may be more convenient and explicit to be transformed to the expression in continuous time.

$$T_{ij}^s = t^* - 1 \Leftrightarrow X_{ij,t^*} = 1 \text{ and } X_{ij,t^*-1} = 0 \tag{2.20}$$

$$T_{ij}^e = T_{ij}^s + t_{ij} \tag{2.21}$$

Then equations {(2.18), (2.13), (2.14), (2.15), (2.16), (2.17)} could be used to construct the machine utilization function $M_k(t)$. Because of discrete time domain, $\{(T_{ij}^s, T_{ij}^e) : i = 1,2,...,N; j = 1,2,...,o_i\}$ are all positive integers, the problem of rounding error does not exist here. Furthermore, the following procedure could be used to build the machine utilization function.

Machine Utilization Function in Discrete Time Domain

1: *Initialization* $M_k(t) = 0, k = 1,2,...,K, t = 1,2,..,\tilde{T}$

2: *Construct Machine Utilization Function*, For $i = 1, 2, ..., N; j = 1, 2, ..., o_i$

i) $k = m_{ij}$.

ii) For $t = T^s_{ij}, T^s_{ij} + 1, ..., T^e_{ij} - 1$

$$M_k(t+1) = M_k(t+1) + 1;$$

The operation of $t+1$ is because the discrete time slot variable t starts from 1, while the continuous time variable T^s_{ij} starts from 0.

2.3 Scheduling for Multi-Machine Flow-shop

The problem is abstracted from a real-world port operation. The problem has the following characteristics:

(i) Each job has at least 3 operations executed consecutively;

(ii) The operation flow may occur in either direction,i.e. the forward flow operation and reverse flow operation;

(iii) There are multiple renewable machines, and machines of the same type are assumed to be exchangeable;

(iv) Machine availability is time-variant, i.e. an agent may have different number of machines in different period; and

(v) There are what we call critical operation sequencing **(COS)** constraints - each job has a critical operation, which can only begin when the previous job's critical operation has been completed.

In (i), the 3 operations can be conceptually seen as preprocessing, transportation(or travel) and postprocessing. Characteristic (ii) states a common feature of logistics problems, which need to care about delivering goods from and to a service center. For example, in a container terminal, the jobs are discharging the containers from a vessel to yard and loading containers from yard onto a vessel. Similarly, a 3^{rd}-party logistics (3PL) provider handles delivery of goods from the port to warehouses or customers and/or reverse direction delivery to the port. Characteristic (iii) makes the model different from classical single-machine flow-shop, and note that goods can be delivered in both directions by same group of machines. Characteristic (iv), called as **Multi-Period** constraint, is also different from classical job-shop (flow-shop) problems. In (v), the presence of COS constraints is motivated by a problem of operational scheduling in a container terminal, where COS constraints arise from the stacking or unstacking of heavy containers, which must observe certain sequence. This sequence can be generalized to priority sequence of jobs in other applications. On the other hand, the critical operation is usually an interface between two parties, and it requires the most expensive machine to operate and hence it is particularly important to sequence them back-to-back so that the expensive machine can be fully exploited. In this thesis, we assume that the critical operation is either the first or the last operation of each job; if the critical operation is the first operation, it is so called a **Forward Flow Job** and otherwise, it is a **Reverse Flow Job**. In the context of a container terminal for example, a forward flow job is to deliver a container from the vessel to the yard, while the reverse flow is to deliver it from a yard to a vessel.

Although the model could be applied for any fixed number of operations, this thesis puts focus on the simple flow shop scenario with 3 operations. Complete formulation in

$$\textbf{ScheGen-IP: minimize } \sum_t W_m(1 - X_{N,o_N,t}) \quad + \quad \sum_{t>D-p_{N,o_N}} W_d(1 - X_{N,o_N,t}) \tag{2.22}$$

$$\text{subject to: } X_{ijt} - X_{i,j,t+1} \leq 0, \quad \forall i,j, \forall t \in \{1,2,...,\tilde{T}-1\} \tag{2.23}$$

$$f_{OPRECE} = \left\{ \begin{array}{ll} X_{i,j,t} - X_{i,j-1,t-p_{i,j-1}} & \text{if } t > p_{i,j-1} \\ X_{i,j,t} & \text{if } t \leq p_{i,j-1} \end{array} \right\} = 0, \quad \forall i,j \in \{2,...,o_i\} \tag{2.24}$$

$$g_{MACAP}(X) = \sum_{i,j:m_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > t_{ij} & (X_{ijt} - X_{i,j,t-t_{ij}}) \\ \text{if } t \leq t_{ij} & X_{ijt} \end{array} \right\} - C_{k,F_\tau} \leq 0, \forall t \in F_\tau, \quad \forall k \in \{1,2,...,K\} \tag{2.25}$$

$$g_{COS} = \left\{ \begin{array}{ll} X_{i,j_i^*,t} - X_{i-1,j_{i-1}^*,t-p_{\{i-1,j_{i-1}^*\}}} & \text{if } t > p_{\{i-1,j_{i-1}^*\}} \\ X_{i,j_i^*,t} & \text{if } t \leq p_{\{i-1,j_{i-1}^*\}} \end{array} \right\} \leq 0, \forall i. \tag{2.26}$$

$$X_{ijt} \in \{0,1\} \forall i,j,t \in \{1,2,...,\tilde{T}\}. \tag{2.27}$$

$$\text{minimize} \quad W_m T_{MS} + W_d \max\{0, T_{MS} - D\} \tag{2.28}$$

$$\text{subject to: } f_{PREEM} = T_{ij}^s - T_{ij}^e + t_{ij} = 0, \forall i,j. \tag{2.29}$$

$$f_{OPRECE} = T_{i,j-1}^e - T_{ij}^s = 0, \forall i, j \geq 2. \tag{2.30}$$

$$g_{MACAP}(t) = M_k(t) - C_{k,F_\tau} \leq 0, \forall t \in F_\tau, k \in \{1,2,...,K\} \tag{2.31}$$

$$g_{COS} = T_{i,j_i^*}^e - T_{i+1,j_{i+1}^*}^s \leq 0, \forall i \in \{1,2,...,N-1\}. \tag{2.32}$$

$$T_{i,j_i^*}^e - T_{MS} \leq 0, \forall i \tag{2.33}$$

$$T_{ij}^s \geq 0 \tag{2.34}$$

$$T_{ij}^e \geq 0 \tag{2.35}$$

discrete-time are listed from (2.22) to (2.27)and continuous-time are listed from (2.28) to (2.35).

2.3.1 Solution equivalence between discrete time formulation and continuous time formulation

Above formulations are actually equivalent, if we do not consider the digitalization difference. The equivalence is shown in the following proposition.

Proposition 2.3.1 *The solution to Continuous Time Formulation is equivalent to the*

Discrete Time Formulation, under the following conditions:

- *Job-lists are bidirectional flow-shop, and reverse-flow jobs are always following forward-flow jobs; and*
- *For reverse flow jobs, the critical operation is the job's last operation (post-processing).*

Proof: Note the definition of X_{ijt} shows that

$$X_{i,j,t^*-1} == 0 \quad \cap \quad X_{i,j,t^*} == 1$$

$$\Updownarrow$$

$$T^s_{ij} = t^* - 1, \qquad i^{th} \text{ job's } j^{th} \text{ operation starts at time slot } t^*.$$

According to equation (2.23) or non-preemptive assumption, completion time of N^{th} job's o^{th}_N operation is

$$U_T \sum_t (1 - X_{N,o_N,t}) + p_{N,o_N}$$

Because the reverse job's last operation is the critical operation and reverse-flow jobs are always following forward-flow jobs, makespan of such job-list is always the completion time of the last job's last operation, which is N^{th} job's o^{th}_N operation. The makespan cost for discrete time formulation is

$$\begin{aligned} & W_m \cdot U_T \left(\sum_t (1 - X_{N,o_N,t}) + p_{N,o_N} \right) \\ = \quad & W_m U_T \cdot p_{N,o_N} + U_T \cdot \left(\sum_t W_m (1 - X_{N,o_N,t}) \right) \end{aligned}$$

Hence, the first term $W_m T_{MS}$ in continuous time domain objective function (2.28) is related with the first term of discrete time domain objective function (2.22) by a positive factor U_T and an offset $W_m U_T \cdot p_{N,o_N}$.

Similarly, we can see that the tardiness penalty for discrete time formulation is

$$\begin{aligned} & W_d \cdot U_T \sum_{t>D-p_{N,o_N}} (1 - X_{N,o_N,t}) \\ = \; & U_T \cdot \sum_{t>D-p_{N,o_N}} W_d(1 - X_{N,o_N,t}) \end{aligned}$$

The second term $W_d max\{0, T_{MS} - D\}$ in continuous time domain objective function (2.28) is related with the second term of discrete time domain objective function (2.22) just by a positive factor U_T.

For the overall objective function, sum of the weighted makespan cost and tardiness penalty for discrete time formulation is

$$\begin{aligned} & W_m U_T \cdot p_{N,o_N} + U_T \cdot \left(\sum_t W_m(1 - X_{N,o_N,t}) \right) \\ & + U_T \cdot \sum_{t>D-p_{N,o_N}} W_d(1 - X_{N,o_N,t}) \end{aligned}$$

Note that function (2.22) is exactly the same as

$$\left(\sum_t W_m(1 - X_{N,o_N,t}) \right) + \sum_{t>D-p_{N,o_N}} W_d(1 - X_{N,o_N,t})$$

Since $U_T > 0$ and $W_m U_T \cdot p_{N,o_N} > 0$ are all fixed positive parameters given in the job-list information, to minimize function (2.22) is equivalent to minimize function (2.28). **Q.E.D.**

2.3.2 Scheduling heuristic methods based on machine capacity constraint relaxation

For a general flow shop problem, non-preemptive scheduling for two machine flow shop ($K = 2, C_1 = C_2 = 1$) can be obtained in $O(n \log(n))$ using Johnson's algorithm [111]. In [100], it is proven that the flow shop with 3 and above machine types ($K \geq 3, C_k = 1$) is NP-complete. In [34], a greedy algorithm was proposed for port dispatching problem,

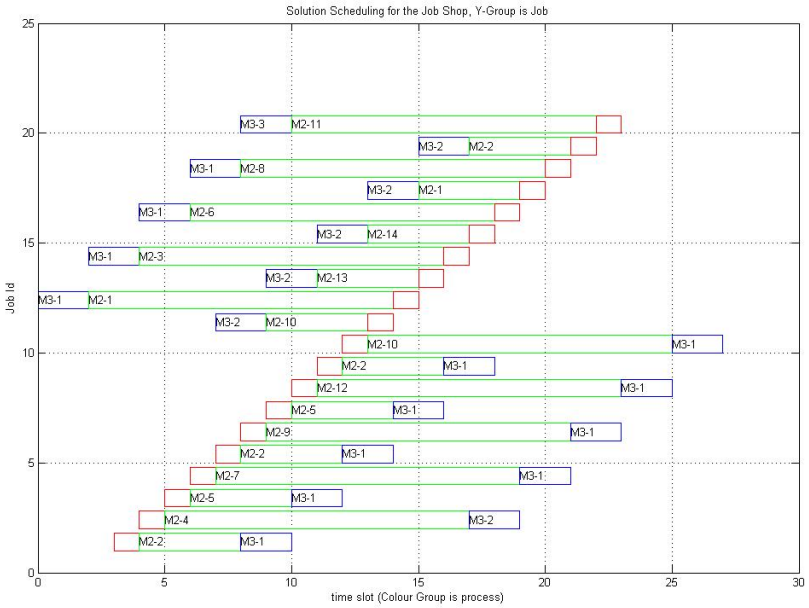
and that is actually a flow shop with three machine types but the 3^{rd} machine has infinite capacity ($K = 3, C_1 = 1, C_2 < \infty, C_3 = \infty$). The following is a heuristic method based on relaxation, which is extendable to arbitrary number of machine types and capacities, and it is implementable for both continuous and discrete time domains.

First, we generate a schedule by relaxing the machine capacity constraints, which means to only consider constraints {(2.29), (2.30), (2.32), (2.33), (2.34), (2.35)} in continuous time domain, or constraints {(2.22), (2.23),(2.24), (2.26), (2.27) } in discrete time domain. The scheduling is generated as in Fig. 2.1.

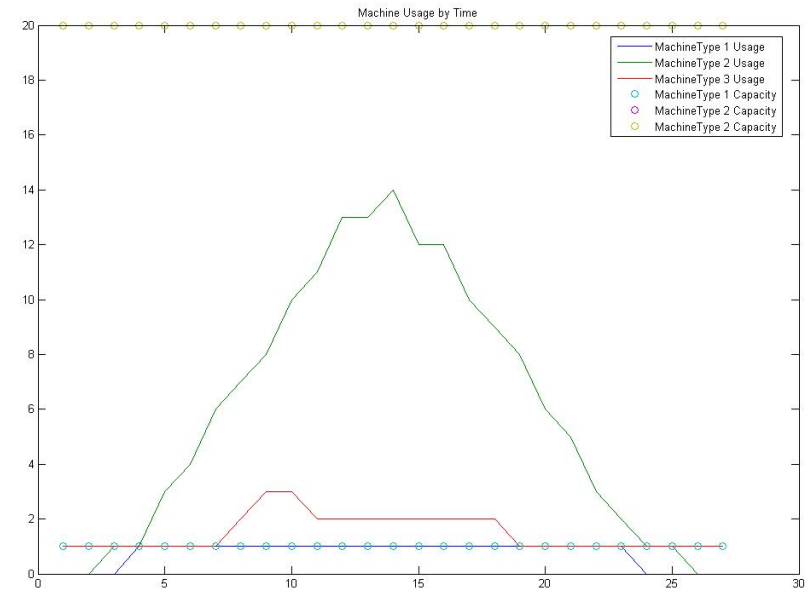
We observe from the machine utilization that the machine capacity constraint has been violated. Two heuristic methods are proposed as follows to handle this machine capacity violation. The first heuristic is a construction method that inserts the I^{th} job after $(I-1)$ jobs have been scheduled, and it is feasibility preserving. The second heuristic is a repair method that first relaxes machine capacity constraints by packing all jobs as early as possible (thereby optimizing the objective function) and then resolving capacity violation through a repair operation based on left-to-right sweep. In the experiments (to be presented later), we also consider a third greedy method proposed by Bish [34] and benchmark three heuristic methods with the optimal solution returned by the ScheGen-IP model.

The basic concept of the proposed heuristic methods are very much similar to Brucker's work [49], while ours are different in the following aspects:

- There is no-wait within consecutive tasks of a job;
- The proposed heuristic methods are suitable for both continuous time (i.e. $t_{ij} \in \mathcal{R}^+$) and discrete time problems (i.e. $t_{ij} \in \mathcal{Z}^+$); and



(a) Schedule



(b) Machine utilization

Fig. 2.1: Schedule generated without machine capacity constraints

- Ours are suitable for both constant machine capacity and period-dependant capacity.

2.3.3 Construction heuristic method(CH)

CH1: *Schedule the I^{th} job* based on the partial schedule of $I-1$ jobs, it can be scheduled under the constraints {(2.29), (2.30), (2.32), (2.33), (2.34), (2.35) } in continuous time domain, or constraints {(2.22), (2.23),(2.24), (2.26), (2.27) } in discrete time domain.

CH2: If $T^s_{I1} < 0$, shift time of all jobs $\{(T^s_{ij}, T^e_{ij}) : 1 \leq i \leq I, 1 \leq j \leq K = o_i\}$ to right by $|T^s_{I1}|$.

CH3: *Construct machine utilization function* according to equations {(2.18), (2.13), (2.14), (2.15), (2.16), (2.17) } in continuous time domain.

CH4: *While* there is any violation in machine capacity, $\max_t M_{kt} > C_{k,F_\tau}, \exists F_\tau, t \in F_\tau, k \in \{1, 2, ..., K\}$

CH4-a: Shift I^{th} job schedule by 1 slot in discrete time domain, or shift it to nearest time point in continuous time domain.

CH4-b: *Construct machine utilization function* according to equations {(2.18), (2.13), (2.14), (2.15), (2.16), (2.17)} in continuous time domain.

CH4: *loop*

2.3.4 Repair heuristic method(RH)

RH1: Construct an initial schedule with infinite resource capacity (i.e. considering only the constraints {(2.29), (2.30), (2.32), (2.33), (2.34), (2.35) } in continuous time

domain, or constraints {(2.23), (2.24), (2.26), (2.27)} in discrete time domain) and set T_{MS}. Initialize $t = 1$.

RH2: *while* $t < T_{MS}$

RH2-a: Construct the *machine utilization function* according to equations {(2.18), (2.13), (2.14), (2.15), (2.16), (2.17)} in continuous time domain.

RH2-b: *While* there is any violation in machine capacity at time t, $M_{kt} > C_{k,F_\tau}, t \in F_\tau, \exists k \in \{1, 2, ..., K\}$

RH2-b1: Construct the *violation job set*, which contains all such jobs that use machine k at time t

RH2-b2: $G = M_{kt} - C_{k,F_\tau}$;

RH2-b3: Within the *violation job set*, find the job who is the G^{th} latest in the job list;

RH2-b4: Shift this job such that next time slot starts its operation using machine k;

RH2-b5: Shift all the following jobs according to the constraints {(2.29), (2.30), (2.32)};

RH2-b6: Update T_{MS};

RH2-b7: Construct the *machine utilization function* according to equations {(2.18), (2.13), (2.14), (2.15), (2.16), (2.17) } in continuous time domain.

RH2-b: *loop*

RH2-c: $t = t + 1$;

RH2: *loop*

Note that the above algorithms can also use the discrete time domain machine utilization function (see Section 2.2.2). Heuristic RH is usually used in Lagrangian relaxation approach to repair the feasibility and get an upper bound estimation of makespan [22].

2.4 Scheduling for Multi-Machine Job-shop

The problem of this section is actually based on the operation of a semiconductor manufacturing plant. Firstly, we study the the behavior differences of the 2 objective function.

2.4.1 An empirical comparison of above two criterions

Classical jobshop problems (one-machine problem) are usually to minimize the makespan [90] [61]. However, we observe that the overall final machine release time, which is more important in the opinion especially for multi-machine problems and it is related with the sum of weighted completion time. Some experiments are given to compare the two criterions, the sample problems are chosen to be $\{MT6, MT10\}$ at machine capacity $C_k \in \{1, 2, 3, ...\}$.

Some terms are defined at first. There is a wait period between i^{th} job's j^{th} process and its precedence process iff. there is $T^e_{i,j-1} \neq T^s_{ij}; i = 1, ..., N, j = 2, ..., o_i$.

- *Total Wait: TW* - the total element count in $\{(i,j)|T^e_{i,j-1} \neq T^s_{ij}, i = 1, ..., N, j = 2, ..., o_i\}$;
- *Max Wait Time:* $WT^M = \max\{T^s_{ij} - T^e_{i,j-1}|i = 1, ..., N; j = 2, ..., o_i\}$;
- *Mean Wait Time:* $\overline{WT} = \frac{TotalWaitTime}{TotalWait} = \frac{\sum_{i,j}\{T^s_{ij} - T^e_{i,j-1}|i=1,...,N;j=2,...,o_i\}}{TW}$.

The relation between the continuous variable T^s_{ij} and discrete variable X_{ijt} has been

studied in previous section and listed as follows:

$$X_{i,j,t^*-1} == 0 \quad \cap \quad X_{i,j,t^*} == 1,$$

$$\Updownarrow$$

$$T^s_{ij} = t^* - 1, i^{th} \quad \text{job's} \quad j^{th} \text{ process starts at the time slot } t^*.$$

For the terms related with the machine utilizations, the definition is based on previous section, i.e. the construction of machine utilization function $M_k(t)$. Their definitions are shown as follows.

- *Mean Machine Utilization for type k:* $\frac{\int_t M_k(t)dt}{C_k}$;
- *Mean Utilization All Machines:* $\overline{AMU} = \frac{\sum_k \int_t M_k(t)dt}{\sum_k C_k}$,

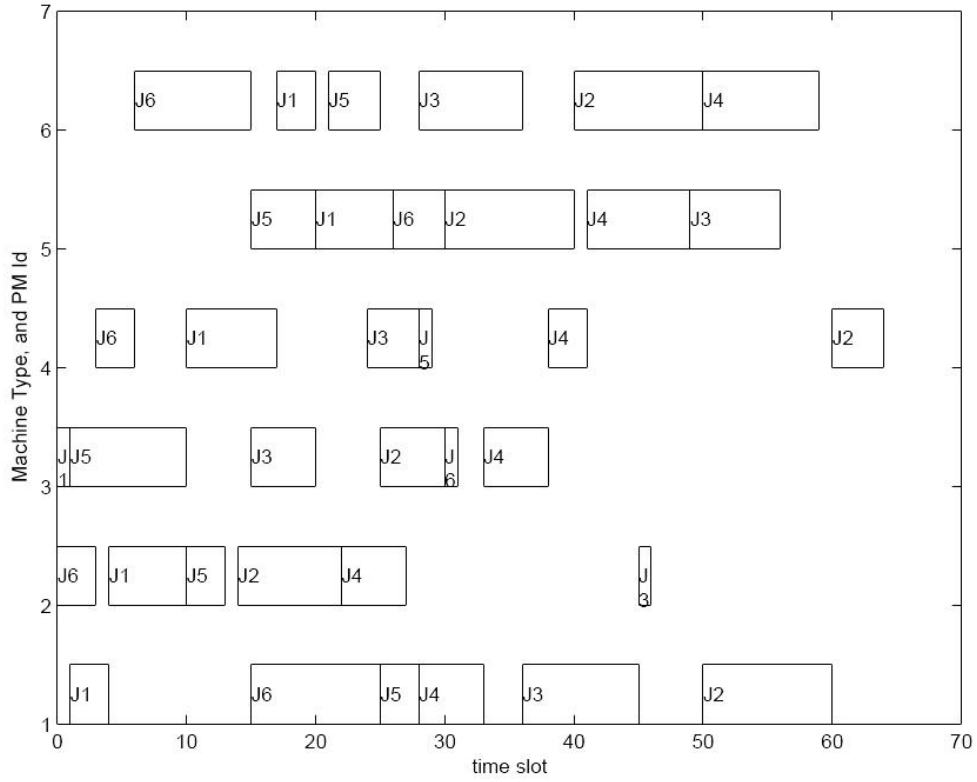
where the integration is from the very first time when the machine type k is put to use to the very last time when the machine type k stops using.

- *Mean Complete Time* is defined as $\overline{CTAJ} = \frac{T^e_{i,o_i}}{N}$. For the discrete formulation under assumption that all jobs' first process is arrived at time 0, it is equivalent to $\frac{\sum_i \sum_{t>0}(1-X_{i,o_i,t})}{N}$.

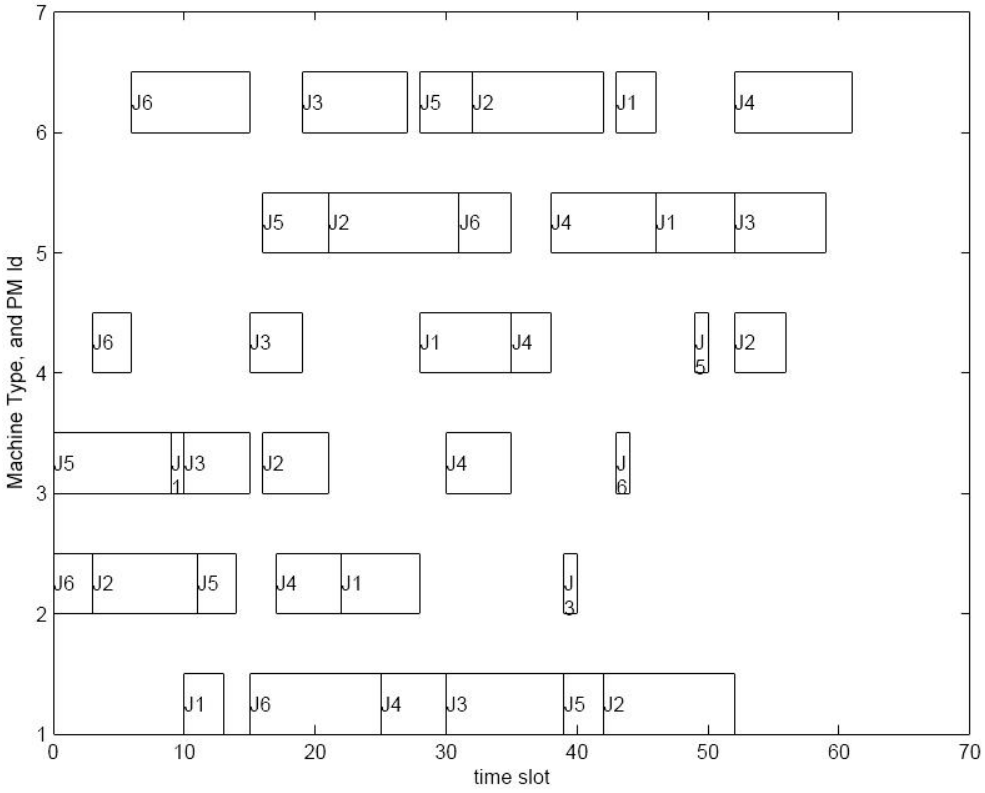
Fig. 2.2 demonstrates a one-machine MT6 optimal solution by CPLEX. Fig. 2.2(a) is to minimize the sum of completion time (or **criterion-2** with $d_i = 0, W_i = 1, \forall i$), while Fig. 2.2(b) is to minimize the makespan as **criterion-1**. The authors suggest to focus on only comparison of the length (*makespan*) and the area of last task with x-axis (*sum of weighted tardiness*).

Precisely, Table 2.1 is used to compare {1, 2}-machine by CRI-{1, 2} on the MT6 and the MT10 problems. All schedule solutions are done by CPLEX.

From Table 2.1, we have the following observations:



(a) To minimize sum of completion time (by CRI-2)



(b) To minimize the makespan (by CRI-1)

Fig. 2.2: Criterion Comparison: One-machine MT6 problem, grouping by machine

- When the total number of machines is increasing, both the makespan and the sum of weighted tardiness are dropping. When it reaches some level, both will keep the same value. This is similar to the Infinite Resource Model in [22].

- The sum of weighted tardiness (criterion-2) model is consistently better than makespan model (criterion-1), w.r.t. the wait property (i.e. less in total wait count and the wait time) and the mean complete time.

- For the mean machine release time, the 1-machine problem and the multi-machine problem are contrary to each other, while for the makespan and the machine utilization, there is no consistency.

Table 2.1: Comparison of 1-Machine v.s. 2-Machine and CRI-1 (Makespan) v.s. CRI-2 (Sum Tardi.) on MT6 & MT10 problems

	Criterion	$\overline{RTAM}$ time slot	T^M time slot	TW count	WT^M / $\overline{WT}$ time slot	$\overline{AMU}$ per cent	$\overline{CTAJ}$ time slot
MT6-Cap1	1	52.0	61	15	12 / 5.7	72.0%	53.7
	2	53.8	64	8	4 / 2.0	67.9%	44.2
MT6-Cap2	1	43.3	51	4	5 / 2.5	48.7%	47.5
	2	31.0	47	1	3 / 3.0	53.3%	35.2
MT10-Cap2	1	558.2	696	48	113 / 15.3	61.8%	661.3
	2	543.8	764	25	71 / 11.6	52.7%	578.9
MT10-Cap3	1	521.2	691	12	40 / 13.6	44.2%	657.8
	2	443.8	701	5	35 / 10.2	40.0%	532.8

2.4.2 Complete model for comparison with heuristic method

Based on above observation, we will further study the heuristic method for multi-machine job-shop with **CRI-2** as objective to minimize. For simplicity without loss of generality, $d_i = 0, W_i = 1, \forall i$. The complete model to benchmark the IMM heuristic method is shown as follows.

$$\textbf{JSPMultiMach} - \textbf{InfWait}$$

$$\text{minimize} \sum_i (W_i \sum_{t>d_i-p_{i,o_i}} (1 - X_{i,o_i,t})) \tag{2.36}$$

$$\text{subject to: } X_{ijt} - X_{i,j,t+1} \leq 0, \qquad \forall i,j,t \in \{1,2,...,\tilde{T}-1\} \tag{2.37}$$

$$\left\{ \begin{array}{ll} X_{i,j,t} - X_{i,j-1,t-p_{i,j-1}} & \text{if } t > p_{i,j-1} \\ X_{i,j,t} & \text{if } t \leq p_{i,j-1} \end{array} \right\} \leq 0, \forall i,j \in \{2,...,o_i\}. \tag{2.38}$$

$$\sum_{i,j:M_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > p_{ij} & (X_{ijt} - X_{i,j,t-p_{ij}}) \\ \text{if } t \leq p_{ij} & X_{ijt} \end{array} \right\} - C_k \leq 0, \forall k \in \{1,2,...,K\} \tag{2.39}$$

$$X_{ijt} \in \{0,1\} \qquad \forall i,j,t \in \{1,2,...,\tilde{T}\}. \tag{2.40}$$

This is the formulation for CPLEX solution of a 2-machine MT6 problem and shown in Fig. 3.1.

2.5 Heuristic Solution: Iterative Minimization of a Micro-model

The micro-model is defined for one particular machine type k with capacity C_k, on which, totally $L_k(R)$ processes are to be carried out. This model is solved and updated iteratively for the overall multi-machine JSP. R is a counter for iterations, and $L_k(R)$ is the total number of executable tasks on machine type k. "Executable" means satisfying sequencing constraints. The complete flow-chart is shown in Fig.2.3 which mainly has two parts, one is to find an local optimal solution to minimize a micro-model, the other is to iteratively select the machine type and build the micro-model accordingly.

2.5.1 Minimization of a micro-model

In the Micro-model, there are C_k exchangeable and renewable machines, serving $L_k(R)$ tasks. Each machine has a release time $t_m^r : m \in \{1,2,...,C_k\}$ while each task has an

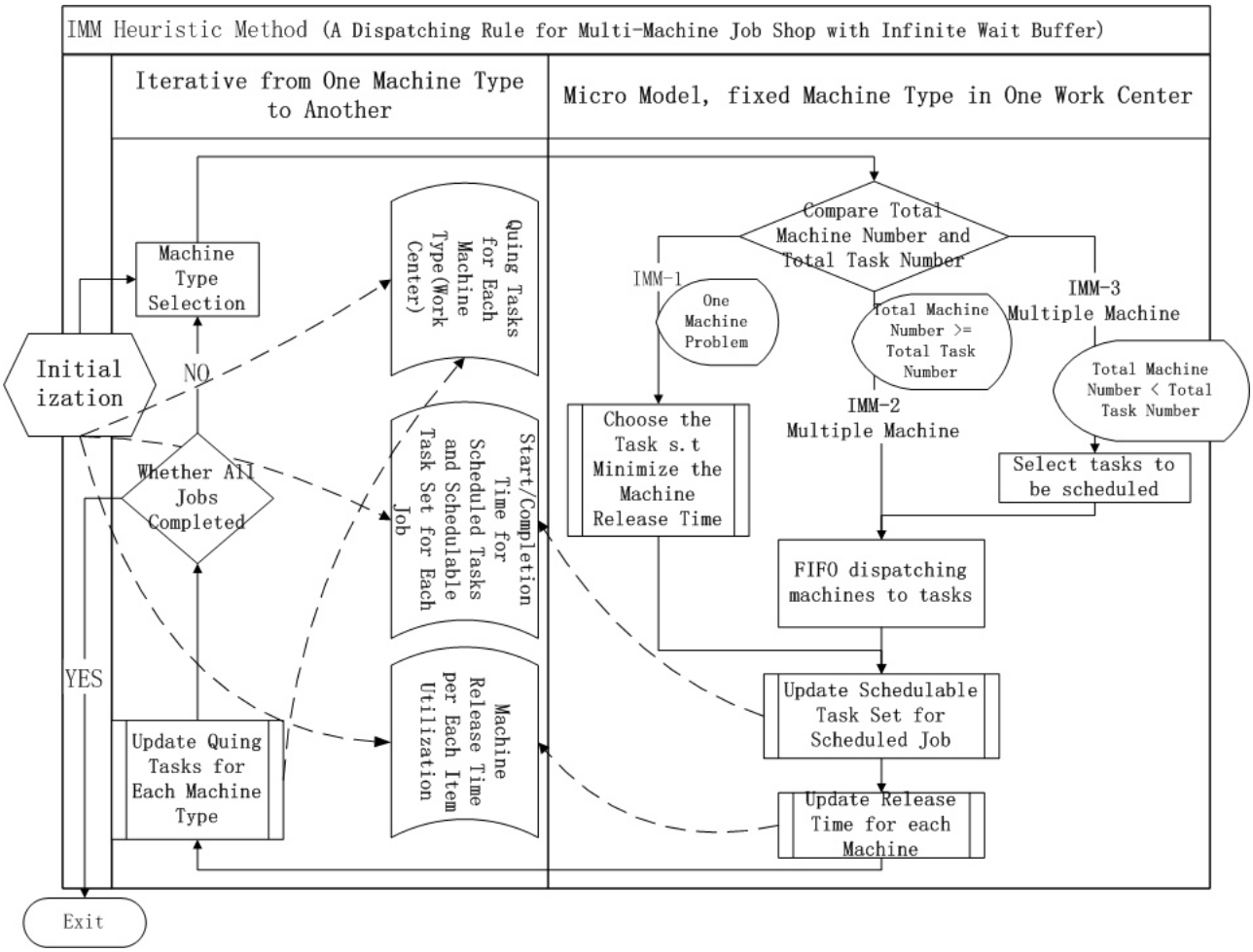


Fig. 2.3: Flow Chart of the Proposed IMM Heuristic Method

arrival time $t_l^a : l \in \{1, 2, ..., L_k(R)\}$ and processing time $p_l : l \in \{1, 2, ..., L_k(R)\}$.

- t_m^r is the time for a particular machine m to be ready to start a new task.
- t_l^a is the time for task l to be ready to start. It is the completion time of task l's precedence task which is specified in sequencing constraint.
- $p_l, 1 \leq l \leq L_k(R)$, processing time for task l.

The problem is to minimize the sum of completion (or end) time. The summation is done for $\min\{C_k, L_k(R)\}$ tasks.

$$\text{Minimize: } \sum_{mi(l)} \sum_{mi(l)\in\{1,...,C_k\}, l\in\{1,2,...,L_k(R)\}} t_l^e \tag{2.41}$$

$$\text{subject to: } t_l^s = \max\{t_l^a, t_{mi(l)}^r\} \forall l \tag{2.42}$$

$$t_l^e = t_l^s + p_l, \forall l \tag{2.43}$$

$$\text{decision variable: } mi(l) \in \Pi(L_k(R) \rightarrow C_k) \tag{2.44}$$

$$t_l^s, t_l^e \in \mathcal{R}, \forall l = 1, 2, ..., L_k(R) \tag{2.45}$$

This micro-model is to minimize the sum of end time in schedulable and executable task schedules as in (2.41) under the constraints of (2.42) and (2.43) with variable mapping (2.44), and the time variable lists (2.45).

From (2.41), the total number of items in the summation is $\min\{C_k, L_k(R)\}$ where C_k is for scheduleable constraints by the machine availability and $L_k(R)$ is for executable constraints of precedence. In (2.41), $mi(l)$ is a 1-to-1 mapping from the task-id $l \in \{1, 2, ..., L_k(R)\}$ to the machine-id $m = mi(l) \in \{1, 2, ..., C_k\}$

There is a post-process after solving above minimization micro-model, which is to update the machine release time for next group of tasks, as shown in (2.46).

$$t^r_{mi(l)} = t^e_l, \forall l \text{ executable \& schedulable} \tag{2.46}$$

2.5.2 Three scenarios and computational complexity

According to the relative value of C_k and $L_k(R)$, there are 3 scenarios in total for consideration:

IMM1: $L_k(R) \geq C_k = 1$, this is actually a one-machine problem. This can be solved optimally by enumeration, which is shown in **Proposition-2.5.1**.

IMM2: $C_k \geq L_k(R) \geq 1$, the total number of machines is not less than the total number of tasks. The optimal solution for this case is summarized in **Proposition-2.5.2**.

IMM3: $L_k(R) > C_k > 1$, the total number of machines is less than the total number of tasks.

Proposition 2.5.1 *For* **IMM1**, *the single machine is* m_1 *and the task list is* $l \in \{1, 2, ..., L_k(R)\}$. *The optimal objective value is following:*

$$\min\left(\min_{l:t^a_l \leq t^r_{m_1}}\{p_l + t^r_{m_1}\}, \min_{l:t^a_l > t^r_{m_1}}\{p_l + t^a_l\}\right)$$

Proof: It can be proved by enumeration. **Q.E.D.**

Proposition 2.5.2 *For* **IMM2**, *the optimal solution is to assign earliest available* $L_k(R)$ *machines for tasks by first come first serve (FCFS) rule.*

Proof: Let us begin with 2 tasks l_1, l_2 on 2 parallel machines m_1, m_2. The processing time and arrival time of the 2 tasks are $\{p_{l_1}, t^a_{l_1}\}$ and $\{p_{l_2}, t^a_{l_2}\}$, while the machine release

time of the 2 machines are $t^r_{m_1}, t^r_{m_2}$. Without loss of generality, we assume that $t^a_{l_1} \leq t^a_{l_2}$ and $t^r_{m_1} \leq t^r_{m_2}$. The optimal objective value is $p_{l_1} + \max\{t^a_{l_1}, t^r_{m_1}\} + p_{l_2} + \max\{t^a_{l_2}, t^r_{m_2}\}$ because it can be verified by all the possible cases as follows:

$$t^a_{l_1} \leq t^a_{l_2} \leq t^r_{m_1} \leq t^r_{m_2}$$

$$t^a_{l_1} \leq t^r_{m_1} \leq t^a_{l_2} \leq t^r_{m_2}$$

$$\vdots$$

$$t^r_{m_1} \leq t^a_{l_1} \leq t^r_{m_2} \leq t^a_{l_2}$$

$$t^r_{m_1} \leq t^r_{m_2} \leq t^a_{l_1} \leq t^a_{l_2}$$

The first come first serve rule will result as following mapping $mi(l_1) = m_1, mi(l_2) = m_2$, so it is true for 2-task-2-machine case.

For general cases, $C_k \geq 2, L_k(R) \geq 2$, for any mapping solution $mi(l)$, if $\exists\{l_1, l_2, m_1, m_2\}$, s.t. $mi(l_1) = m_1, mi(l_2) = m_2$, and $(t^a_{l_1} - t^a_{l_2})(t^r_{m_1} - t^r_{m_2}) < 0$. By swapping the mapping to first come first serve order will always achieve a better solution. **Q.E.D.**

For the case of **IMM3**, we solve the problem in two stages as follows:

1: Select C_k of tasks in total $L_k(R)$ schedule-able tasks;

2: Match above selected task with the C_k machines by FCFS sequence.

IMM3-2 is optimal and the proof is the same as in Proposition-2.5.2. For IMM3-1, the tasks are simply sorted by $(t^a_l + p_l)$ and take the smallest C_k tasks. This is not an optimal solution, but a fast feasible solution. The optimality depends on machine id sequencing.

Computational complexity for **Proposition-1** is $O(L_k(R))$, and for **Proposition-2.5.2** is $O\left(L_k(R) \cdot ln(L_k(R))\right)$, while the fast feasible solution of **IMM3** is $O(L_k(R) \cdot C_k)$

2.5.3 Construct the micro-model Iteratively

Besides the solution to the micro-model, another critical part is how to iteratively select the machine type as shown in the left part of Fig. 2.3. The previous work [92] shows preference to the rule of choosing such machine type that has the smallest expected release time. Empirically, we compared above rule with the following rules and find above rule is consistently better.

- to maximize the expected machine release time;
- to minimize the sum of processing time of all such tasks as quing for the machine;
- to minimize the sum of processing time and arrival time of all such tasks as quing for the machine;
- to maximize the sum of processing time and arrival time of all such tasks as quing for the machine;
- to minimize the sum of machine release time; and
- to maximize the sum of machine release time;

However, above rule of minimize the expected machine release time is still not optimal. We choose the idea of *selection* in genetic algorithm. That is, we randomly generate a population of sequence of machine type and then run above micro model. Then we choose the best solution from above population. This turns out to be a better solution than the rule based one. In order to save computational time, we do not perform the complete genetic algorithm including cross-over, mutation and iteratively generation.

2.6 Experiment and Preliminary Results

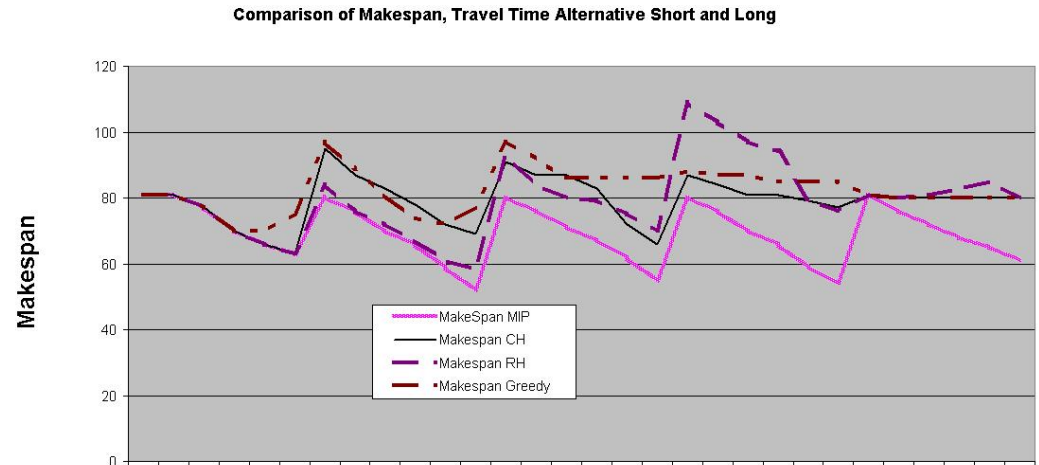
For above two kinds of problems, bi-directional flow-shop with COS constraints and multi-machine job-shop with infinite wait buffer, we explain the experiments correspondingly.

2.6.1 Transportation scheduling for a port

The experiments' result is to compare the makespan derived from the heuristic methods *CH*, *RH*, Greedy algorithm [34] and ScheGen-IP model. The common setting is as follows:

- 20 jobs are executed on 3 types of machines;
- Machine capacity is $C_1 = 1, C_2 = 4, C_3 = 2$;
- Critical operation is on 1^{st} machine type;
- Forward jobs followed by reverse jobs; and
- Processing time on 1^{st} machine is taken as unit, and on 3^{rd} machine is 2.

For the travel time, it alternately switches between two values (short, long). The mean value is set at 12. The (short, long) pairs are selected to be {[12, 12], [10, 14], [8, 16], [6, 18], [4, 20]}. The reverse job percentage *ReverseJobRatio* spans from {0, 25%, 50%, 75%, 100%}. Hence, a total of 25 job lists are generated for experiment. Fig. 2.4-(b) gives the comparison result. It shows that none of the heuristic methods works always better than the others, while in most of the cases *CH* is the best among all heuristic methods. Specially for pure *Forward Flow Jobs* with small variance in processing time,

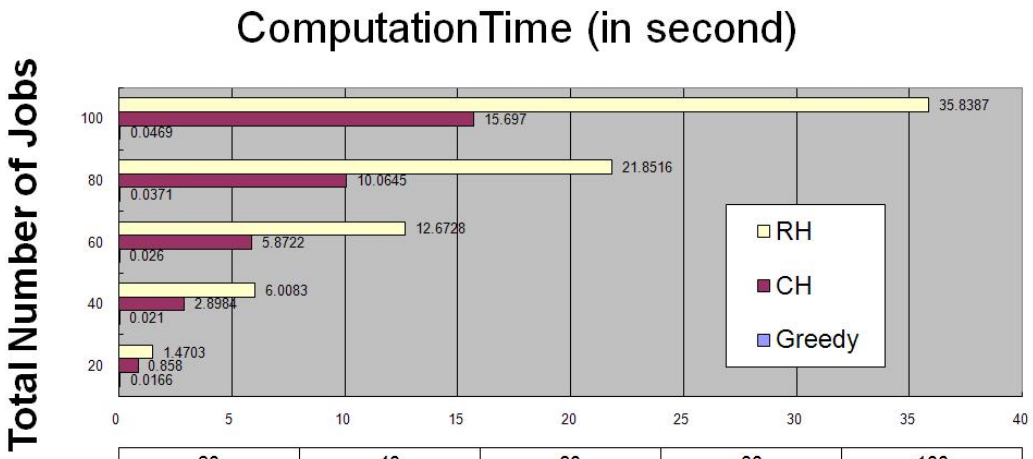


	0 0	0 1	0 4	0 9	0 17	0 26	0.25 0	0.25 1	0.25 4	0.25 9	0.25 17	0.25 26	0.5 0	0.5 1	0.5 4	0.5 9	0.5 17	0.5 26	0.75 0	0.75 1	0.75 4	0.75 9	0.75 17	0.75 26	1 0	1 1	1 4	1 9	1 17	1 26
MakeSpan MIP	81	81	77	70	66	63	80	76	70	66	59	52	80	76	71	67	62	55	80	76	70	66	59	54	81	76	72	68	65	61
Makespan CH	81	81	77	70	66	63	95	87	83	78	72	69	91	87	87	83	72	66	87	84	81	81	79	77	81	80	80	80	80	80
Makespan RH	81	81	77	70	66	63	84	76	72	67	61	58	93	84	80	79	75	70	109	103	97	94	79	76	81	80	81	83	85	80
Makespan Greedy	81	81	77	70	70	75	97	89	81	74	72	77	97	92	86	86	86	86	88	87	87	85	85	85	81	80	80	80	80	80

Fig. 2.4: Schedule makespan comparison: heuristic methods and ScheGen-IP

the heuristic method could achieve real optimal solutions. Generally, greedy algorithm performs worse when *Travel Time Variance* grows larger.

Next, the run times for *CH*, *RH* and Greedy algorithm are compared with more experiments. We measure run time for solution to job-lists with {20, 40, 60, 80, 100} jobs. The mean transportation time is 12 units. The average run time for *CH*, *RH* and Greedy algorithm are compared. The result is in Fig. 2.6.1. From the figure, it is clear that the Greedy algorithm is the fastest, while *RH* is slowest. However ScheGen-IP model is even much slower. It take 1 minute for 20 jobs, and around 40 minutes for 40 jobs.



	20	40	60	80	100
RH	1.4703	6.0083	12.6728	21.8516	35.8387
CH	0.858	2.8984	5.8722	10.0645	15.697
Greedy	0.0166	0.021	0.026	0.0371	0.0469

Fig. 2.5: Solution time comparison among scheduling heuristics

2.6.2 Benchmark on classical problems {MT*, ABZ*}– a multi-machine version

In order to show the effectiveness of our proposed IMM heuristic, we carried an extensive number of experiment on $\{MT6, MT10, ABZ5, ABZ6, ABZ7, ABZ8, ABZ9\} \times \{C_k = 1, C_k = 2, C_k = 3, C_k = 4 | \forall k\}$. Generally, we start from classical one-machine problem. Then we increase the machine capacity for each machine type until the makespan stop decreasing. Specially, we have 3 pairs $\{C_k = 1, C_k = 2, C_k = 3\}$ for MT6 problem, while we have 4 pairs $\{C_k = 1, C_k = 2, C_k = 3, C_k = 4\}$ for other problems, $\{MT10, ABZ5, ABZ6, ABZ7, ABZ8, ABZ9\}$

For each pair of experiments, comparison is done between our proposed IMM-heuristic method with the 0-1 formulation by CPLEX solution. In the 0-1 formulation, we use the sum of completion time (or equivalently mean completion time) as the objective function to minimize, which is different from the definition of classical job shop problem where the makespan is to be minimized. Except this, the job-process-machine setting is exactly

same as the classical MT* and ABZ* problems correspondingly.

MT6: First 3 pairs in Table 2.2 are the MT6 problems with $\{C_k = 1, C_k = 2, C_k = 3\}$– machine(s) for each type;

MT10: In Table 2.3, machine capacity spans as $\{C_k = 1, C_k = 2, C_k = 3, C_k = 4\}$;

ABZ5: In Table 2.4, machine capacity spans as $\{C_k = 1, C_k = 2, C_k = 3, C_k = 4\}$;

ABZ6: In Table 2.5, machine capacity spans as $\{C_k = 1, C_k = 2, C_k = 3, C_k = 4\}$;

ABZ7: In Table 2.6, machine capacity spans as $\{C_k = 1, C_k = 2, C_k = 3, C_k = 4\}$;

ABZ8: In Table 2.7, machine capacity spans as $\{C_k = 1, C_k = 2, C_k = 3, C_k = 4\}$;

ABZ9: In Table 2.8, machine capacity spans as $\{C_k = 1, C_k = 2, C_k = 3, C_k = 4\}$;

There are following points for notice:

- both CPLEX and IMM-heuristics solver runs on Intel(R) Xeon(R) CPU of X3220 2.4 GHz with 2.4GHz & 4.0 GB RAM;
- IMM-heuristic method runs roughly about 1 second for MT6 problems, 3 seconds for $\{MT10, ABZ5, ABZ6\}$ problems or 5 seconds for $\{ABZ7, ABZ8, ABZ9\}$ problems.
- Generally, CPLEX solver engine is stopped after one hour and return the best feasible solution, which is filled in the table;
- Specially for the problems marked with * in the table, i.e. MT10-Cap1, ABZ5-Cap1, ABZ6-Cap1, ABZ7-{Cap1, Cap2, Cap3}, ABZ8-{Cap1, Cap2} and ABZ9-{Cap1, Cap2}, CPLEX alone cannot even achieve a feasible solution if starting

from scratch. If we feed the CPLEX solver by an initial solution which is obtained from IMM heuristic method, after one hour CPLEX just return the same as the initial solution. So for these problems, we fill same IMM-Heuristic solution in the table as the CPLEX solution;

- Except for above specialities, CPLEX solution is generally better in the sense that it is smaller in mean completion time;

From the table, we can see that the performance of IMM-heuristic method excel when machine capacity is small, while the overall optimality is within 110% compared with CPLEX.

Table 2.2: Performance Bench Mark MT6 (IMM-Heuristics v.s. CPLEX)

Problem	Method	Mean Release Time (time slot)	Makespan (time slot)	TotalWait (count)	Max WaitTime (time slot)	Mean WaitTime (time slot)	Mean Utilization All Mach (per cent)	Mean Complete Time (time slot)
MT6-Cap1	IMM-Heuristic	45.8	63	10	20	7.5	79.1%	38.8
	CPLX	53.8	64	8	4	2.0	67.9%	44.2
MT6-Cap2	IMM-Heuristic	32.3	48	6	5	3.0	51.1%	36.8
	CPLX	31.0	47	1	3	3.0	53.3%	35.2
MT6-Cap3	IMM-Heuristic	26.0	47	0	0	0.0	36.6%	32.8
	CPLX	28.1	47	0	0	0.0	36.6%	32.8

Table 2.3: Performance Bench Mark MT10 (IMM-Heuristics v.s. CPLEX)

Problem	Method	Mean Release Time (time slot)	Makespan (time slot)	TotalWait (count)	Max WaitTime (time slot)	Mean WaitTime (time slot)	Mean Utilization All Mach (per cent)	Mean Complete Time (time slot)
MT10-Cap1	IMM-Heuristic	993.7	1066	41	173	49.6	66.1%	840.5
	CPLX *	993.7	1066	41	173	49.6	66.1%	840.5
MT10-Cap2	IMM-Heuristic	604.8	779	27	104	37.5	54.5%	631.3
	CPLX	543.8	764	25	71	11.6	52.7%	578.9
MT10-Cap3	IMM-Heuristic	467.1	655	18	66	27.8	41.4%	565.5
	CPLX	443.8	701	5	35	10.2	40.0%	532.8
MT10-Cap4	IMM-Heuristic	428.6	655	16	70	25.9	32.3%	552.4
	CPLX	416.7	655	2	13	9.0	32.0%	517.8

2.6.3 Comparison with existing job dispatch rules for a semiconductor manufacturing scheduling problem

The model of multi-machine JSP with an infinite wait buffer is just suitable for schedule application in back-end semiconductor manufacturing which has the following features:

- each job represents particular part (or one lot of chips), whose volume is not so

Table 2.4: Performance Bench Mark ABZ5 (IMM-Heuristics v.s. CPLEX)

Problem	Method	Mean Release Time (time slot)	Makespan (time slot)	TotalWait (count)	Max WaitTime (time slot)	Mean WaitTime (time slot)	Mean Utilization All Mach (per cent)	Mean Complete Time (time slot)
ABZ5-Cap1	IMM-Heuristic	1318.0	1389	45	210	62.9	69.3%	1114.5
	CPLX *	1318.3	1389	45	210	62.9	69.3%	1114.5
ABZ5-Cap2	IMM-Heuristic	818.4	923	17	74	33.8	51.1%	834.7
	CPLX	720.1	913	28	70	12.3	51.7%	812.8
ABZ5-Cap3	IMM-Heuristic	877.0	861	4	52	34.5	36.2%	791.1
	CPLX	660.7	868	3	16	8.7	36.1%	783.8
ABZ5-Cap4	IMM-Heuristic	842.8	859	1	38	38.0	27.6%	781.1
	CPLX	670.8	859	0	0	0	27.6%	777.3

Table 2.5: Performance Bench Mark ABZ6 (IMM-Heuristics v.s. CPLEX)

Problem	Method	Mean Release Time (time slot)	Makespan (time slot)	TotalWait (count)	Max WaitTime (time slot)	Mean WaitTime (time slot)	Mean Utilization All Mach (per cent)	Mean Complete Time (time slot)
ABZ6-Cap1	IMM-Heuristic	934.4	1065	45	171	49.1	69.0%	822.4
	CPLX *	934.4	1065	45	171	49.1	69.0%	822.4
ABZ6-Cap2	IMM-Heuristic	648.2	784	22	167	48.1	43.0%	700.5
	CPLX	560.7	751	18	69	16.8	47.1%	625.2
ABZ6-Cap3	IMM-Heuristic	552.8	742	5	98	46.2	32.7%	617.7
	CPLX	535.1	742	1	20	20.0	33.9%	598.6
ABZ6-Cap4	IMM-Heuristic	500.0	742	3	125	74.7	23.9%	617
	CPLX	534.6	742	0	0	0	25.5%	594.6

Table 2.6: Performance Bench Mark ABZ7 (IMM-Heuristics v.s. CPLEX)

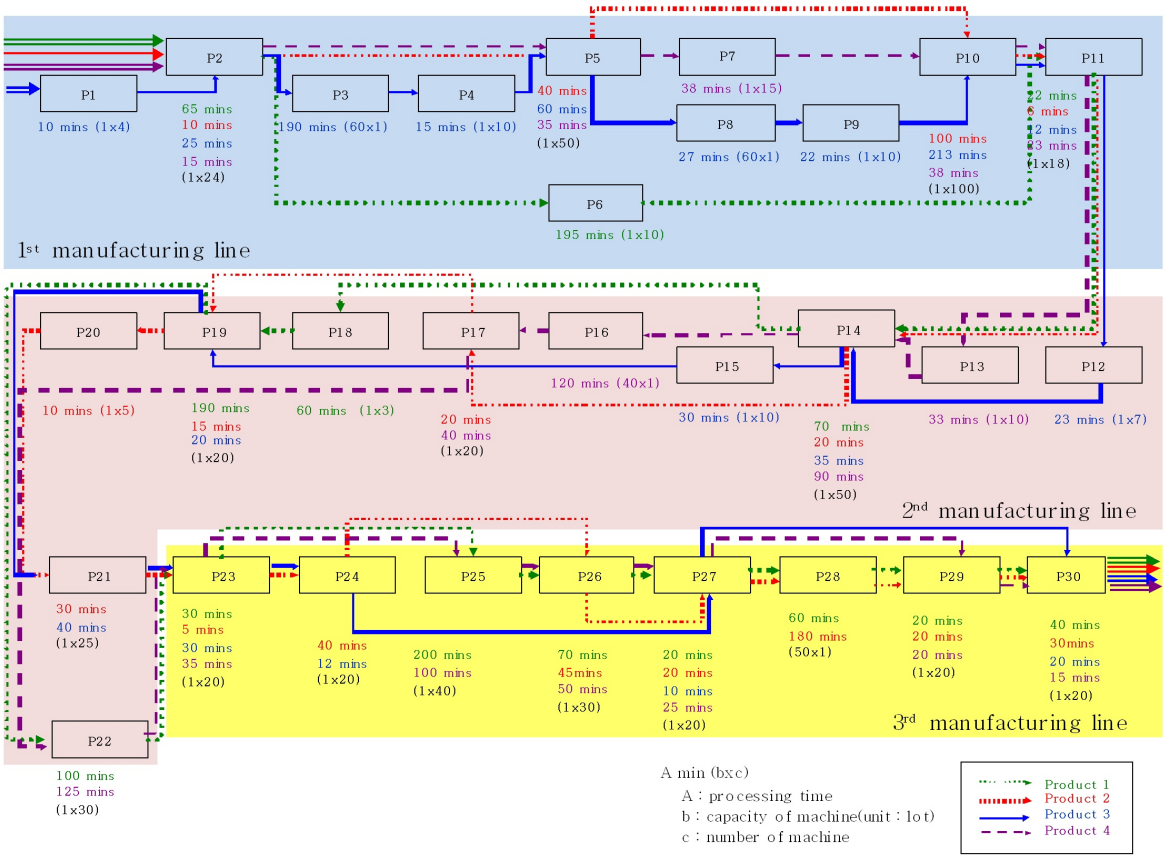
Problem	Method	Mean Release Time (time slot)	Makespan (time slot)	TotalWait (count)	Max WaitTime (time slot)	Mean WaitTime (time slot)	Mean Utilization All Mach (per cent)	Mean Complete Time (time slot)
ABZ7-Cap1	IMM-Heuristic	736.5	841	156	392	37.2	69.2%	675.35
	CPLX *	736.5	841	156	392	37.2	69.2%	675.35
ABZ7-Cap2	IMM-Heuristic	444.9	509	94	68	16.1	55.4%	446.65
	CPLX *	444.9	509	94	68	16.1	55.4%	446.65
ABZ7-Cap3	IMM-Heuristic	391.8	454	43	72	14.0	42.7%	398.95
	CPLX *	391.8	454	43	72	14.0	42.7%	398.95
ABZ7-Cap4	IMM-Heuristic	381.5	425	19	69	19.6	33.0%	386.95
	CPLX	317.5	411	14	11	3.5	34.7%	370.8

Table 2.7: Performance Bench Mark ABZ8 (IMM-Heuristics v.s. CPLEX)

Problem	Method	Mean Release Time (time slot)	Makespan (time slot)	TotalWait (count)	Max WaitTime (time slot)	Mean WaitTime (time slot)	Mean Utilization All Mach (per cent)	Mean Complete Time (time slot)
ABZ8-Cap1	IMM-Heuristic	855.3	873	150	263	38.0	66.3%	693.65
	CPLX *	855.3	873	150	263	38.0	66.3%	693.65
ABZ8-Cap2	IMM-Heuristic	478.6	524	85	66	18.7	53.8%	464.1
	CPLX *	478.6	524	85	66	18.7	53.8%	464.1
ABZ8-Cap3	IMM-Heuristic	420.0	470	39	73	15.0	40.4%	410.1
	CPLX	364.6	458	31	13	4.1	42.0%	387.55
ABZ8-Cap4	IMM-Heuristic	380.9	447	26	45	13.5	31.2%	396.8
	CPLX	335.7	443	3	7	3.0	31.9%	380.85

Table 2.8: Performance Bench Mark ABZ9 (IMM-Heuristics v.s. CPLEX)

Problem	Method	Mean Release Time (time slot)	Makespan (time slot)	TotalWait (count)	Max WaitTime (time slot)	Mean WaitTime (time slot)	Mean Utilization All Mach (per cent)	Mean Complete Time (time slot)
ABZ9-Cap1	IMM-Heuristic	855.4	890	147	291	35.9	65.2%	660.95
	CPLX *	855.4	890	147	291	35.9	65.2%	660.95
ABZ9-Cap2	IMM-Heuristic	535.9	553	92	65	19.0	53.2%	462.1
	CPLX *	535.9	553	92	65	19.0	53.2%	462.1
ABZ9-Cap3	IMM-Heuristic	400.8	484	50	60	14.6	40.6%	408.5
	CPLX	356.5	467	22	37	7.2	42.4%	380.35
ABZ9-Cap4	IMM-Heuristic	368.2	467	22	46	16.4	31.9%	390.1
	CPLX	328.7	467	2	3	2.5	32.4%	373.85



(a) Production Line with 4 Types of Products and 30 Types of Machines

Fig. 2.6: A Semiconductor Manufacturing Problem

Table 2.9: Sample Job List in a Semiconductor Manufacturing Problem

Process Machine Type	Capacity of machine	Total Number of machines	Process time (time units)			
			Product Id			
			1	2	3	4
P1	1	4			10	
P2	1	24	65	10	25	15
P3	60	1			190	
P4	1	10			15	
P5	1	50		40	60	35
P6	1	10	195			
P7	1	15				38
P8	60	1			27	
P9	1	10			22	
P10	1	100		100	213	38
P11	1	18	22	6	12	23
P12	1	7			23	
P13	1	10				33
P14	1	50	70	20	35	90
P15	1	10			30	
P16	40	1				120
P17	1	20		20		40
P18	1	3	60			
P19	1	20	190	15	20	
P20	1	5		10		
P21	1	25		30	40	
P22	1	30	100			125
P23	1	20	30	5	30	35
P24	1	20		40	12	
P25	1	40	200			100
P26	1	30	70	45		50
P27	1	20	20	20	10	25
P28	50	1	60	180		
P29	1	20	20	20		20
P30	1	20	40	30	20	15
Total Processes Per Job			14	16	18	16
Total 409 Jobs, Contribution from Each Product			12	161	117	119

Table 2.10: Solution Comparison for above Job List in Semiconductor Manufacturing

Tools	Method	Mean Release Time (time slot)	Makespan (time slot)	TotalWait (count)	Max WaitTime (time slot)	Mean WaitTime (time slot)	Mean Utilization All Mach (per cent)	Mean Complete Time (time slot)
IMM-Heuristic		983.1	1462	1504	193	46.1	50.5%	1042.0
ProModel	FIFO	1014.0	1575	2004	368	59.3	45.5%	1160.4
	LIFO	963.7	1601	2075	540	49.5	47.6%	1136.7
	SPT	1003.3	1592	1968	373	63.4	44.1%	1165.0

large and has standard size for easy storing;

- the temperature of the parts does not need strict control.

A complex manufacturing floor shop in a semiconductor industry is used as a further test bed in this study. Using ProModel, which is an industrial manufacturing modeling software, a simulation model of the floor shop is created. The specifications of the shop floor are summarized in Fig. 2.6 and the detailed processing time and process-machine mapping are shown in Table 2.9. There are 30 types of machines corresponding to 30 different processes and there are 4 product types, each with its own process flow and planed daily production throughput. Such problem is too hard to be solved by the 0-1 IP model as in previous section, while it can be solved by the IMM-heuristic method by around 1 minute for all 409 jobs. IMM-heuristic method achieves slightly better performance than the existing dispatching rules (FIFO, SPT and etc.) used in ProModel. The solution comparison is available in Table 2.6.3.

The job-shop diagram for the problem instance in Table is shown in Figure 2.6, which is drawn by ProModel.

2.6.4 Conclusion for IMM Heuristics

Based on the comparisons in above experiments, we can see that IMM heuristic achieves a fast feasible solution and the performance is comparably worse (within around 110%) than CPLEX solutions. Especially the IMM heuristic has the following features:

- It supports fractional processing time, or it supports continuous-time problems;
- The micro-model considers machine release time and task arrival time, which can be extended as an input from the whole problem of multi-machine JSP;
- It outputs machine dispatching solution and machine release time are also available, so the micro-model can be easily applied to construct a multiple batch of jobs problems in multi-machine environments;
- The overall computational complexity of IMM-heuristic is no more than

$$O(\left(N \cdot K \cdot \max_{i=1,2,...,N}\{o_i\}\right)^2)$$

- The heuristic achieves local optimality for micro-model in 2 scenarios, which are **IMM-1** and **IMM-2**.
- The optimality of the overall solution is affected by the sequencing in selection of machine type k at iteration R for solving the **Minimization Micro-model** and the selection of solving **IMM-3** during a specific iteration. This deserves further research.

Chapter 3

Dynamics Scheduling with Batch Job Arrival and Machine Degradation/Upgradaion

The previous chapter deals with static problems. In another words, it predictively generates the schedule, which will serve for operational time-line to be carried out in shop-floor production or transportation. This chapter deals with dynamic problems, i.e. to update schedule reactively to unexpected random events.

In the robotics and/or mechatronics control systems, there are the command generation and real-time feedback controller. While in the scheduling and resource allocation systems, there are the scheduling generation and real-time reactive controller. The controller in the robotics/mechatronics system, usually compares the pre-generated command and real-time feedback, then gives a decision that how much to output from a driver to correct the error between command and feedback. The reactive controller, in

the shop-floor production or transportation, monitors the schedule execution (i.e. feedback) and compares it with the pre-generated schedule. However, the reactive controller will cope with unexpected events and revise the schedule (command) for later execution.

In this chapter, our contribution is to extend the previous heuristics to handle some kinds of dynamics. The extended IMM heuristic is to handle batch job arrival in Section 3.3. We highlight the contribution of the circular charts in Section 3.2, which are studied in comparison with the rectangular charts. We also extend the RH and CH to handle machine de-gradation / up-gradation problems in Section 3.4. The implementation is explained with consideration of fast computation.

3.1 Specification of Machine Dynamics and Job Dynamics

We follow the literature [58, 80], and classify possible dynamics into 4 categories.

- Job dynamics (JD);
- Machine dynamics (MD);
- Process dynamics (PD); and
- Communication dynamics (CD).

First 3 categories of dynamics are from classical scheduling where a centralized scheduler/controller is applied. The 4th category is somehow from 2000s and it is based on decentralized scheduling or multi-agent's system.

Specifically for the first 3 kinds, the job dynamics include the following factors. Each has the preliminary work following.

JD-1: Job cancelation and new job arrival [59, 60, 75, 87];

JD-2: Due date/time change, (delay or advance) [8];

JD-3: Urgent (rush or hot) job arrival [76, 81, 86]; and

JD-4: Change in job priority [87].

The machine dynamics include the following details, each with the preliminary works.

MD-1: Machine degradation (speed lower down);

MD-2: Machine failure (or break down regular maintenance) [45, 65, 69, 75–77, 80, 81, 84, 86, 87, 94] and etc.;

MD-3: Delay in arrival or shortage of materials [31]; and

MD-4: Operator absenteeism.

It is noted that more than a dozen of papers are there to handle machine breakdown (or machine failure), but almost no reference can be found to handle machine degradation.

Process dynamics include the following details.

PD-1: Over- or under- estimation of process time [80, 101];

PD-2: Rework or quality problems.

Beside above specifications, the dynamic scheduling has 2 measurements for the solution quality. One is efficiency and the other is stability [1]. The efficiency measure is often the same as that used in static scheduling and typically reflects schedule quality using objectives such as makespan, tardiness or total cost. The stability measures how much the new schedule is different from the initial schedule and it reflects how much costs will be incurred due to the schedule changing.

With respect to the implementation of real-time system, there are generally 2 ways. One is event-driven and the other is periodically re-scheduling. In both ways, there is such requirement that the rescheduling algorithm should be fast. The total computation time spent for the reactive schedule must be shorter than the period in a periodically re-scheduling system. In either situation, the fast feasible and near-optimal solution is more preferred than the optimal solution, however with too long solution time.

Based on above observations, my work in this thesis are in the following 2 aspects;

1. To extend IMM heuristic in Sec. 2.5 to handle batch job arrival, at the same time, we will introduce the circular schedule chart.

2. To adjust CH in Sec. 2.3.2 to handle machine degradation and up-gradation.

3.2 Rectangular and Circular Gantt Chart

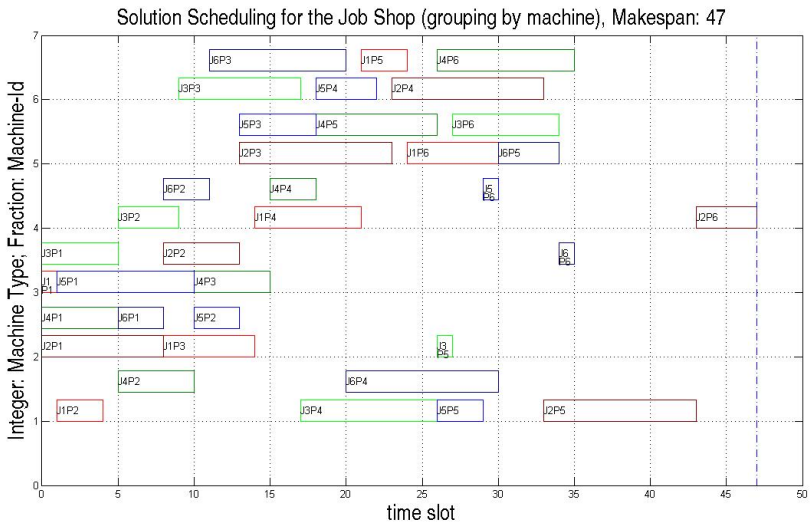
Conventionally, there are two types of charts. Both of them have the x-axis to represent the time, while they are different in y-axis, i.e.,

1. y-axis in 1st type represent machines, as Fig. 3.1-(a) and 1.3-(a);

2. In 2nd type of Gantt chart, the y-axis represents jobs, as Fig. 3.1-(b) and 1.3-(b).

The charts in Fig. 3.1 are typical Gantt charts for a 2-machine job-shop schedule. Fig. 3.1-(a) and Fig. 3.1-(b) actually originate from classical MT6 problem. Except that for each machine-type, there are 2 exchangeable machines. In Fig. 3.1-(a), y-axis integer is machine-type while fraction is machine-id.

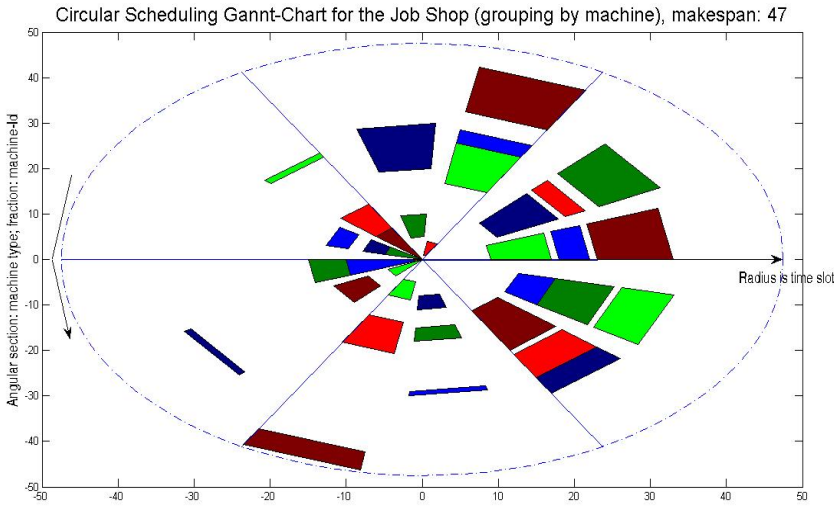
For classical one-machine job-shop problems (i.e. MT6, MT10 and etc.), 1st type and 2nd type are one-to-one mapped. For multi-machine problems, however, 1st type



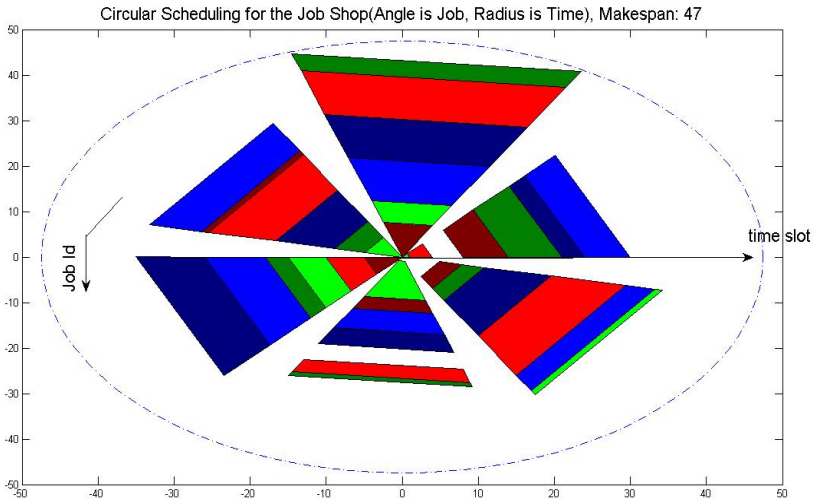
(a) schedule (by machine type (integer) and ID(fraction))

(b) schedule (by job)

Fig. 3.1: Typical Gantt chart for multi-machine job-shop schedule



(a) Dynamic scheduler (by machine type (integer) and ID(fraction))



(b) Dynamic scheduler (by job)

Fig. 3.2: Circular Gantt chart for multi-machine job-shop dynamic schedule

chart have multiple solutions which are mapped to the same job-schedule in 2nd type chart. For example, by swapping machine-id, there may be different machine-dispatching schedule, all of them satisfy the same job schedule.

Recently [11], there is the circular chart to represent dynamic task and/or resource allocation. For clear understanding by comparison, the same schedule as Fig. 3.1 is redrawn in circular type as in Fig. 3.2.

The Fig. 3.2-(a) is machine schedule and Fig. 3.2-(b) is job schedule.

From the above circular charts, we can see there are two dimensions:

Dim-1: Radius direction $[0, +\infty)$ is time-line;

Dim-2: Theta direction $[0, 2\pi)$ is job in job schedule as Fig. 3.2-(b), or machine type in machine-dispatching schedule as Fig. 3.2-(a).

The advantage for circular schedule is that the framework need not adjust and the latest job and/or machine schedule has the largest area and so they are best focused. This will be further explained in later section.

3.3 Extending IMM heuristic to Handle Batch Job Arrival

This batch job arrival is meaningful in many applications. For example, there are the cyclic scheduling in robotic cell [32], the dynamic single machine job-shop [59], the FMS (Flexible Manufacturing System) [60], the semiconductor production line in my previous Sec. 2.6.3 and etc.

For the mercy of easy representation, we take MT6 problem [110]. Let us consider a work shop with 6 work centers (for 6 different processes), each work center has 2 exchangeable machines. In each batch, there are 6 jobs, the machine-routing for the 6

jobs are exactly the same as MT6 problems. Totally there are 3 batches, and they will arrive in around every 45 minutes.

As stated in Sec. 2.6.4, the IMM heuristic can be easily extended to handle multiple batch of jobs. Note the following definition of t_l^a.

DEFINE: **Task Arrival Time,** an integer t_l^a is the time for task l to be ready to start. It is the completion time of task l's precedence task.

For the first task of a job, t_l^a is exactly the same as the job arrival time. For the dynamic jobs arrived in batches, there will be additional N variables to state the N jobs arrival time. For each job, it is the earliest start time of its first task.

For this problem, the mean time and standard deviation of job arrival are listed in the following Table. 3.1. As for the simulation of dynamic scheduling, job arrival time should be under some stochastic distribution. For example, inter-job arrival time is exponentially distributed in Bierwirth's work [60] and Vieira's [59].

The mean inter-job arrival time in [60], λ is related with a desired machine utilization rate U and mean mean processing time of all jobs $\bar{P}$.

In Vieira's model, the arrival rate for type-j jobs is θ_j. The job inter-arrival time is exponentially distributed, the probability that a job of type j arrives during a rescheduling period P_r is $1 - exp(\theta_j P_r)$. Vieira also assumes the arrival of jobs of one type is *independent* of the arrivals of jobs of another type.

In the example case, MT6, each batch consists 6 jobs and they are all different types to each other. We simply assume that the 6 jobs' arrival time is uniformly distributed. Above all, this example is just to verify my *IMM* algorithm and to introduce the circular chart.

As shown in Table. 3.1, the mean arrival time of 2nd and 3rd batches of jobs are

Table 3.1: Batch job arrival time

Batch No.	1	2	3
Mean Arr. Time $E[t_j^a]_{j=1}^{j=N}$	0	45	90
StDev Arr. $StD[t_j^a]_{j=1}^{j=N}$	0	3	3

45 and 90, their standard deviations are all 3. As $StD(X : U[a, b]) = \frac{(b-a)^2}{12} = 3$, which implies the range $|b - a| = \sqrt{3 \times 12} = 6$. In discrete time domain, the period of one unit slot is 1 minute. So the 6 jobs in 2nd batch can arrive at any time between $[42, 48]$, while for the 3rd batch it is $[87, 93]$.

For the 1st batch, it is assumed to be deterministic scheduling, s.t. all jobs are ready to start from 0.

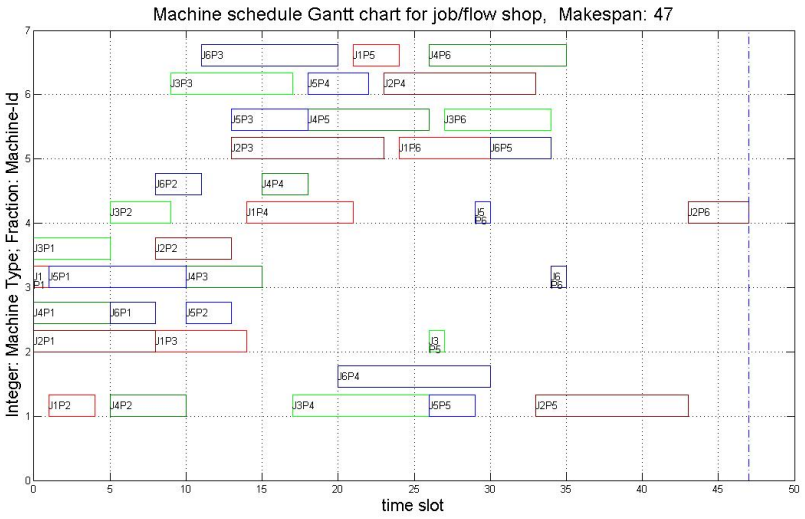
The solution is illustrated in Fig.3.3 to Fig.3.5.

Fig. 3.3 is the machine schedule for 1st and 2nd batch of jobs. They are represented by Gantt chart, which is rectangular ones. Fig. 3.4-(a) is the moving window machine schedule to handle 3rd batch of jobs. By including all 3 batches, Fig. 3.4-(b) is the Gantt chart of machine schedule for all 3×6 jobs.

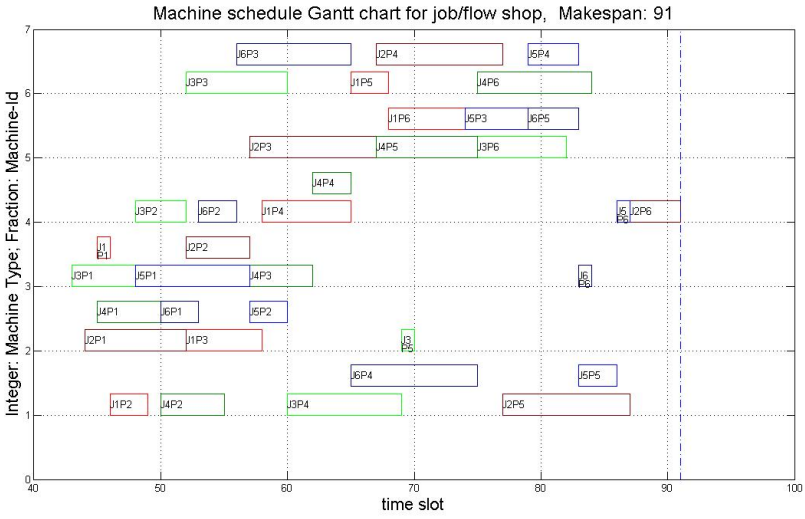
Finally in this section, Fig. 3.5 compares the rectangular Gantt chart in Fig. 3.5-(a) and circular chart in Fig. 3.5-(b). The circular chart is originally proposed by Fua in [11], which is used to represent robotic's job schedule.

Just like the Gantt chart, circular chart can represent both job schedule and machine-dispatching schedule. However, here we only have one type in dynamic schedule. The Fig. 3.5-(b) is machine-dispatching schedule. Radius direction $[0, +\infty)$ is time-line, theta direction $[0, 2\pi)$ is machine type in the 6 main sectors, and machine-id in the fractional sections.

In dynamic schedule of job-arrival, the incoming jobs can be more and more while the machine environment is relatively fixed (w.r.t. total machine types and machine

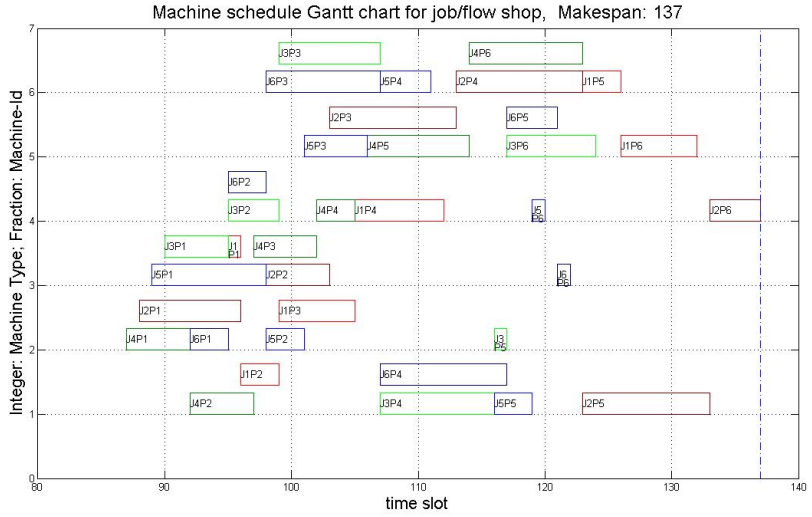


(a) 1st batch of jobs

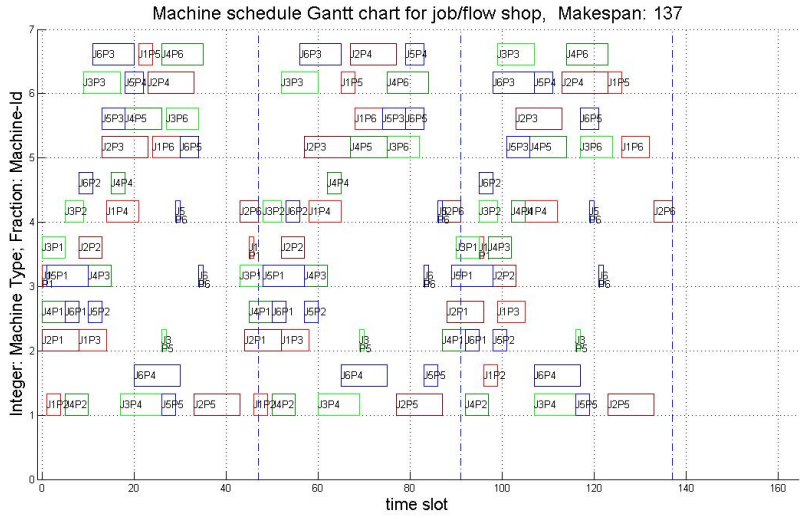


(b) 2nd batch of jobs

Fig. 3.3: Machine Schedule, moving window Gantt chart

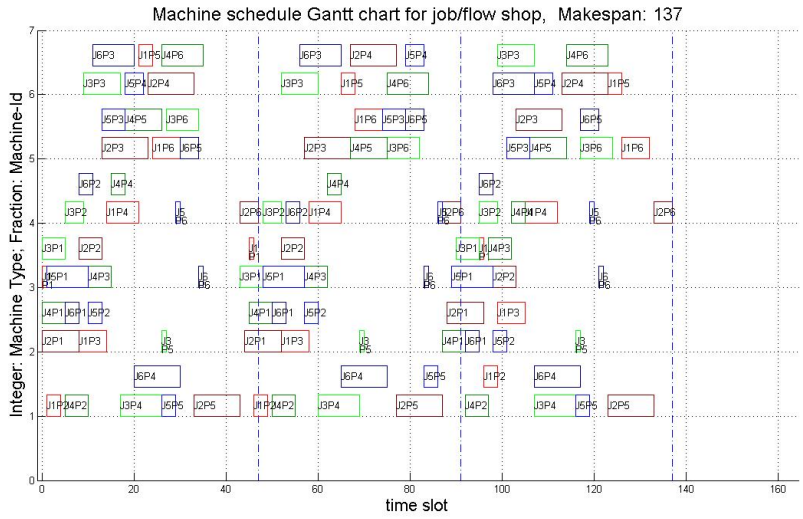


(a) Moving window for 3rd batch of jobs

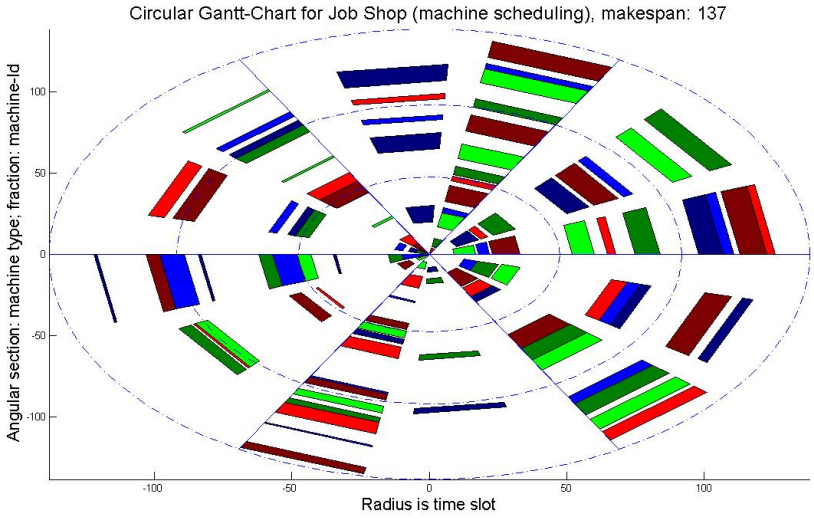


(b) All three batches of jobs

Fig. 3.4: Machine Schedule, moving window Gantt chart and total schedule chart



(a) Gantt chart



(b) Circular chart

Fig. 3.5: Machine Schedule, rectangular Gantt chart v.s. Circular chart

capacity of each type). That is why only the machine-dispatching is used in circular chart. On the other hand, when total number of jobs are fixed, while machine(or robot) settings are changing, perhaps job-schedule circular chart is preferred.

For such case that each machine type has multiple machines, we follow the similar convention in rectangular Gantt chart, s.t. the full circle 2π is divided into K sector pieces, each with $\frac{2\pi}{K}$ for one machine type. For each sector, it is further divided into C_k of pieces for each machine-id. In above Fig. 3.5-(b), $K = 6$ and $C_k = 2, \forall k \in \{1, 2, 3, 4, 5, 6\}$.

By comparison of rectangular Gantt and circular chart, it is clearly that in circuit chart the most recent jobs are largest in size. So the most recent jobs have the best focus.

3.4 Handling Machine Degradation/Upgradation with Adjusted CH (in Sec. 2.3.2)

While the machine breakdown is well-studied, the dynamics of machine degradation and up-gradation are not so. Some examples of machine degradation and or up-gradation are following:

- Machine running speed is changing for the variation of environment, i.e. cars' running speed is lower down in the raining or snowing situations;
- Machine power is changing, i.e. 1 robot is designed to carry up to 100 kg load but somehow it can only carry 80 kg load after it has been used for 5 years;
- A computation center is designed to have 10 workstations (a kind of super PC). Since the size of a PC is reduced and cooling is improved, the same area of the computation center may hold 15 workstations after 5 years of operation.

- My computer is bought with 1 GB RAM, it may be upgraded to 2 GB RAM. So for normal software applications, it is faster and the OS can hold more applications.
- and etc.

In my thesis, we follow the same problem setting as in Sec. 2.3.2.

- There are multiple, exchangeable machines for each type of process;
- Each machine can take no more than one job at any time, i.e. a truck can carry exactly one container;
- The machine capacity (total number of available machines) is changing from period to period;

Above assumption is reasonable. It is quite often that in a container terminal port, some trucks are not available for the reasons of maintenance, operator absenteeism or etc. In this situation, the LUT (Look-Up Table) representation is proposed to formulate the machine capacity. Then my previous algorithm (RH and CH in Sec. 2.3.2) can be extended to handle degradation and up-gradation. Although both RH and CH in 2.3.2 can be extended, we focus on extending CH for its faster computation.

3.4.1 A LUT(Look-Up Table) Representation of Machine Capacity

Similar to the construction of time-cell matrix in Sec. 2.2.1, the machine capacity is listed together with the starting time of a particular period. The reason of using this method is that it consumes very little memory for storage. The memory consumption is linear to the total number of variations.

When machine capacity profile is mapped with time point, a look-up-table is generated. For each type of machine, it is composed of 1 integer and 2 arrays.

Table 3.2: Machine Capacity Profile for M2, total num. of variation = 5

Time Slot	0	12	18	24	36	48
Mach. Cap. C	4	3	2	5	7	2

Table 3.3: Machine Capacity Profile for M3, total num. of variation = 3

Time Slot	0	10	20	40
Mach. Cap.	3	2	3	2

- Integer: length of array N_v (total number of variations + 1), if there is no variation (machine capacity is constant along the time), it is 1;
- Array of Time Slot TS: indicating the starting time of the corresponding machine capacity;
- Array of Machine Capacity C: machine capacity, mapped with above time slot;

In order for clearly understanding, let us take an example from a flow shop agent with 3 kinds of machines. M2 and M3 has time-dependent capacity profile. The LUT-based machine capacity profile for M2 and M3 are listed in Tables 3.2 and 3.3.

In Table 3.2, there are totally 5 times of variation (degradation or up-gradation). From very beginning to time slot-12 (not inclusive), there are 4 machines. From time slot-12 (inclusive) to time slot-18 (not inclusive), there are 3 machines, ... From time slot-48 onwards, there are 2 machines.

3.4.2 Adjusting CH with LUT representation of machine capacity

In Sec. 2.3.3, the condition of machine capacity violation is defined to be

$$\max_t M_{kt} > C_{k,F_\tau}, \exists F_\tau, k \in \{1, 2, ..., K\}$$

Above F_τ is a set of time slot. To enumerate all time slots in F_τ will take more computation effort than LUT-based method.

For LUT representation, there is the machine capacity violation, if and only if:

$$\max_t M_{kt} > C_{k,[TS_i,TS_{i+1})}, t \in [TS_i, TS_{i+1}), i \in \{1, ..., N_v\}, k \in \{1, 2, ..., K\}$$

Just by replacing the above machine capacity violation into *CH4* of the Constructive Heuristic in Sec. 2.3.3, it will be able to handle the LUT-based machine degradation or up-gradation. So it is clear that handling machine capacity variation in LUT format is just a special case of the generalized CH algorithm. However, consider the real-time requirement of dynamic scheduling, the reactive scheduling must be very fast. This is another reason we use LUT-based method, beside the consideration of memory consumption.

3.4.3 Machine Dispatching Rules

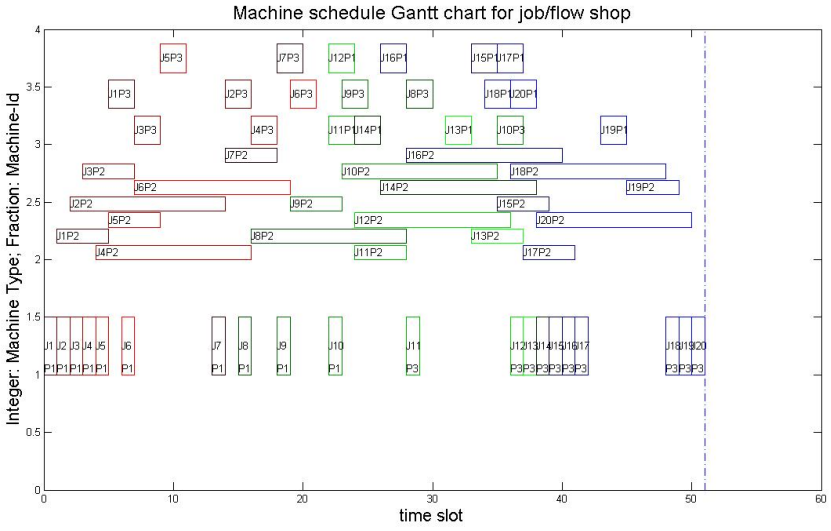
We considered 2 kinds of machine dispatching rules.

1. Round-robin rules, all machines have the same priority.

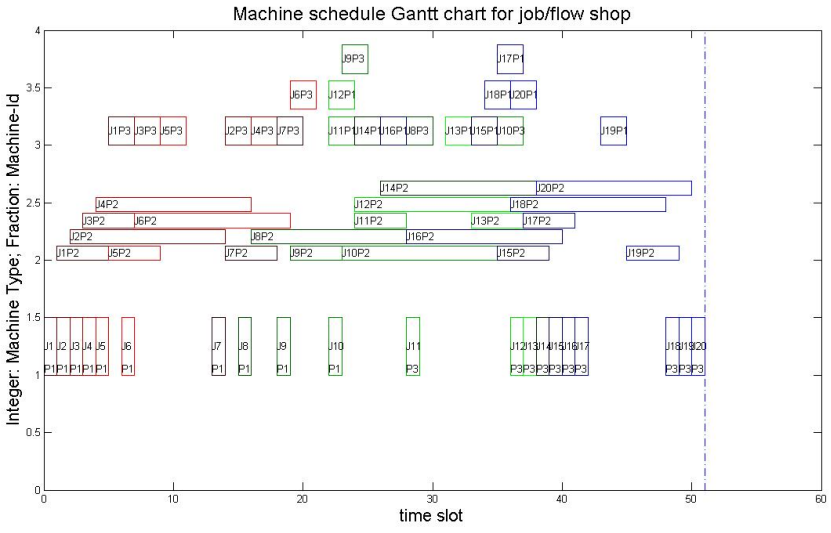
2. Priority based rules. All machines are sequenced by utilization priority, i.e. lower index with higher priority. Each time for dispatching machine, search from higher to lower priority. If any machine is available, it will be used, otherwise, go to next machine.

Both dispatching solutions are valid, w.r.t. the machine capacity constraints. For the above machine capacity profile in Tables 3.2 and 3.3, the machine dispatching with both rules are listed in Fig. 3.6.

Further, the feasibility of above schedule can be verified. Both machine dispatching schedules are mapped to the same machine utilization profile and job schedule, which are shown in Fig. 3.7.

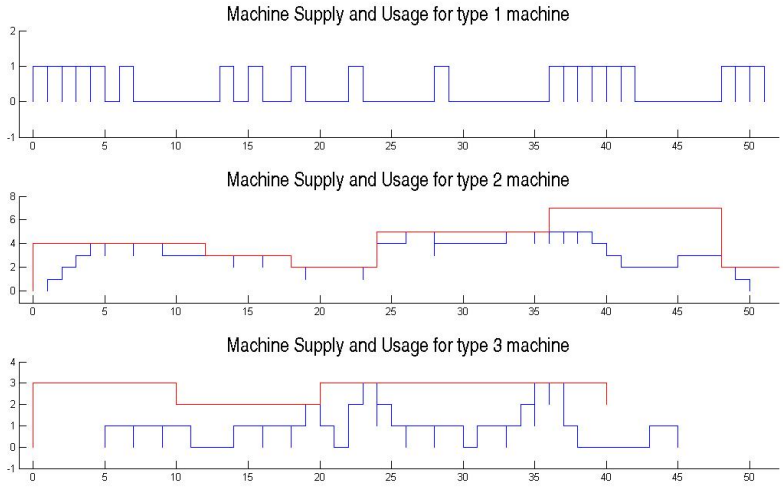


(a) Machine Dispatching by Round Robin

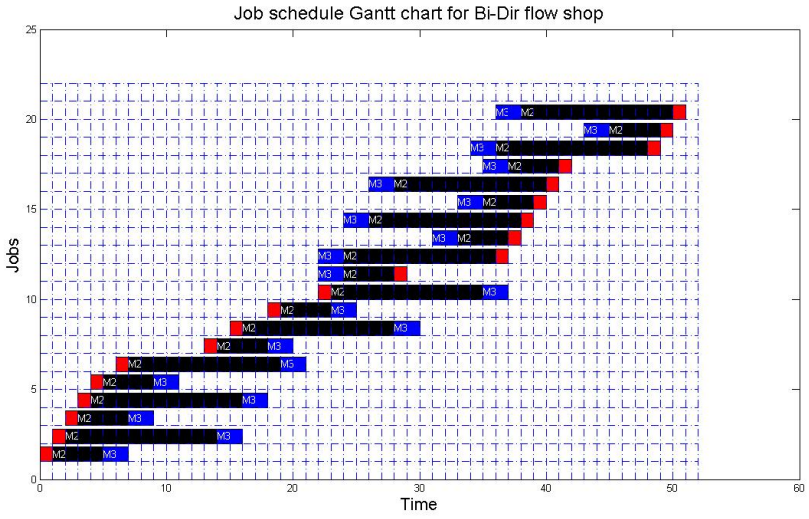


(b) Machine Dispatching by Priority Sequence

Fig. 3.6: Schedule for Machine Dispatching in Degradation/Upgradation



(a) Machine supply and usage



(b) Job scheduling

Fig. 3.7: Feasibility Check, Schedule with Machine Degradation/Upgradation

Regarding the solution, there are several issues to take note:

- In Fig. 3.6-(a), it is clear that all machine are similarly utilized, it is just the property of round-robin, i.e. no machine has higher priority than others. While in Fig. 3.6-(b) it is clear that the machine with smaller-Id has higher utilization. This dispatching just matches the designed priority-based rule.

- In Fig. 3.7-(a), the red color is machine capacity, which is stated in Tables 3.2 and 3.3. The blue color is machine usage. By comparison of the machine capacity and usage, it is clear that it is a feasible solution.

- In Fig. 3.7-(a), it is noticed that the usage for M3 does not hit the capacity through all the periods. This is because of the machine complementarity. One machine's utilization depends on the other machine's schedule.

- From Fig. 3.7-(b), please pay attention that it is bi-directional no-wait flow-shop with COS constraints. The COS constraints will be relaxed in next chapter, so the same algorithm can be applied to general bi-directional no-wait flow-shop scheduling.

- This LUT based representation of machine capacity profile will be used in next chapter, multiple-period resource allocation.

- Here we do not verify the optimality of above heuristic, because dynamic scheduling focus on fast feasible solutions. The optimality is verified in next chapter in the whole problem of resource allocation.

Chapter 4

Resource Allocation and an Adaptive Approach in Price Adjustment

In this chapter, we are going to consider multiple agents, each of whom is handling a job-lists $l \in \{1, 2, ..., L\}$. All the agents are sharing same pool of resources. Each job list has its own start time R^l, due time D^l, delay penalty W_d^l, and makespan price W_m^l. The problem is how to allocate resources to each agent and how to schedule each job-list with the allocated resources. In the real application, the total time slots are usually partitioned into several periods (time frames or hours, $\{\mathcal{T}_\tau : 1 \leq \tau \leq F\}$), resource allocation is done for multiple items on multiple periods.

Especially, our contributions are the concept of utility price and an adaptive price adjustment strategy. The utility price is studied both in theory and by numerical experiments. The fast bid-generation algorithm is based on previous study of CH and greedy heuristics. Combined with the pre-processing and post-processing steps, the power of the

entire system (Adaptive Multi-period Resource Allocation) is demonstrated in solving large-scale decentralized scheduling problems.

4.1 Problem Definition with a Sample Instance

We are concerned with the problem of allocating machine resources to multiple agents. Each agent, without loss of generality, is assumed to be solving a multi-machine flow-shop problem where jobs comprise multiple operations requiring distinct machine types.

Notationally, each agent $l = 1, 2, \cdots, L$ is associated with a job-list, and is associated with the following attributes:

- R^l: release time;
- D^l: due time;
- W_m^l: makespan price (i.e. weight factor for makespan); and
- W_d^l: tardiness penalty (i.e. weight factor for delay/tardiness).

All agents share a common pool of K different types of machine resources available over $\mathcal{T}$ periods ($\tau : \tau = 1, 2, \cdots, \mathcal{T}$), and the resource capacity is denoted as $S_{k\tau} : k = 1, \cdots K, \tau = 1, \cdots, \mathcal{T}$. Without loss of generality in this thesis, it is assumed that each agent objective function is to minimize the sum of weighted total makespan and tardiness. Let $\mathcal{M}(\mathbf{U}^l)$ denote agent makespan and tardiness is defined as $\max\{0, \mathcal{M}(\mathbf{U}^l) + R^l - D^l\}$.

As an example, we give a problem instance, comprising 4 agents (or simply 4 job-lists), each with 10 jobs on 2 types of machines. The basic information is found in Table 4.1 while the detailed job-list information is in Table 4.2. (This instance will again be used in the Section 4.11.1 on Experimental Results.) Table 4.1 gives the work

Table 4.1: Sample Problem for 4 agents and 2 machines

Agent -Id	Release Time	Due Time	Make-span Price	Tardiness Penalty	Work Load Total Time	
					Machine Type1	Machine Type2
1	8:00	10:00	100	500	68	20
2	8:00	10:00	200	600	74	20
3	9:00	11:00	100	500	86	20
4	9:00	12:00	250	800	58	20
$S_{k\tau}$ $\forall\tau$	Machine 1 16			Machine 2 8		
B_k^l $1 \leq l \leq 4$	Machine 1 2			Machine 2 1		

Table 4.2: Processing time for the 4 agents in Sample Problem

Job Id	Processing Time for Agents 1/2/3/4	
	Machine-1	Machine-2
F-1	[5/7/9/3]	[2/2/2/2]
F-2	[9/8/9/10]	[2/2/2/2]
F-3	[5/7/7/3]	[2/2/2/2]
F-4	[9/8/7/10]	[2/2/2/2]
F-5	[5/7/9/3]	[2/2/2/2]
F-6	[9/8/9/10]	[2/2/2/2]
F-7	[5/7/7/3]	[2/2/2/2]
F-8	[9/8/7/10]	[2/2/2/2]
F-9	[5/7/9/3]	[2/2/2/2]
F-10	[9/8/9/10]	[2/2/2/2]

load statistic, which will be used as job-list priority in the heuristics to be presented subsequently.

In this instance, we take 1 hour as one time period, and a feasible solution is found in Table 4.4 at Section 4.11. With reference to the makespan price and tardiness penalty in Table 4.1, it is clear that the solution is reasonable, since the higher importance of the job-list, the more resources are allocated to the respective agents. For agents with equal price and tardiness penalty, the agent with higher work load will have more resources. This solution is derived by the auction approach, details of which will be explained in the later sections.

Above sample problem is abstracted from a container-port operation. It has the following features:

- One job is done by 3 operations and there is no wait allowed between consecutive operations;
- 3 operations are done be {quay crane, truck, yard crane} and they are non-preemptive;
- Each job has a release time and due time. All jobs must be finished, a tardiness penalty will be incurred if it is done after due time; and
- A quay crane is assumed as the agent while trucks and yard cranes are common resources to be shared.

My previous work [4] gives the detailed explanation of model abstraction, now we extend this work in the following aspects:

- The different agents may have different job release times, job due times, makespan cost rates and tardiness penalty;

- The COS constraints can be relaxed, so the mathematical model is for non-preemptive flow-shop (or bi-directional flow-shop) without wait; and
- A Decentralized method (auction) is applied for better solution quality.

Although the auction method can adapt to dynamic resource allocation, we focus on static allocation problem in this thesis and only demonstrate the computational efficiency.

4.2 An Integrated IP Formulation for Resource Allocation

The 0-1 integer problem formulation was originally from Pritsker [109] and lately studied by Kutanoglu. Both of them, however, focused on one-machine problem. This model is extended to multiple machine scheduling in [22] and further extended to multiple periods in [4]. Different from the previous work related with container-port operation, the model here is applied for general flow-shop and/or bi-directional flow-shop.

This model is found to be a genetic formulation for non-preemptive multi-machine shop-scheduling and multi-period resource allocation problems. Besides the non-preemptive assumption, this thesis considers following assumptions from the application background:

1. There is no wait between consecutive operations(or tasks) of a job; and
2. Each machine can handle exactly one job at a time.

The model parameters are explained in the following list

- Partition of total time horizon $\{1, 2, ..., \tilde{T}\}$ into F time frames, $\{\mathcal{T}_\tau : 1 \leq \tau \leq F\}$, all the agents observe the same time partition;

$$\textbf{ResAlloc-MIP: minimize} \sum_{l\geq 1}^{L} MTC^l \tag{4.1}$$

$$\text{subject to: } X^l_{ijt} - X^l_{i,j,t+1} \leq 0, \qquad \forall l,i,j,t \in \{1,2,...,\tilde{T}-1\}. \tag{4.2}$$

$$\left\{ \begin{array}{ll} X^l_{i,j,t} - X^l_{i,j-1,t-t^l_{i,j-1}} & \text{if } t > t^l_{i,j-1} \\ X^l_{i,j,t} & \text{if } t \leq t^l_{i,j-1} \end{array} \right\} = 0, \forall l,i,j \in \{2,...,o_i\}. \tag{4.3}$$

$$\sum_{i,j:m_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > t^l_{ij} & (X^l_{ijt} - X^l_{i,j,t-t^l_{ij}}) \\ \text{if } t \leq t^l_{ij} & X^l_{ijt} \end{array} \right\} - U^l_{k,\mathcal{T}_\tau} \leq 0, \forall l, t \in \mathcal{T}_\tau,$$

$$\forall k \in \{1,2,...,K\} \tag{4.4}$$

$$X^l_{ijt} - X^l_{0,t} \leq 0, \forall l,i,j,t \in \{1,2,...,\tilde{T}\}. \tag{4.5}$$

$$X^l_{ijt} - X^l_{N+1,t} \geq 0, \forall l,i,j,t \in \{1,2,...,\tilde{T}\}. \tag{4.6}$$

$$X^l_{0,R^l-1} = 0, \forall l. \tag{4.7}$$

$$X^l_{N+1,\tilde{T}} = 1, \forall l. \tag{4.8}$$

$$\sum_{l\geq 1}^{L} U^l_{k,\mathcal{T}_\tau} \leq C^M_{k,\mathcal{T}_\tau}, \forall \mathcal{T}_\tau, k. \tag{4.9}$$

$$W^l_d \cdot \sum_{t>D^l-t^l_{N,o_N}} (1 - X^l_{N,o_N,t}) +$$

$$W^l_m \cdot \sum_t (X^l_{0,t} - X^l_{N+1,t}) = MTC^l \tag{4.10}$$

$$X^l_{ijt} \in \{0,1\} \qquad \forall l,i,j,t \in \{1,2,...,\tilde{T}\}. \tag{4.11}$$

$$U^l_{k,\mathcal{T}_\tau} \in \mathcal{N}, \qquad \forall k, \mathcal{T}_\tau \tag{4.12}$$

$$\textbf{BidGen-l minimize } MTC^l + \sum_{k,\tau} p_{k,\mathcal{T}_\tau} U^l_{k,\mathcal{T}_\tau} \tag{4.13}$$

$$\text{subject to: } X^l_{ijt} - X^l_{i,j,t+1} \leq 0, \qquad \forall i,j,t \in \{1,2,...,\tilde{T}-1\}. \tag{4.14}$$

$$\left\{ \begin{array}{ll} X^l_{i,j,t} - X^l_{i,j-1,t-t^l_{i,j-1}} & \text{if } t > t^l_{i,j-1} \\ X^l_{i,j,t} & \text{if } t \leq t^l_{i,j-1} \end{array} \right\} = 0, \forall i,j \in \{2,...,o_i\}. \tag{4.15}$$

$$\sum_{i,j:m_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > t^l_{ij} & (X^l_{ijt} - X^l_{i,j,t-t^l_{ij}}) \\ \text{if } t \leq t^l_{ij} & X^l_{ijt} \end{array} \right\} - U^l_{k,\mathcal{T}_\tau} \leq 0, \forall t \in \mathcal{T}_\tau, \forall k \tag{4.16}$$

$$X^l_{ijt} - X^l_{0,t} \leq 0, \forall l,i,j,t \in \{1,2,...,\tilde{T}\}. \tag{4.17}$$

$$X^l_{ijt} - X^l_{N+1,t} \geq 0, \forall l,i,j,t \in \{1,2,...,\tilde{T}\}. \tag{4.18}$$

$$X^l_{0,R^l-1} = 0. \tag{4.19}$$

$$X^l_{N+1,\tilde{T}} = 1, \forall i,j. \tag{4.20}$$

$$W^l_d \cdot \sum_{t>D^l-t^l_{N,o_N}} (1 - X^l_{N,o_N,t}) +$$

$$W^l_m \cdot \sum_t (X^l_{0,t} - X^l_{N+1,t}) = MTC^l \tag{4.21}$$

$$X^l_{ijt} \in \{0,1\} \qquad \forall i,j,t \in \{1,2,...,\tilde{T}\}. \tag{4.22}$$

$$U^l_{k,\mathcal{T}_\tau} \in \mathcal{N}, \qquad \forall k, \mathcal{T}_\tau \tag{4.23}$$

- $t^l_{i,j}, l \in \{1,2,...,L\}, i \in \{1,2,...,N\}, j \in \{1,2,...,o_i\}$: Processing time of operation j in job i for job-list l;
- $D^l, l \in \{1,2,...,L\}$: Due time of job-list l;
- W^l_d: Delay penalty of job-list l;
- W^l_m: Makespan price for job-list l; and
- $C^M_{k,\mathcal{T}_\tau}$: maximum capacity constraints of machine type k in time frame $\mathcal{T}_\tau$.

The complete model, listed from (4.1) to (4.12), is to minimize the sum of weighted makespan cost and tardiness penalty (4.1) under the constraints { (4.2), (4.3), (4.4), (4.5), (4.6), (4.7), (4.8), (4.9), (4.10), (4.11), (4.12) }.

Constraints (4.2) assumes that all tasks are non-preemptive.

Constraints (4.3) states the task sequence of a job. Here we use equality for the assumption of no wait between tasks. By replacing the $=$ to be $\leq$, it will be a formulation s.t. wait is allowed between consecutive tasks.

Constraints (4.4) states the machine utilization. In order to handle flexible machine environment, a fractional job size s_{ij} could be introduced as a predefined parameter, such that each machine type m_{ij} could handle $\frac{1}{s_{ij}}$ jobs at the same time. Then, equation (4.16) is replace to be the following forms.

$$\sum_{i,j:m_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > t^l_{ij} & s_{ij}\left(X^l_{ijt} - X^l_{i,j,t-t^l_{ij}}\right) \\ \text{if } t \leq t^l_{ij} & s_{ij}X^l_{ijt} \end{array} \right\} - U^l_{k,\mathcal{T}_\tau} \leq 0$$

However in the application of this thesis, we have $s_{ij} = 1$, which means that each machine can handle no more than one job at any time. For agent l, there are $U^l_{k,\mathcal{T}_\tau}$ items of machine type k in the period of $\mathcal{T}_\tau$.

Table 4.3: Problem Size Comparison

Comparison l Items	Scheduling in Continuous Time	Scheduling in Discrete Time	Integrated Model Allocation and Scheduling in Discrete Time
Number of Variables	$1 + 2 \cdot \sum_{1 \le i \le N} o_i$	$\tilde{T} \cdot \sum_{1 \le i \le N} o_i$	$L \cdot \tilde{T} \cdot \sum_{1 \le i \le N} o_i + L \cdot K \cdot \mathcal{T} + L$
Variable Type	All Floating non-negative real numbers	All 0-1 Integers	$\mathcal{T} \cdot K \cdot L$ positive integers $+ L$ floating $+$ $+ L \cdot \tilde{T} \cdot \sum_{1 \le i \le N} o_i$ 0-1 integers
Number of Constraints	$2 \cdot \sum_{1 \le i \le N} o_i + K - N - 1$	$\tilde{T} \cdot \sum_{1 \le i \le N} o_i + K \cdot \tilde{T}$	$L \cdot \tilde{T} \cdot \sum_{1 \le i \le N} o_i + L \cdot K \cdot \tilde{T} + 2L$ $+L \cdot \sum_{1 \le i \le N} o_i + \mathcal{T} \cdot K$
Problem Structure	Linear Problem with Look-Up Table functions	Pure Integer Problem	Mixed Integer Problem
Solution	heuristic only	Solvable in exponential time	Solvable in exponential time

Constraint (4.5) formulates the dummy start operation, and Constraints (4.6) formulates the dummy end operation.

Constraint (4.7) states that the job list cannot start until the it is released.

Constraint (4.8) states that all jobs must be finished at the end. It is a redundent constraint just to prevent any singularity happens.

Constraint (4.9) states the the sum of machine quotas by all job-lists should be within the capacity limit. It is this constraint that links all L scheduling problems together. So by relaxing this constraints, the problem can be decentralized.

4.3 Lagrangian Relaxation of Machine Capacity Constraints

Let us study the problem size in above complete formulation as in previous Section 4.2. we will make an comparison with another continuous model and a decomposed scheduling problem.

In Table 4.3, the problem size of discrete time formulation is compared with that of continuous time [4]. Even though the continuous time formulation is much smaller than the discrete time formulation, it is still an open problem to solve, since there are δ functions within the model which have not been solved so far.

It is also noticed that when the problem size is not very large (i.e. less than 10,000

constraints and 10,000 variables) current commercial solvers (i.e. CPLEX, MOSEK) are quite efficient. For the sample problem proposed in my previous work [4], a system with 4 agents and each agent has 10 jobs and 30 operations, the complete IP model (ResAlloc-MIP) has 16,000 constraints and 7,900 variables. It takes around 5 minutes to solve by MOSEK or around 1 minute to solve by CPLEX. The measurement is on a computer with Pentium IV CPU at 3GHZ and 1GB memory. However, for larger problems i.e. 4 agents and each has 40 jobs and 60 operations, even CPLEX itself cannot return a feasible solution within 5 hours.

Now we turn to an agent-based approach - auction, which is not targeting at the optimal, but to give a near optimal solution within much shorter time. What's more, this approach has a characteristic of problem decomposition and each sub-problem could be solved by different agents. In this way, parallelism is possible.

Let us consider the problem of BidGen-l defined for each agent, which is to minimize weighted makespan cost, tardiness cost and resource cost (4.13) under the constraints of { (4.14), (4.15), (4.16), (4.17), (4.18), (4.19), (4.20), (4.22), (4.23) }.

By comparing problems of BidGen-l and ResAlloc-MIP, the machine capacity constraints 4.9 in problem ResAlloc-MIP doesnot exist in problem BidGen-l, while the object function of BidGen-l includes one more term $\sum_{k,\tau} p_{k,\mathcal{T}_\tau} U^l_{k,\mathcal{T}_\tau}$, which is the resource cost. What's more, price of resource at time periods $\{p_{k,\mathcal{T}_\tau} \geq 0 | k = 1, 2, ..., K, \tau = 1, 2, ..., F\}$ are decision parameters newly introduced in BidGen-l but not exist in ResAlloc-MIP.

Proposition 4.3.1 *Optimal Solution of L problems of BidGen-l with respect to* $\{p_{k,\mathcal{T}_\tau} > 0 | k = 1, 2, ..., K, \tau = 1, 2, ..., F\}$ *will contribute to a lower bound of optimal solution to problem ResAlloc-MIP. And this lower bound solution is closer to optimal than linear problem solutions.*

Proof: According to [91], $Z_L \leq Z_D \leq Z_{IP}$, where Z_L is the optimal value of linear problem relaxation of the original Integer Problem ResAlloc-MIP and Z_D is the optimal value of *Lagrangian dual* problem of ResAlloc-MIP.

Consider the problem of **RlxResAlloc**, which is relaxing the machine capacity contraints (4.9) of problem ResAlloc-MIP.

$$\begin{array}{ll} & \text{RlxResAlloc} \\ \text{minimize} & \sum_{l\geq 1}^{L} MTC^l + \sum_k \sum_\tau p_{k,\mathcal{T}_\tau} \left(\sum_{l\geq 1}^{L} U^l_{k,\mathcal{T}_\tau} - C^M_{k,\mathcal{T}_\tau} \right) \\ \text{subject to} & \{(4.2),(4.3),(4.4),(4.5),(4.6),(4.7),(4.8),(4.10),(4.11),(4.12)\} \end{array}$$

The optimal value of RlxResAlloc is noted as a function of price: $Z(\mathbf{p})$, then the Z_D and *Lagrangian dual* problem is just as follows:

$$Z_D = \max_{\mathbf{p}\geq 0} Z(\mathbf{p}) \tag{4.24}$$

$$\begin{aligned} & \sum_{l\geq 1}^{L} MTC^l + \sum_k \sum_\tau p_{k,\mathcal{T}_\tau} \left(\sum_{l\geq 1}^{L} U^l_{k,\mathcal{T}_\tau} - C^M_{k,\mathcal{T}_\tau} \right) \\ = & \sum_{l\geq 1}^{L} \left(MTC^l + \sum_k \sum_\tau p_{k,\mathcal{T}_\tau} U^l_{k,\mathcal{T}_\tau} \right) - \sum_k \sum_\tau p_{k,\mathcal{T}_\tau} C^M_{k,\mathcal{T}_\tau} \end{aligned}$$

Note that $MTC^l + \sum_k \sum_\tau p_{k,\mathcal{T}_\tau} U^l_{k,\mathcal{T}_\tau}$ is exactly the objective function (4.13) in problem BidGen-l. And when $p_{k,\mathcal{T}_\tau} \geq 0$ is a given parameter for problem of RlxResAlloc, $\sum_k \sum_\tau p_{k,\mathcal{T}_\tau} C^M_{k,\mathcal{T}_\tau}$ is a constant.

So the optimal solution in L problems of BidGen-l with respect to $\{p_{k,\mathcal{T}_\tau} > 0 | k = 1,2,...,K, \tau = 1,2,...,F\}$ will contribute to Z_D which is a lower bound to Z_{IP} and closer than Z_L. **Q.E.D.**

Above proof also gives the relation between ideal auction approach solution and the exact solution to original problem. We are most interested at the problem decomposition

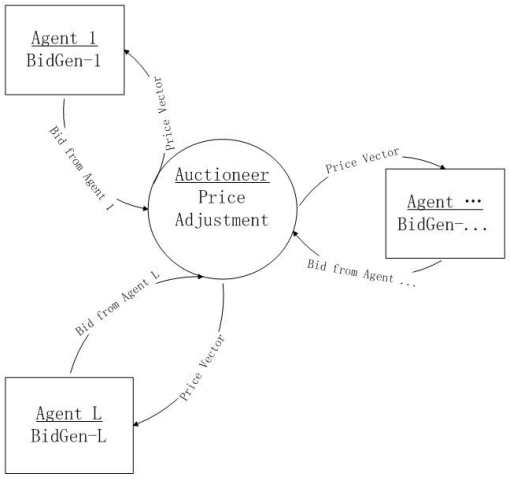


Fig. 4.1: Auction Structure

characteristic of this approach. From the implementation point of view, we have the following assumptions:

1. There is communication access between every agent and the auctioneer, but there is no channel between any pair of agents;

2. Each agent honestly minimizes its overall cost including makespan tardiness cost and resource cost;

3. Resource price is defined on each particular resource on specific time period, is released only from the auctioneer; and

4. The communication bandwidth between each agent and the auctioneer is high enough for data transfer. The time for communication can be ignored in comparison with the total computation time.

Based on above assumption, the auction structure is shown in Figure 4.1.

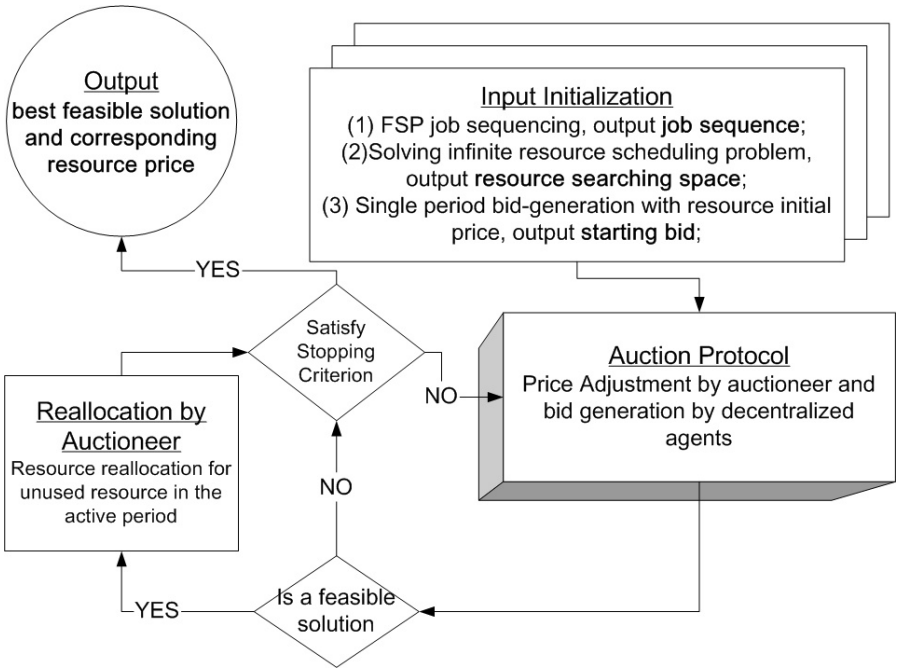


Fig. 4.2: Overall System Flow

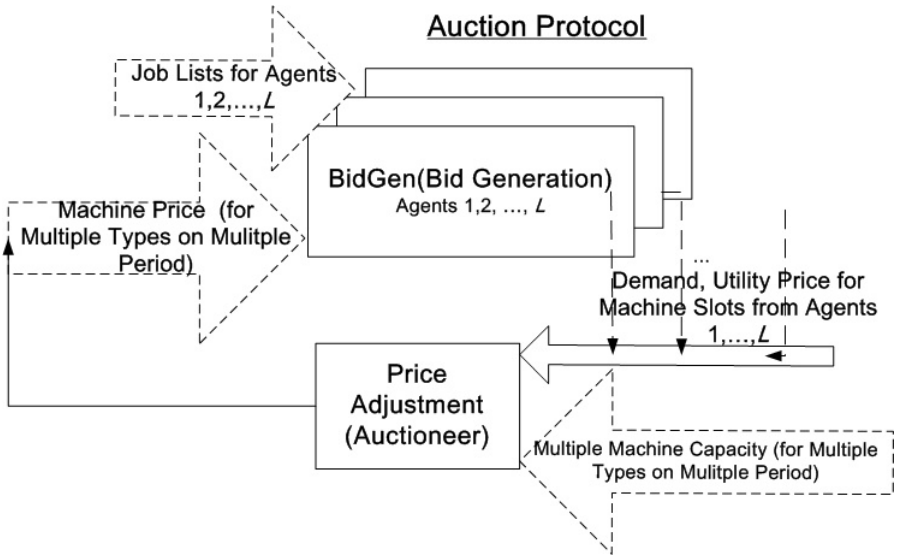


Fig. 4.3: Basic Structure of Auction Protocol

4.4 Solution by Auction Approach and Price Adjustment

In this section, the overall system architecture is presented and the proposed auction protocol.

A diagram of the system architecture is shown in Fig. 4.2.

The proposed system architecture comprises three modules:

- System initialization, which will perform initialization and trigger the auction (details below);
- Auction protocol, which includes the bid-generation and price adjustment procedures. Bid-generation will be discussed in detail in Section 4.5 while price adjustment, in Section 4.6. For the latter, the ideas of utility prices and variable step-size will be separately discussed in sections 4.7 and 4.8 respectively; and
- Resource reallocation, which will be triggered each time a feasible solution has been found. Intuitively, this is a post-processing step that improves the quality of the feasible solution. Details will be discussed in Section 4.10.

4.4.1 Initialization and Stopping Criteria

In the initialization step, there is a preprocessing performed in order to improve the subsequent bid generation and price adjustment procedures. First, we fix the job sequences that agents will subsequently employ in their bid generation (which assumes the job sequence to be given). For this purpose, each agent will apply the genetic algorithm (GA) proposed in [35]. Second, we determine the upper bound resource capacities that will be required by the agents. This is carried out by solving a centralized flow-shop problem with infinite resources, using the algorithm proposed in [4]. This will enable us

to obtain the upper bound profile of resource requirements by the agents which sets the upper resource limits for price adjustment. Based on the resource profile, the resource prices are then initialized by performing a single period bid generation procedure (see details in Section 4.5 below).

Upon initialization, the auction protocol will be triggered, and it will be executed until any of the following stopping criteria is satisfied.

- An equilibrium solution has been achieved, which means there is a feasible solution under a particular resource price vector and this price vector does not change for a prescribed number of iterations. This is a standard stopping criterion used by a tatonnement auction.
- Maximum number of iterations has been performed.

The system will then return the best feasible solution found. The reader may note that the best feasible solution may not be the last solution found when an equilibrium solution has been reached. Nevertheless, in order to terminate the auction process quickly, it is important to be able to obtain an equilibrium solution quickly.

4.4.2 Overview of Auction Protocol

The proposed auction protocol is given in Figure 4.3. It comprises mainly two components: (a) Bid Generation, and (b) Price Adjustment. The entire auction process will occur in rounds, as shown in Figure 4.3. Within each round, all agents will perform Bid Generation to generate their respective bids; then, all bids are submitted simultaneously to the auctioneer who will perform price adjustment. The process will be repeated until a price equilibrium has been achieved, i.e. when supply matches demands, which means a feasible solution has been found. Once equilibrium has been achieved, a greedy step

is perform to improve the quality of the solution as follows. We assigned un-utilized resources to agents in non-increasing order of their MTC value until either the resource pool is empty or no more resources are needed to improve the agent's MTC value.

In the following, the resource allocation problem is casted in the auction context:

On the agent's end, each agent l has to decide on its bid

$$\mathbf{U}^l = \{U^l_{1,1}, U^l_{2,1}, \cdots, U^l_{K,1}, \cdots, U^l_{k,\tau}, \cdots, U^l_{1,\mathcal{T}}, \cdots, U^l_{K,\mathcal{T}}\}$$

for machine type k at period τ for all k and τ. This is the Bid Generation ($BidGen$) problem which is an optimization (or multi-machine flow-shop scheduling) problem that minimizes the total cost made up three components makespan, tardiness penalty and resource cost.

On the auctioneer end, price adjustment is performed at each auction iteration as follows:

- Consolidate all agent bids and compute the aggregate demand $D_{k\tau}$ for all machine type k at time slot τ;
- By comparing $D_{k\tau}$ and supply capacity $S_{k\tau}$, the new price is calculated and announced to all bidders;
- Announce to start a new round of BidGen, or stop the auction according to the stopping criterion.

Kutanoglu and Wu [63] observed that the above iterative combinatorial auction protocol is analogous to Lagrangian relaxation [93] for the case of job shop scheduling, and hence inherits all properties related to it (including convergence). More precisely, they

showed that the iterative process for finding the price vector λ in Lagrangian relaxation actually corresponds to price adjustment in an iterative combinatorial auction, the mapping is described as follows:

- The auctioneer initializes the prices for the machine resource time slots λ^0.
- At auction round r, each bidder agent performs Bid Generation by solving its local scheduling problem using the prevailing resource prices λ^r in its objective function, and submits the resulting bid to the auctioneer. This process corresponds to solving the sub-problems of the relaxed problem in Lagrangian relaxation.
- The auctioneer collates all the bids and resolves resource conflicts by computing new prices for the resources using a tatonnment price adjustment scheme. This corresponds to an iteration in solving the Lagrangian dual master problem (by using the sub-gradient search method) that updates the price vector λ^{r+1} for the next iteration.
- The auctioneer checks whether a feasible schedule has been found, and if not, it starts the next round of auction by announcing the new prices λ^{r+1} to the bidders.

Hence, in the same way, this analogy carries over nicely to other deterministic scheduling/resource allocation problems so long as it follows the same Lagrangian-based decomposition procedure. In my case, we follow the same idea of dualizing the resource capacity constraints, resulting in independent agent bid generation problems (which will be discussed in detail in the next section), while the price adjustment strategy builds on classical sub-gradient methods with two key ideas - utility pricing and variable step size (which will be discussed in detail in the section after next).

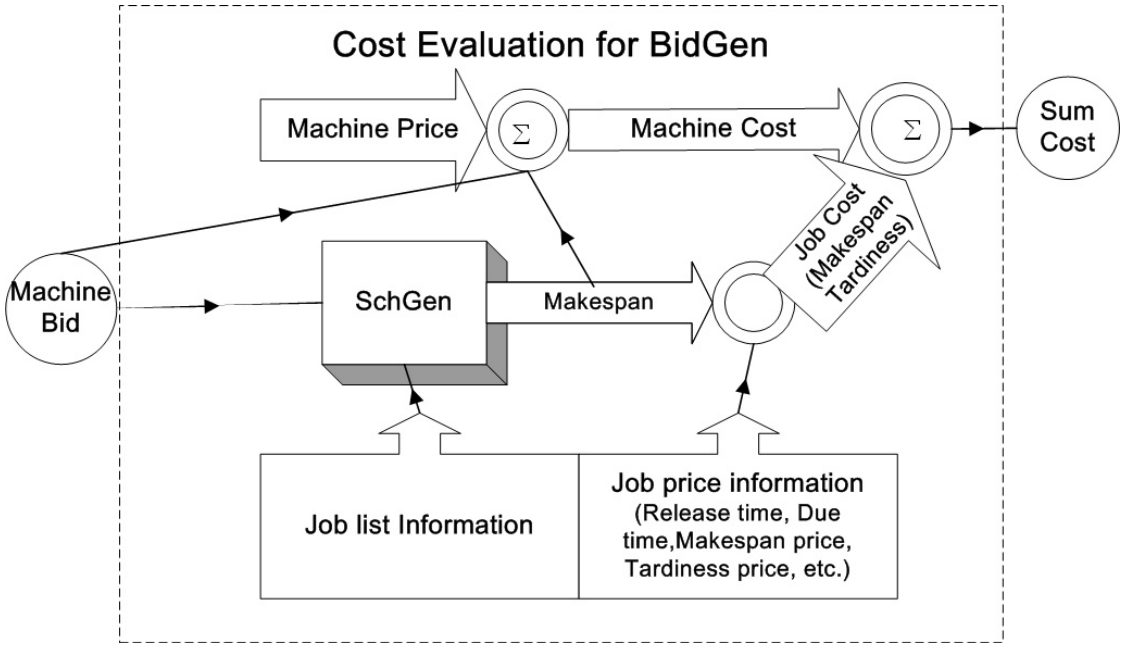


Fig. 4.4: Cost Evaluation for BidGen

4.5 Bid Generation

In this section, we discuss technical details related to bid generation (BidGen). We will first consider a single-period bid generation problem, where we introduce the notion of a makespan matrix. Then the approach is extended to solve the multi-period problem, and from there, we derive the utility prices. The utility prices are then used in the price adjustment process discussed in the next section.

BidGen is essentially a search procedure to find a bid with the minimum total cost, given the prevailing resource pricing information which changes from round to round. It searches the space of possible bids and evaluates the total cost specified in the previous section. The cost evaluation procedure is described pictorially in Figure 4.4. SchGen (Schedule Generation) is a scheduling algorithm that solves the single agent scheduling problem using resources specified in a given bid. For this section, the focus is not to investigate algorithms for SchGen (actually it can be Greedy, CH, RH or IMM in previous

or any best method for a special problem), but rather SchGen is treated as a black box $\mathcal{A}$ that returns the optimal schedule.

One critical concern in bid generation is to be able to find the makespan, which is computationally intensive to find. we also note that the makespan does not depend on the resource prices, and hence once the makespan for a given resource level is computed, it can be stored and looked up for future computation.

For an arbitrary agent l, given the job-list information $\mathcal{L}^l$, bid $\mathbf{U}^l$ and algorithm $\mathcal{A}$, the resulting makespan is expressed as,

$$\begin{aligned} \mathcal{M}(\mathbf{U}^l) &\triangleq SchGen(\mathcal{L}^l, \mathbf{U}^l, \mathcal{A}) \\ \mathbf{U}^l &= (\cdots, U^l_{k,\tau}, \cdots), k = 1, \cdots, K, \tau = 1, \cdots, \mathcal{T} \end{aligned} \tag{4.25}$$

Similarly, over the entire bid space $\mathcal{U}^l$, given that the job-list information is fixed throughout the auction process, the **Makespan Matrix** is defined for an agent l as $\mathcal{M}(\mathcal{U}^l)$.

Strictly speaking, the term *matrix* can be used simple case when $K = 2, \mathcal{T} = 1$. For higher dimensions, it will become a hyper-cube. Under the assumption that the computational time for SchGen is high (which is usually the case for NP-hard scheduling problems), we create the makespan matrix to store and look up the makespan and its corresponding bid during the search process (to be discussed further below). Moreover, we also show how the utility prices can be derived from the values in the makespan matrix.

Hence, the objective function for BidGen is given by Equation (4.26) as follows:

$$\begin{aligned} \mathcal{SC}^l(\mathbf{U}^l) &\triangleq MTC^l + \sum_{\tau}\sum_{k} p_{k\tau} U^l_{k\tau} \qquad (4.26) \\ &= W^l_m \cdot \mathcal{M}(\mathbf{U}^l) + W^l_d \cdot \max\left\{\mathcal{M}(\mathbf{U}^l) + R^l - D^l, 0\right\} \\ &\quad + \sum_{\tau}\sum_{k} p_{k\tau} U^l_{k\tau} \\ &= MTC^l + \mathcal{RC}^l \\ \mathcal{RC}^l &\triangleq \sum_{\tau}\sum_{k} p_{k\tau} U^l_{k\tau} \qquad (4.27) \end{aligned}$$

Above deduction gives the definition of $\mathcal{RC}^l$ in Equation (4.27), which is *resource cost* for agent-l. we have the following notes for $\mathcal{RC}^l$:

- $\mathcal{RC}^l$ is a function of resource price (i.e. bidding price) $p_{k\tau}$, which may change from iteration to iteration;
- $\mathcal{RC}^l$ is a function of bid $\mathbf{U}^l = (\ldots, U^l_{k\tau}, \ldots)$, the resource bid for earlier period may affect the makespan, and so affect the total number of actual periods;
- In BidGen, $p_{k\tau}$ is fixed, each agent-l is trying to find an optimal bid to minimize its sum of cost $\mathcal{SC}^l$.

4.5.1 Single Period BidGen

In the single-period BidGen problem (i.e. $\mathcal{T} = 1$), bids are made on resources at a constant level throughout the time horizon. In the following, we discuss how the makespan matrix is used to efficiently solve the single-period BidGen problem.

Consider the sample problem instance given above, the vector of $\mathbf{U}^l$ simplifies to $[U^l_1, U^l_2]$. For agent 3, suppose we apply algorithm $\mathcal{A}$ to be FCFS (*First Come First Serve*), the makespan matrix is presented in Figure 4.13 in Appendix-4.13, where the x- and y-axis represent the values of U_1 and U_2 respectively.

The proposed approach to solve the single-period BidGen is given pictorially in Figure 4.13 in Appendix-4.13. It is a local search method that begins with an initial bid (say (2,1)) and searches along either dimension until it reaches a makespan value that does not decrease further. It then searches along the other dimension, and the process is repeated until no more improvement is possible (i.e. the sum cost value is at its local minimum). As the search proceeds, undefined matrix values become defined, and are stored (for future search). At the same time, the corresponding sum cost matrix corresponding to the current resource prices are also computed (as shown in Figure 4.13(b)). The details of transition points during search are given in Fig. 4.14 in Appendix-4.13.

In order to show the effectiveness and efficiency of the search approach, the complete total weighted cost matrix is shown in Fig. 4.15 in Appendix-4.13, which is for the same agent 3 in the sample problem. For this particular agent, the complete search boundary is $U_1^l \leq 9, U_2^l \leq 4$. The complete search will evaluate 32 points while the proposed approach only evaluates 22 points and returns the same solutions.

Note that although the search procedure is a local search, Dimitris showed [91] that the local optima is global optima under the assumption that the objective function(total cost $\mathcal{SC}^l$)is convex with respect to the searching space ($\mathbf{U}^l$).

4.5.2 Multiple Period BidGen

The search algorithm to handle the multi-period BidGen is an extension of single-period BidGen search presented above. For simplicity in notation, we omit the subscript l, and hence $U_{k\tau}^l$ is simplified to be $U_{k\tau}$, while keeping in mind this BidGen is done for each agent. The following pseudo code describes the proposed approach:

1. based on the current resource prices p, recompute the sum cost matrix and initialize

$$u^l_{k\tau} \triangleq \frac{\partial MTC^l}{\partial U^l_{k\tau}} = \begin{cases} W^l_m \left|\frac{\partial \mathcal{M}}{\partial U^l_{k\tau}}\right| & if \mathcal{M} < D^l - R^l \\ (W^l_m + W^l_d)\left|\frac{\partial \mathcal{M}}{\partial \mathbf{U}^l_{k\tau}}\right| & if \mathcal{M} \geq D^l - R^l \end{cases} \tag{4.28}$$

$U^*_{k\tau}$ based on the recomputed sum cost matrix as given in equation (4.26)

2. for $(iter = 1..MaxIterations)$ or until no further improvement,

   - for $(\tau = 1..\mathcal{T})$
     - perform single-period search for period τ with bids for other periods τ' kept as $U^*_{k\tau'}$
     - update makespan matrix
   - update $U^*_{k\tau}(1 \leq k \leq K, 1 \leq \tau \leq \mathcal{T})$

4.5.3 Extracting Utility Prices

Finally, we show how the utility prices can be extracted from the entries in the makespan matrix.

equation (4.28) shows the definition of utility prices. It is the makespan price W^l_m (or $W^l_m + W^l_d$ if there is tardiness) times $\frac{\partial \mathcal{M}}{\partial \mathbf{U}^l_{k\tau}}$, which is the marginal (incremental) makespan per unit change on particular machine type k at a fixed period τ. For simplicity of explanation, suppose the unit for the makespan price W^l_m is (virtual) dollars per hour. The unit $\frac{\partial \mathcal{M}}{\partial U^l_{k\tau}}$ is hour per unit machine slot. Hence, the unit for utility price is dollar per unit machine slot. It is called a utility price because it reflects the marginal utility value of a particular machine. It is interpreted as how many dollars could be saved (or lost) if one such machine slot is added (or removed). It is defined on the particular period τ, it is also related with the specific bid point $\mathbf{U}^l_k$. Although each point in the

bid space $\mathbf{U}^l \in \mathcal{U}^l$ has an associated utility price, we only need to calculate this price for the final bid $\mathbf{U}^* = [\cdots, {U^*}_{k\tau}, \cdots]_{1\leq k\leq K, 1\leq\tau\leq\mathcal{T}}$ that has the minimum sum cost. For the implementation, we apply a backward difference procedure:

$$\begin{aligned} & u^l_{k\tau}|_{\mathbf{U}^*} \\ = \; & MTC^l\left((\cdots, {U^*}_{k-1,\tau}, {U^*}_{k\tau} - 1, {U^*}_{k+1,\tau}, \cdots, {U^*}_{K\mathcal{T}})\right) \\ & -MTC^l(\mathbf{U}^*) \end{aligned} \tag{4.29}$$

4.6 Overview of Price Adjustment Strategy

In this section, we present an overview of my contribution in the light of literature.

The classical price adjustment procedure is given as follows:

1. Collect bids from all agents and compute aggregate demand, as equation (4.30).
2. According to the aggregate demand and supply (i.e. capacity) of each resource at each period, compute the price adjustment step size s^r.
3. Update the new price for next iteration for each resource at each period, as equation (4.31), which is dependent on the step size and the demand-supply gap and must be non-negative.

$$D_{k\tau} = \sum_{l=1}^{L} U^l_{k\tau} \tag{4.30}$$

$$p^{r+1}_{k\tau} = \max\{0, p^r_{k\tau} + s^r \cdot (D_{k\tau} - S_{k\tau})\} \tag{4.31}$$

Fisher [93], suggests that the step size s^r is a function of the difference between $\mathcal{UB}$ Upper Bound and $\mathcal{LB}$ Lower Bound estimation of the objective value as numerator, and

the sum of squares of all relaxed constraints as denominator. Kutanoglu [63], follows the same strategy. For the highly intensive computation due to the NP-hardness of the underlying scheduling problem, recent researchers (i.e. [15], [17]) turn to such methods as not to calculate $\mathcal{LB}$ The features of this thesis are listed as follows:

- We do not calculate $\mathcal{LB}$;
- We use the *Average Price* as the numerator and *Root Mean Square* of the gaps between supply and demand as the denominator;
- By continuing from my previous work [17], we make more comprehensive comparison between bidding price and utility price;
- Especially, we propose a technique to compute the step size by combining two ideas from micro-economics and control theory, to help us achieve equilibrium quickly.

The first idea is to incorporate what is termed as the utility price *instead of bid price* in the price adjustment formula. Although PAB is straight-forward and computationally fast within a single iteration, it often exhibits poor convergence, especially when the price values are near to zero. This idea will be further discussed in Section 4.7.

The second idea is that of having variable step size s^r (which changes from one iteration r to the next). A continuous function has been designed for obtaining a feasible solution quickly without much loss of optimality. The function is defined within $(-\infty, \infty)$ and the value is within $(0, 2)$ based on the theoretical result in [105]. The proposed idea, however, is different from the conventional approach in adaptive control, which is based on differential equations, whether Partial Differential Equations *PDE* or Ordinary Differential Equations *ODE* [62, 72]. The energy function in adaptive control is usually symmetrical around some special point. For example, both the Lyapunov function $V_{\bar{q}_c} =$

$\|\bar{q} - \bar{q}_c\|^2$ used by Fua [16], and the trench function $f(d) = \sqrt{1 + d^2} - 1$ used by Sam [37] are symmetric. We would like the readers to note the following differences.

- In application adaptive control, convergence is defined such that the local and/or global control error (setpoint - feedback) are approaching to zero, while in the auction method to solve the optimization problem, convergence is defined s.t. prices are approaching to some unknown values (i.e. optimal price leads to optimal solution);

- In application adaptive control, the error could approach to zero in either directions (positive or negative) symmetrically, while in the problem the price must always be positive. So the problem is unsymmetrically in nature.

- The fundamental difference of above symmetry and asymmetry is that symmetry comes from equality constraints while asymmetry comes from inequality constraints. The PDEs and/or ODEs, on which adaptive control is based, are equations while the optimization problems (i.e. the resource allocation problems) usually have inequality constraints (i.e. capacity constraints).

- Another asymmetry is the way of treating feasibility and optimality. In this design, we prefer fast feasible solution, which is near optimal but may not be exactly optimal in the global sense.

Details of the variable step size technique will be explained in Section 4.8.

4.7 Price Adjustment with Utility Price

In the following, we present two price adjustment strategies - one using the conventional bid prices, while the other using the proposed utility prices. These two approaches will

be compared experimentally in Section 4.11.

4.7.1 Price Adjustment With Bid Price (PAB)

For the strategy of (PAB), the detailed formulation is as follows:

1. Calculate the aggregate demand as equation(4.30).
2. Step size $s^r = \alpha \cdot \frac{\frac{\sum_k \sum_t p^r_{k\tau} D_{k\tau}}{\sum_k \sum_t D_{k\tau}}}{\sqrt{\frac{\sum_{k=1}^{K} \sum_{\tau=1}^{\mathcal{T}^r} (D_{k\tau} - S_{k\tau})^2}{K \cdot \mathcal{T}^r}}}$
3. Calculate new price for next iteration as Eqn.(4.31)

4.7.2 Price Adjustment With Utility Price (PAU)

For this strategy, each agent must submit a utility price together with the bids $u^l_{k\tau}$. The auctioneer will calculate the average utility price $u_{k\tau}$ for each resource on each period and this average is weighted by individual agent's demand:

1. Calculate the aggregate demand as (4.30).
2. Calculate average utility price, $u_{k\tau} = \frac{\sum_l^L u^l_{k\tau} U^l_{k\tau}}{\sum_l^L U^l_{k\tau}}$, for $\forall 1 \leq k \leq K, 1 \leq \tau \leq \mathcal{T}$.
3. Step size $s^r = \alpha \cdot \frac{\frac{\sum_k \sum_t u_{k\tau} D_{k\tau}}{\sum_k \sum_t D_{k\tau}}}{\sqrt{\frac{\sum_{k=1}^{K} \sum_{\tau=1}^{\mathcal{T}^r} (D_{k\tau} - S_{k\tau})^2}{K \cdot \mathcal{T}^r}}}$.
4. Calculate new price for next iteration as Eqn.(4.31)

4.7.3 Comparing PAB and PAU

The major differences between PAB and PAU are the following:

1. The convergence of PAB depends on the bid price, while PAU does not. During the auction process, once there is a round r^* such that $p^{r^*}_{k\tau} = 0, \forall k\tau$, then $p^r_{k\tau} \equiv 0, \forall r \geq r^*, \forall k\tau$, which means the price thereafter will always be 0, whether or not a feasible solution has been reached.

$$
\begin{aligned}
\alpha^r &= \frac{AVE_{kt}(UtilityPrice)}{RMS_{kt}(Demand - Supply)} \\
&= \frac{\frac{\sum_k \sum_t u_{kt} D_{kt}}{\sum_k \sum_t D_{kt}}}{\sqrt{\frac{\sum_k^K \sum_t^{T^r} (D_{kt} - S_{kt})^2}{K \cdot T^r}}} \qquad (4.32) \\
\beta_k^r &= \beta_{std} \cdot \beta_{speed} \\
&= \max\left\{1, sign(\mathcal{N^M}_k) \cdot std_{t=1}^{T^r - 1}(D_{kt})\right\} \cdot \begin{cases} exp(-\mathcal{N^M}_k{}^2) & if \mathcal{N^M}_k < 0 \\ 2 - exp(-\mathcal{N^M}_k{}^2) & if \mathcal{N^M}_k \geq 0 \end{cases} \qquad (4.33) \\
p_{kt}^{r+1} &= \max\{0, p_{kt}^r + \alpha^r \beta_k^r \cdot (D_{kt} - S_{kt})\} \qquad (4.34)
\end{aligned}
$$

2. The rate of convergence of PAB depends on the initial price while PAU is insensitive to it. The closer the initial bid price is to zero, the slower the convergence for PAB. These differences are precisely needed to speed up convergence.

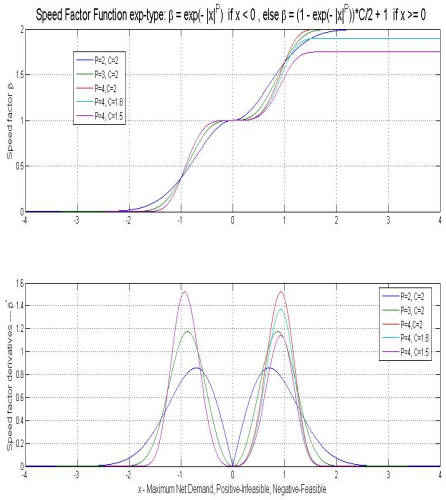
4.8 Price Adjustment with Variable Step Size (VSS)

The condition of convexity presented in the previous section may not always be true in practice. Relaxing these assumptions may, sometimes, result very small values for utility prices, which will in turn impact the speed of convergence. In this section, we address this shortcoming by the technique of computing variable step size.

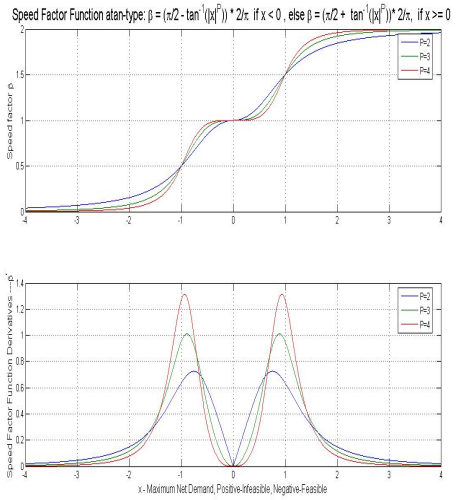
The intuitive need for variable step size is listed as follows:

- When the maximum net demand $\mathcal{N^M}_k = \max_t\{D_{kt} - S_{kt}\}$ is > 0, which means the resource capacity constraint is violated, a larger factor is chosen for speeding up the process to get a feasible solution.

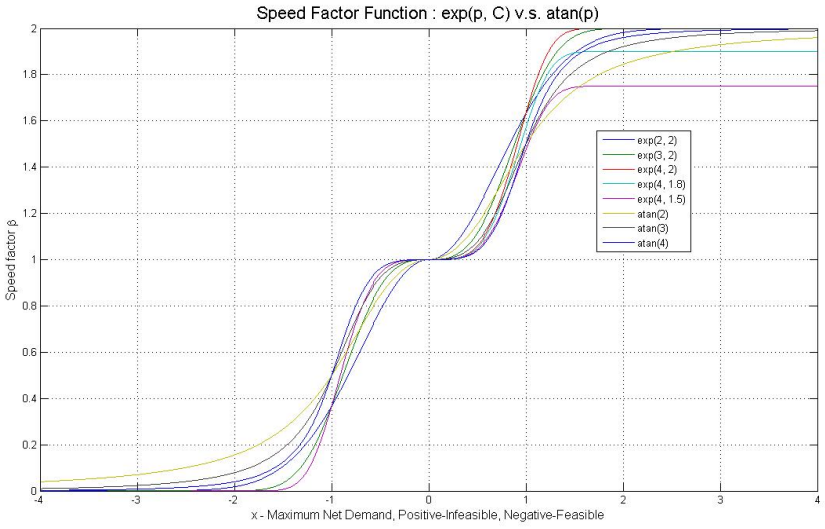
- When the maximum net demand $\mathcal{N^M}_k = \max_t\{D_{kt} - S_{kt}\}$ is $<= 0$, which means the resource capacity constraint is already satisfied (so there are already feasible solutions), a smaller factor is chosen to fine tune the optimality.



(a) Speed Factor exp-type, $f_1(x)$ and $f_1'(x)$



(b) Speed Factor atan-type, $f_2(x)$ and $f_2'(x)$



(c) Speed Factor in one plot

Fig. 4.5: Comparison of Speed Factor

- The standard deviation is considered. In an optimal solution, we find the $std_{t=1}^{T^r-1}(D_{kt})$ should be as small as possible.

In this version of variable step-size price adjustment, the step-size s^r given in equation (4.31) will be replaced by a composite $\alpha^r \cdot \beta_k^r$, where α^r intuitively is the utility price component. As for β_k^r, we further break into $\beta_{std}^r(k) \cdot \beta_{speed}^r(k)$ representing the factor of standard deviation $\beta_{std}^r(k)$ and the speed factor $\beta_{speed}^r(k)$, which are dependent on specific machine type k. $\beta_{std}^r(k)$ is built from the standard deviation of machine utilization for each type, and $\beta_{speed}^r(k)$ is a function of maximum net demand. They are defined as follows.

$$x_k = \mathcal{N}^M{}_k = \max_t\{D_{kt} - S_{kt}\} \tag{4.35}$$

$$\beta_{std} = \max\left\{1, sign(x_k) \cdot std_{t=1}^{T^r-1}(D_{kt})\right\} \tag{4.36}$$

$$\beta_{speed} = f_{speed}(x_k) = \begin{cases} exp(-{x_k}^2) & \text{if } x_k < 0 \\ 2 - exp(-{x_k}^2) & \text{if } x_k \geq 0 \end{cases} \tag{4.37}$$

Details of the considerations are given as follows:

- The standard deviation is from the first period to the second last period $T^r - 1$, as we find the agent's job-list may not fully utilize the resource in the last period. This is a property of multi-period scheduling and resource allocation problems. Resource utilization in the last period depends on each specific scheduling problem, so it usually cannot be as fully utilized as the previous periods.

- Factor of β_{std} is only considered when the solution is NOT feasible, i.e. $sign(\mathcal{N}^M{}_k) > 0$, for some specific machine type. When it is a feasible solution, $\beta_{std}^r(k) = 1$ since $sign(\mathcal{N}^M{}_k) < 0$ and there is always

$$std_{t=1}^{T^r-1}(D_{kt}) > 0$$

- The speed factor is a function of the maximum net demand. Above implementation in equation (4.37) is just one option. More options are further studied later.
- The speed factor function is continuous up to at least 2^{nd} order. They are all equal to 1 when $x_k = 0$, which means a feasible solution with best resource utilization.
- Every optional speed factor $\beta_{speed} = f_{speed}(x_k)$ is less than 1 for a feasible solution and greater than 1 for an infeasible solution.

The design of the speed function $f_{speed}(x_k) \Rightarrow f(x)$ bears the following considerations.

FuncSpeed-1: Defined for all real number, better with some order of continuity, s.t. $f(x) \in C^p[-\infty, \infty]$ is *p-th* order continuous;

FuncSpeed-2: $f(0) = 1$, as explained in above, it can be omitted when perfect utilization is achieved;

FuncSpeed-3: It approaches 0^+ when x approaches $-\infty$, for convergence with feasible solutions, $f(-\infty) = 0^+$;

FuncSpeed-4: It approaches monotonically 2^- when x approaches ∞, i.e. $f(\infty) < 2^-$.

The above *FuncSpeed-1* and *FuncSpeed-2* are easy to understand, while *FuncSpeed-3* and FuncSpeed-4 are actually adopted from [93]. Furthermore, in [93], it has been shown that the following two properties for the step-size s^r are sufficient but NOT necessary conditions for achieving convergence:

- $s^r \to 0^+$, i.e. the price adjustment procedure is converging;
- $\sum_{i=0}^{r} s^i \to \infty$, i.e. the convergence does not occur too quickly.

While there are many functions that can satisfy the above specifications, we focus the study on two types of of candidate speed functions. They are $e^{-|x|^p}$ type or $tan^{-1}(x)$ type, which will be further studied in next sub section.

4.8.1 Study on two types of speed function candidates

The first types of speed function candidates $f_1(x)$ is exponential and has two parameters, i.e. index parameter $p \geq 2, p \in \mathcal{N}$ and offset parameter $C \in (0, 2]$.

$$f_1(x) = \begin{cases} exp(-|x|^p) & , x < 0 \\ 1 + \frac{C}{2}(1 - exp(-|x|^p)) & ,x \geq 0 \end{cases} \tag{4.38}$$

Above function is $p-th$ order continuous since it can be verified that $f_1'(0^-) = f_1'(0^+) = 0$.

$$f_1'(x) = \begin{cases} p \cdot (-x)^{p-1} \cdot exp(-|x|^p) & , x < 0 \\ \frac{p \cdot C}{2} x^{p-1} \cdot exp(-x^p) & , x \geq 0 \end{cases} \tag{4.39}$$

Besides the continuous property, there are $f_1(-\infty) = 0^+$ and $f_1(\infty) = \frac{C}{2} + 1$. Since $C \in (0, 2]$, there is $\frac{C}{2} + 1 \in (1, 2]$. Intuitively, the larger C is, the larger step-size will be introduced whenever an infeasible solution happens. Comparisons of some $f_1(x)$ with different parameters (p, C) are shown in Fig. 4.5-(a).

The second types of speed function candidates $f_2(x)$ is $tan^{-1}x$ type. It has only one index parameter p, s.t. $p \geq 2, p \in \mathcal{N}$.

$$f_2(x) = \begin{cases} 1 - \frac{2}{\pi} \cdot tan^{-1}(|x|^p) & , x < 0 \\ 1 + \frac{x}{\pi} \cdot tan^{-1}(|x|^p) & , x \geq 0 \end{cases} \tag{4.40}$$

It can be verified that $f_2(x)$ is also $p-th$ order continuous.

$$f_2'(x) = \begin{cases} \frac{2p}{\pi} \frac{(-x)^{p-1}}{1+x^{2p}} & , x < 0 \\ \frac{2p}{\pi} \frac{x^{p-1}}{1+x^{2p}} & , x \geq 0 \end{cases} \tag{4.41}$$

Further, $f_2(-\infty) = 0^+$ and $f_2(\infty) = 2^-$. Comparisons of some $f_2(x)$ with different parameters p are shown in Fig. 4.5-(b).

The overall comparisons among exp-type $f_1(x)$ and $tan^{-1}x$ type $f_2(x)$ are shown in Fig. 4.5-(c). we will further present their difference empirically in experiment parts.

4.8.2 Integrated price adjustment

Combining both utility price and variable step size concepts, the following price adjustment formulas are listed in equation (4.32) - equation (4.34).

The step-size s^r in Eqn.(4.31) is now replaced by two components, α^r and β_k^r.

- equation (4.32) shows that α^r is the average of utility price over RMS *Root Mean Square* net demand;
- equation (4.33) shows that β_k^r which augments the standard deviation factor by considering variable step size;
- equation (4.34) updates the price vector for the next round.

Again, the reader will note that the formulation does not calculate the lower bound $\mathcal{LB}$ and upper bound $\mathcal{UB}$, as in [63, 93], since they are computationally expensive.

4.9 Further insight for utility price and BidGen

4.9.1 Utility price and local optimality

Now we focus on discrete resource scheduling problems, and show some further results.

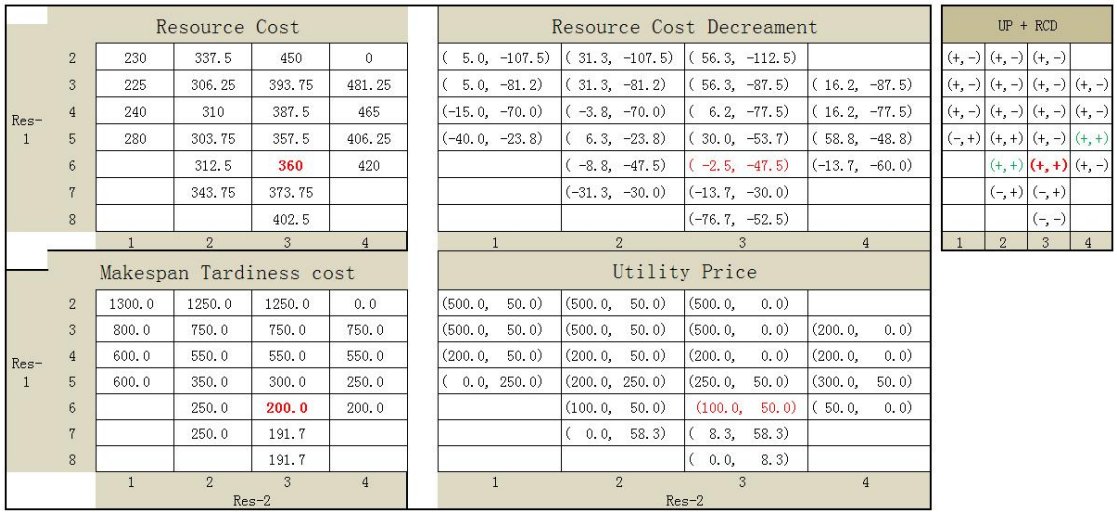


Fig. 4.6: Demonstration of local minimum and utility price

Following previous definition of resource cost in Equation (4.27), we define the decrement of resource cost

$$\begin{aligned} \rho_{k\tau} &\triangleq \frac{\partial \mathcal{RC}}{\partial U_{k\tau}} \\ \mathcal{RCD}(U)_{k\tau} &\triangleq \mathcal{RC}(\ldots, U_{k\tau} - 1, \ldots) - \mathcal{RC}(\ldots, U_{k\tau}, \ldots) \end{aligned} \tag{4.42}$$

We have the following notes for above definition,

- Resource cost $\mathcal{RC}$ is a scalar function of a vector in $K\mathcal{T}$;
- In continuous time, $\rho = (\rho_{11}, \ldots, \rho_{k\tau}, \ldots, \rho_{K\mathcal{T}})^T$, a vector of $K\mathcal{T}$, is the gradient of $\mathcal{RC}$;
- Resource cost decrement is a vector in discrete resource scheduling problems; for implementation, it must be consistent with utility price, i.e. backward difference as that in Equation (4.29);
- Physical meaning of $\mathcal{RCD}(U)_{k\tau}$ is how much cost can be saved if there is a unit decrement of resource bid $U_{k\tau}$;

- Resource cost $\mathcal{RC}$, defined in Equation (4.27), is found to be a non-linear function of resource bid, as bid variation may change the makespan and then $\mathcal{T}$ is changed;
- $\mathcal{RCD}(U)_{k\tau}$ is also a function of resource bid;
- When $U_{k\tau}$ is at the search boundary (i.e. $U_{k\tau} - 1$ not exist in the search domain), for the purpose of 1^{st} order continuity, we have

$$\mathcal{RCD}(U)_{k\tau} \triangleq \mathcal{RC}(\ldots, U_{k\tau}, \ldots) - \mathcal{RC}(\ldots, U_{k\tau} + 1, \ldots)$$

For better understanding of above notation, we list the *utility price* and *resource cost decrement, RCD* (i.e. $\mathcal{RCD}$) in Fig. 4.6. It is a single-period, 2-resource instance, exactly the same instance and same bidder as Fig. 4.16 and Fig. 4.13. For this BidGen, bidding price for resources, $p_{Res-1} = 15, p_{Res-2} = 30$.

Then we have the condition of local optimality of bid-generation.

Proposition 4.9.1 *For a discrete resource allocation scheduling problem,* $\mathbf{U}^*$ *is a local optima w.r.t. sum cost* $\mathcal{SC}$ *in equation (4.26) for any agent, if and only if, the following conditions apply*

- $u(\mathbf{U}^*)_{k\tau} + \mathcal{RCD}(\mathbf{U}^*)_{k\tau} \geq 0, \forall k, \tau$;
- $u(\ldots, U^*_{k\tau} + 1 \ldots)_{k\tau} + \mathcal{RCD}(\ldots, U^*_{k\tau} + 1 \ldots)_{k\tau} \leq 0 \forall k, \tau$;

In above there is

$$\mathbf{U}^* = (U^*_{11}, U^*_{21}, \ldots, U^*_{K1}, \ldots, U^*_{k\tau}, \ldots, U^*_{1\tau}, \ldots, U^*_{K\mathcal{T}}) \in \mathcal{R}^{K \times \mathcal{T}}$$

Proof: Following the definition in equation (4.26), $\mathcal{SC} = MTC + \mathcal{RC}$, that $\mathbf{U}^*$ is a local optima

$$\Updownarrow$$

$$MTC(\mathbf{U}^*) + \mathcal{RC}(\mathbf{U}^*) \leq$$

$$MTC(\ldots, U^*_{k\tau} - 1, \ldots) + \mathcal{RC}(\ldots, U^*_{k\tau} - 1, \ldots) \tag{4.43}$$

$$MTC(\mathbf{U}^*) + \mathcal{RC}(\mathbf{U}^*) \leq$$

$$MTC(\ldots, U^*_{k\tau} + 1, \ldots) + \mathcal{RC}(\ldots, U^*_{k\tau} + 1, \ldots) \tag{4.44}$$

both $\forall k, \tau$

By moving the left-hand-side of (4.43) to right-hand-side and rearranging, we have

$$\begin{aligned} & u(\mathbf{U}^*)_{k\tau} + \mathcal{RCD}(\mathbf{U}^*)_{k\tau} \\ = \;& MTC(\ldots, U^*_{k\tau} - 1, \ldots)_{k\tau} - MTC(\mathbf{U}^*)_{k\tau} + \\ & \mathcal{RC}(\ldots, U^*_{k\tau} - 1, \ldots)_{k\tau} - \mathcal{RC}(\mathbf{U}^*)_{k\tau} \\ \geq \;& 0 \end{aligned} \tag{4.45}$$

And by moving the right-hand-side of (4.44) to left-hand-side and rearranging, we have

$$\begin{aligned} & MTC(\mathbf{U}^*)_{k\tau} + \mathcal{RC}(\mathbf{U}^*)_{k\tau} - \\ & (MTC(\ldots, U^*_{k\tau} + 1, \ldots) + \mathcal{RC}(\ldots, U^*_{k\tau} + 1, \ldots)) \\ = \;& MTC(\mathbf{U}^*)_{k\tau} - MTC(\ldots, U^*_{k\tau} + 1, \ldots) + \\ & \mathcal{RC}(\mathbf{U}^*)_{k\tau} - \mathcal{RC}(\ldots, U^*_{k\tau} + 1, \ldots) \\ = \;& u(\ldots, U^*_{k\tau} + 1 \ldots)_{k\tau} + \mathcal{RCD}(\ldots, U^*_{k\tau} + 1 \ldots)_{k\tau} \\ \leq \;& 0 \qquad \forall k, \tau \end{aligned} \tag{4.46}$$

Q.E.D.

For better understanding, we list the sign of $u_{k\tau} + \mathcal{RCD}_{k\tau}$ (i.e. $RCD + UP$), beside in Fig. 4.6. Here, $\mathcal{T} = 1, K = 2$ stands for a single period, 2 resource-type problem. Both utility price (i.e. UP) and resource cost decrement (RCD) are vectors. First element of this vector stands for Res-1 while 2nd element stands for Res-2. Bid $(U_1, U_2) = (6, 3)$ is an optima. It can be verified that

$$\begin{aligned} u(6,3)_1 + \mathcal{RCD}(6,3)_1 &= -2.5 + 100 = 97.5 > 0 \\ u(6,3)_2 + \mathcal{RCD}(6,3)_2 &= -47.5 + 50 = 2.5 > 0 \\ u(6+1,3)_1 + \mathcal{RCD}(6+1,3)_1 &= -13.7 + 8.3 = -5.4 < 0 \\ u(6,3+1)_2 + \mathcal{RCD}(6,3+1)_2 &= -60 + 0 = -60 < 0 \end{aligned}$$

4.9.2 Makespan convexity and non-zero utility price

To provide some insights on the desirable property that PAU is not dependent on the bid price, we show the following lemma and proposition:

Lemma 4.9.2 *If the makespan matrix $\mathcal{M}(\mathcal{U}^l)$ satisfies the following conditions:*

Monotonicity: *When τ is fixed, it is monotonic non-increasing in any machine $U^l_{k\tau}, \forall k$;*

Convexity: *It is convex with respect to machine utilization $U^l_{k\tau}, \forall k$;*

BackDiff: *The calculation of utility price is based on backward difference, as in (4.29).*

Then, once $\exists \tau, k$ such that $u_{k\tau}|_{U^*_k} = 0$, $u_{k\tau}|_{U_k} \equiv 0, \forall U_k \geq U^*_k$ on that period τ.

Proof: For a particular agent l, machine bid $\mathbf{U}^*_{k,\tau}$ is represented by a vector of

$$\left[U^*_{1,\tau_1}, U^*_{2,\tau_1}, ..., U^*_{k\tau}, ..., U^*_{K,\tau_\mathcal{T}}\right]$$

and $u_{k\tau}|_{U_k^*} = 0$ means that

$$\mathcal{M}(..., U_{k\tau}^*, ...) = \mathcal{M}(..., U_{k\tau}^* - 1, ...)$$

Due to convexity,

$$\mathcal{M}(..., U_{k\tau}^* + 1, ...) + \mathcal{M}(..., U_{k\tau}^* - 1, ...) \geq 2\mathcal{M}(..., U_{k\tau}^*, ...)$$

then

$$\mathcal{M}(..., U_{k\tau}^* + 1, ...) \geq \mathcal{M}(..., U_{k\tau}^*, ...)$$

Because of monotonic non-increasing property,

$$\mathcal{M}(..., U_{k\tau}^* + 1, ...) \leq \mathcal{M}(..., U_{k\tau}^*, ...)$$

Then there will be

$$\mathcal{M}(..., U_{k\tau}^* + 1, ...) = \mathcal{M}(..., U_{k\tau}^*, ...)$$

It is further extended as

$$\mathcal{M}(..., U_{k\tau}, ...) \equiv \mathcal{M}(..., U_{k\tau}^*, ...), \forall U_{k\tau} \geq U_{k\tau}^*$$

So, $MTC^l(U_k, \tau)$ will not change for $U_{k\tau} \geq U_{k\tau}^*$; Then $u_{k\tau}|_{U_k} \equiv 0, \forall U_k \geq U_k^*$ on that period τ. **Q.E.D.**

In particular, for competitive situations where the total number of operations to be performed on a particular machine is greater than the baseline allocation. i.e. for agent-l, $\sum_{m_{ij}=k}^{1\leq i\leq N^l, 1\leq j\leq o_i^l} 1 > B_k^l$, a non-zero utility price can be guaranteed. we have the following corollary.

Proposition 4.9.3 *In a* competitive *environment, if the makespan matrix* $\mathcal{M}(\mathcal{U}^l)$ *satisfies the above 3 conditions* { *Monotonicity, Convexity, BackDiff* }*,* it never happens that $u_{k\tau}|_{U_k^*} = 0, \forall \tau, k$ except the region near the upper bound of machine capacity.

Proof: Following the proposition, in the competitive environment, simply there is the following strict inequality.

$$\mathcal{M}(..., B^l_{k\tau} + 1, ...) < \mathcal{M}(..., B^l_{k\tau}, ...)$$

Then $u^l_{k\tau}|_{B^l_k} > 0$. **Q.E.D.**

Intuitively, the optimal bid submitted by the bidder is depending on the bid price released from the auctioneer. While the utility price is depending on the local property (i.e. partial derivatives) of the optimal bid, it is also related with the scheduling problem of each agent. For this relationship, we declare that method of using utility price is adaptive to the decentralized agent's scheduling problem.

Theoretically for scheduling problems, the condition of monotonicity should be always true, since adding resource should never decrease the schedule. But convexity is NOT so.

The convexity condition for makespan matrix is somewhat stringent. For example, consider the agent 3 in the sample problem, the complete makespan matrix is shown in Fig.4.16 in Appendix-4.13. In 4.16-(A), the monotonicity is studied clearly. We can see it is surely monotonic decreasing for any direction, either Res-1 or Res-2, from any point. However, it is NOT convex. Take points $2.92at[3,3], 2.58at[4,3], 2.17at[5,3]$ for example, $2.92 + 2.17 = 5.09 < 2.58 \times 2 = 5.16$.

For such non-convexity points, we explain the cause is that the schedule is related with more than 2 resources and dispatching resources are inter-dependant. Above example also shows **that *convexity* is a sufficient but NOT necessary condition for non-zero utility price**.

Before Reallocation							
Agent-Id	(Res-1, Res-2) per Hour					Makespan (hour)	Makespan Tardiness Cost
	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00		
1	(2, 1)	(9, 3)	(6, 4)	(3, 1)		3.95	395
2	(3, 1)	(6, 2)	(3, 2)	(5, 2)	(5, 1)	5	500
3	(2, 1)	(4, 2)	(5, 2)	(8, 3)	(7, 1)	5	500
4	(1, 0)	(5, 2)	(6, 4)	(6, 3)	(6, 1)	4.8	480
	Free Resources per Hour						
	(16, 13)	(0, 7)	(4, 4)	(2, 7)	(6, 13)		

After Reallocation							
Agent-Id	(Res-1, Res-2) per Hour					Makespan (hour)	Makespan Tardiness Cost
	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00		
1	(2, 1)	(9, 3)	(6, 4)	(3, 1)		3.95	395
2	(14, 3)	(6, 5)	(2, 1)			2.6	260
3	(6, 2)	(4, 4)	(10, 5)	(8, 1)		3.75	375
4	(2, 0)	(5, 2)	(6, 4)	(8, 2)	(2, 1)	4.3	430
	Free Resources per Hour						
	(0, 10)	(0, 2)	(0, 2)	(5, 12)	(22, 15)		

(a) Before and after resource re-allocation

Fig. 4.7: Demonstration of effect of resource re-allocation

For this problem, it turns out that optimal point located in a convex region, specifically, $4 \leq U_1^l \leq 9, 1 \leq U_2^l \leq 4$. This observation leads us to restrict the searching domain, s.t. makespan should drop whenever the amount of resource is increased. This is the reason why the BidGen in 4.5 could work fine, i.e. faster and without loss of optimality.

4.10 Resource Re-allocation Strategy

Although the adaptive auction scheme can improve speed of convergence and achieving feasibility, it often results as an local optimal solution. On careful examination, it is possible to achieve better solutions through reallocation of resources that has not been fully utilized. We perform a post-processing phase with the following resource re-allocation strategy. We apply the simple priority rules proposed in [92] and the priority is defined

as its makespan-tardiness cost as follows in (4.47).

$$\begin{aligned} \rho^l &= \mathcal{MTC}^l \\ &= W_m^l \cdot \mathcal{M}(\mathbf{U}^l) + W_d^l \cdot \max\left\{\mathcal{M}(\mathbf{U}^l) + R^l - D^l, 0\right\} \end{aligned} \tag{4.47}$$

We consider unallocated resources in each period, along the time-line. For each case of un-fully utilization, the free resources are simply allocated to the single agent, s.t.

- its ρ^l is the largest in value;
- it has started its job-list on above period;

For the complexity of schedule generation and resource's inter-dependance, the agent may not fully utilize the reallocated resource. So after each time of reallocation, the agent will regenerate its schedule and re-calculate the ρ^l. Above procedure is repeated until any of the agents' makespan doesnot decrease further.

For the mercy of understanding, we take an example from later Section 4.11.2. In this example, the effect of this re-allocation is shown in Fig. 4.7. A basic point is that resource re-allocation must be carried out after there is already a feasible solution. The adaptive auction (i.e. BidGen, PAU, Variable Step-Size) is an important process to achieve a feasible solution, and this process doesnot depend on initial price.

4.11 Experimental Results

We first present detailed results on the sample problem instance given in 4.1, followed by a number of problem instances. We benchmark against a MIP (mixed integer programming) model **ResAlloc-MIP** whose objective function is the overall sum of weighted makespan cost and tardiness penalty of all agents (Details on the ResAlloc-MIP formulation is presented in the Appendix):

Table 4.4: Multi-period Resource Allocation, Solution Comparison MIP, PAB and PAU

Agent -Id	Algo Option	Quota (Machine 1, Machine 2)			
		1^{st} Period	2^{nd} Period	3^{rd} Period	4^{th} Period
		8:00 - 9:00	9:00 - 10:00	10:00 -11:00	11:00 - 12:00
1	MIP	(6, 2)	(3, 2)	(0, 0)	(0, 0)
	PAB	(5, 2)	(4, 2)	(1, 1)	(0, 0)
	PAU	(7, 2)	(4, 2)	(0, 0)	(0, 0)
2	MIP	(7, 2)	(4, 2)	(0, 0)	(0, 0)
	PAB	(8, 3)	(3, 3)	(0, 0)	(0, 0)
	PAU	(7, 3)	(3, 2)	(1, 1)	(0, 0)
3	MIP	(0, 0)	(3, 2)	(7, 4)	(0, 0)
	PAB	(0, 0)	(4, 2)	(4, 1)	(1, 2)
	PAU	(0, 0)	(5, 2)	(5, 4)	(0, 0)
4	MIP	(0, 0)	(5, 1)	(5, 2)	(0, 0)
	PAB	(0, 0)	(3, 1)	(2, 2)	(0, 0)
	PAU	(0, 0)	(4, 1)	(5, 3)	(0, 0)
Sub -total	MIP	(13, 4)	(15, 7)	(12, 6)	(0, 0)
	PAB	(13, 5)	(16, 8)	(11, 6)	(1, 2)
	PAU	(14, 5)	(16, 7)	(11, 8)	(0, 0)

Table 4.5: Solution Comparison for Sample Problem

Method	Solution Makespan (hour)					Total Cost	Run Time (sec)
	$l = 1$	$l = 2$	$l = 3$	$l = 4$	Ave		
MIP	1.8	1.7	2.0	1.7	1.8	1133.3	7200
PAB	2.1	2.0	2.3	1.6	2.0	1445.8	53.8
PAU	2.0	2.1	1.9	2.0	2.0	1358.3	32.7

Table 4.6: Comparison for Different Initial Prices

Method	Level of Initial Price	NO. Iteration for 1^{st} Feasible Solution	Run Time(sec)
PAB	High	1	53.8
	Low	30	67
	Zero	Fail	Fail
PAU	High	1	32.7
	Low	2	35.7
	Zero	3	35.7

$$\textbf{ResAlloc-MIP: minimize } \sum_{l=1}^{L} MTC^l \tag{4.48}$$

4.11.1 Comparison on Sample Problem Instance

First, we compare both the solution quality and run time of the MIP solutions against solutions with both PAB and PAU strategies. Table (4.4) gives the allocation for each agent for MIP, PAB and PAU solutions respectively. Table 4.5 shows the comparison of the total cost and average makespan for 4 agents. The solution by PAU is closer to optimal and the run time is comparable with that of PAB.

The feasibility of allocation solution and scheduling are also verified and compared between the PAU-VarStepSize and by CPLEX, MIP formulation details are available in Appendix).

Another observation is that the run-time performance of PAB is dependent on the initial price settings as shown in 4.6. In all above solutions, the initial prices for machine types 1 and 2 are respectively $[50, 50, 80, 150]$ and $[80, 80, 130, 200]$ corresponding to each time period. Given this sufficiently high price levels, we observe that the rate of convergence of PAB is comparable with that of PAU. In another experiment, we investigate the setting for low initial prices. The initial price for machine 1 is now set to $[5, 5, 8, 15]$ and for machine 2, $[8, 8, 13, 20]$. We record the time of obtaining the first feasible solution. When initial price is low, each agent will bid a higher amount such that the overall demand exceeds supply and hence there is an infeasible solution. Under the PAU strategy, a feasible solution can be available within 3 iterations while it takes about 30 iterations for the PAB strategy. When the initial price is zero, PAB fails to find a feasible solution while PAU stills achieves a feasible solution within a short time. This result is consistent

with the proposition.

4.11.2 Comparison with MIP model

Then we compare the solution quality of the proposed approach with ResAlloc-MIP, a mixed integer programming model solved by CPLEX. We generated 3 groups, each with 10 problem instances to compare on both the solution quality and run time performance. We would like to remark that for the MIP model, CPLEX cannot even find a feasible solution within 6 hours for large-scale problem instances. In the experimental setup, we consider medium-sized hard problem instances with 4 agents, each agent has 20 jobs on 3 types of machine.

The common settings for all 3 groups (30 problem instances) are following:

- A discrete time domain where one period is equal to 40 time units;
- The tardiness penalty and makespan price are equal for all agents - 500 and 100 per period respectively;
- There are 3 types of machines in total. For the machine capacity, there are 4 machines of type 1, 16 machines of type 2, and 24 machines of type 3; and
- We take machine type-1 as agent, and so there are 4 agents in total.

We make difference for 3 groups in the job setting and processing time variation. The purpose is to show that the proposed auction architecture is capable to handle flow-shop, bidirectional flow-shop with / without COS constraints.

For the 1st 10 problem instances, each has a 20 standard flow-shop scheduling problem (i.e. a job-list with 20 jobs on 3 machines), and the following settings:

- Two agents start their job-lists at 8:00 and the other 2 start at 9:00 (or 20 time-slot later), and each job comprises 3 operations;
- Processing time on type 1 machine is 1 unit, and type 3 machine is 2 units for all jobs.
- Processing time on type 2 machine is uniformly distributed over a range. For the experiments, we set the bounds to be [10, 16] for agent 1, [13, 17] for agent 2, [14, 18] for agent 3 and [11, 21] for agent 4.

For the 2nd and 3rd group of 10 problem instances, the flow-shops have bi-directional job-features and there are COS(Critical Operational Sequencing) constraints. For the 2nd group, each agent has 10 forward-flow jobs followed by 10 reverse-flow jobs. **For the 3rd group of 10 problem instances**, each agent has a 10 forward-flow jobs mixed with 10 reverse-flow jobs. Different from the 2nd group, these 20 jobs are mixed s.t. each forward job is immediately followed by a reverse job.

- All the 4 agents start their job-lists at the same time;
- The processing time on type 1 machine is a variable uniformly chosen from {1, 2} unit, the processing time on type 3 machine is a variable uniformly chosen from {1,2,3,4,5} units;
- Processing time on type 2 machine is uniformly distributed over a range. For the experiments, we set the bounds to be [10, 16] for agent 1, [13, 17] for agent 2, [14, 18] for agent 3 and [11, 21] for agent 4; and
- There is COS constraints, the critical operation is performed on machine type-1, which also represents the job-agent.

The above problem instances are extracted from a real-world resource allocation problem in distributed container terminal management. The details of the model explanation are available in the previous work [4]. For clearly understanding the process-job settings, the machine dispatching and jog-schedule for both the MIP-CPLEX solution and Auction-PAU-VSS are shown in appendix. The Figures 4.17 and 4.18 illustrate a sample from 1st group. The Figures 4.19 and 4.20 illustrate a typical setting in the 2nd group, while Figures 4.21 and 4.22 illustrate the 3rd group.

In terms of computation time, we let CPLEX run for 2 hours for each problem and record the best solution with the duality gap. On the same computer with a Pentium-4 CPU at 3GHz with 1GB memory, we run the PAU auction for the same 10 problem instances. The solution for the 1st group is shown in and the comparison is listed in Figure 4.8. The 2nd group's result is in Figure 4.9 and the 3rd group's result is in Figure 4.10. We would like the readers to notice the following points:

- The 1st group of problem instances are exactly the same as in [9]. The current solution is done by auction with both PAU and **Variable Step-Size** (*VSS*), while the preliminary result in [9] is done by solution with PAU and fixed step size.

- The 2nd and 3rd group of problem instances are actually from the previous research on bidirectional flow-shops with COS constraints [4].

- The previous version of CPLEX is 10.0 while this time the version of CPLEX is 10.1.1.

- By comparison of the total cost (i.e. the smaller objective function the better), the solution is slightly better than the ResAlloc-MIP solution by CPLEX in 2 hours, but worse than that in 12 hours.

Solution Comparison 2 agents start 8:00 and the other 2 start at 9:00							
	SumCost (Makespan + Tardiness)			PAU over CPLEX		CPLEX Dual Gap	Solution Time by DAT with PAU & VSS (sec)
	PAU & VSS	CPLEX in 2 hours	CPLEX in 12 hours	CPLEX in 2 hours	CPLEX in 12 hours	12 hours	
Case-1	1055	1085	885	97%	119%	49.74%	342
Case-2	1065	1120	915	95%	116%	55.32%	87
Case-3	1165	1130	930	103%	125%	55.40%	416
Case-4	895	1025	825	87%	108%	49.59%	204
Case-5	1040	1080	880	96%	118%	58.41%	330
Case-6	1200	1140	940	105%	128%	59.57%	279
Case-7	1140	1140	940	100%	121%	46.27%	447
Case-8	1085	1080	880	100%	123%	56.44%	166
Case-9	1050	1095	895	96%	117%	52.30%	133
Case-10	1220	1140	935	107%	130%	57.68%	255
Average Comparison with CPLEX				99%	121%		4% of 2 hours

Fig. 4.8: Solution Comparison for 10 Cases

- By comparison of the solution time, solution by the auction with PAU and VSS is much faster. It can achieve price equilibrium within 10 minutes while the comparable results by CPLEX within 2 hours.

We will further show that both PAU and VSS are important components in this auction solutions. It is just due to PAU and VSS that we call this auction has adaptive price adjustment. The reason why PAU is better than PAB, has some theory explanation and can been clearly illustrated in previous experiment. However, the theory behind that VSS out-performs fixed-step-size needs further research. Right now, we will show this point empirically as follows.

Solution Comparison

4 Agents start @ same time, COS BiDir-FSP, 10 Forward + 10 Reverse

CaseId	SumCost (Makespan + Tardiness)		PA-U over CPLEX	CPLEX Dual Gap (in 2hours or 7200 seconds)	Solution Time by PAU-VSS (sec)
	PA-U	CPLEX			
Case1	1300	1185	110%	37.11%	305
Case2	1425	1280	111%	24.04%	305
Case3	1385	1265	109%	33.23%	222
Case4	1225	1130	108%	47.89%	292
Case5	1305	1205	108%	32.69%	419
Case6	1395	1250	112%	25.94%	1120
Case7	1335	1275	105%	39.25%	1258
Case8	1275	1185	108%	31.48%	980
Case9	1345	1225	110%	43.30%	264
Case10	1455	1265	115%	23.65%	1420
Average Comparison with CPLEX			110%		9%

Fig. 4.9: Solution Comparison for 2nd group of 10 Cases

Solution Comparison

All 4starts at the same time, alternative COS BiDir-FSP, 10 Forward + 10 Reverse

Case-Id	SumCost (Makespan + Tardiness)		PA-U over CPLEX	CPLEX Dual Gap (in 2hours or 7200 seconds)	Solution Time by PAU-VSS (sec)
	PA-U	CPLEX			
Case1	1560	1445	108%	1.33%	1434
Case2	1645	1550	106%	3.08%	1427
Case3	1680	1535	109%	5.22%	1417
Case4	1445	1350	107%	1.18%	1185
Case5	1585	1430	111%	2.63%	1327
Case6	1675	1590	105%	2.58%	1422
Case7	1685	1520	111%	1.66%	1507
Case8	1545	1430	108%	1.29%	2085
Case9	1620	1505	108%	2.52%	2680
Case10	1755	1600	110%	6.87%	1426
Average Comparison with CPLEX			108%		22%

Fig. 4.10: Solution Comparison for 3rd group of 10 Cases

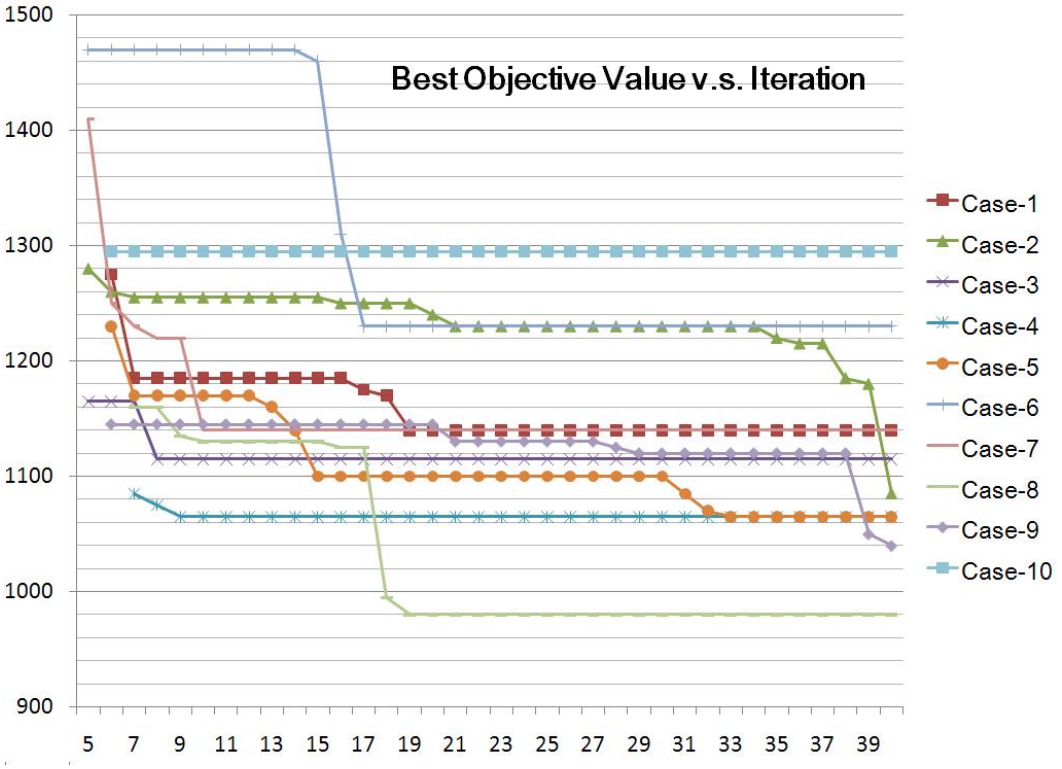


Fig. 4.11: Convergence Study for the Same 10 Cases

4.11.3 Empirical Study on Convergence

Further we study the convergence property empirically by solving first group of 10 problem instance (previous Section 4.11.2). We study the best objective value within 40 iterations. The chart of *Best Objective Value* v.s. *Auction Iteration* is shown in Figure. 4.11.

Based on the experiments' data, we have the following observations.

- The convergence seems to be problem-dependent. There are two cases (*Case-2 and Case-9*), s.t. the best objective value keeps improve until last iteration. On the other hand, there is one special case (*Case-10*), whose best objective value remains at the very beginning feasible solution to the last iteration.

- All the 10 cases can get feasible solutions within 5 to 6 iterations;

- In 7 out of 10 cases, the best objective value keeps the same level from 20 iteration onwards, this is why we usually choose 20 iterations and return the best solutions.

4.11.4 Comparison among different price adjustment strategies

In order to study the effects of utility pricing and variable step size, we focus on a single problem instance taken from the set of problem instances above. A total of 4 price adjustment settings are studied for comparison:

-UP-VSS: with bid price and fixed step size (i.e. no utility price nor variable step size)

+UP-VSS: with utility price and fixed step size;

-UP+VSS: with bid price and variable step size;

+UP+VSS: with both utility price and variable step size.

In order to clearly show the effect of combined utility price and variable step size, we set the initial prices of all machines to be 0. And it is terminated after a maximum of 15 iterations. The result is shown in Fig.4.23 at appendix of this chapter.

From the experiment result shown in Fig.4.23, we have the following observations.

- For the 2 settings that do not use utility price, i.e. *-UP-VSS* and *-UP+VSS*, they fail to get a feasible solution after the maximum iteration. This result matches that in previous section 4.11.1. This point is clear for the formula in Eqn.(4.34). The factor includes two parts α^r and β^r. α^r is related with average price. If bid price is used and the initial condition is zero bid price, α^r will always remain as 0, resulting in the failure to find a feasible solution.

- For the setting of *+UP-VSS* (utility price and fixed step size), a feasible solution is found at the 5th iteration. In this fixed step size price adjustment, the auction process cannot find a better feasible solution thereafter, and the best objective function value remains at the same level until the maximum iteration.

- For the setting of *+UP+VSS* (both utility price and variable step size), a feasible solution is found at the 4th iteration. The objective function value *1205*of this solution is near optimal but not the best one. Subsequently at the 5th iteration, a better feasible solution is found with the objective function value as *1055*. This solution turns out to be the best within the maximum iterations.

In summary, the above experiment shows empirically the power of price adjustment with both utility price and variable step size. The performance of such setting as +UP+VSS exceeds those of all other settings.

4.11.5 Comparison among different variable step size parameters

With exactly the same problem instance, we further study the sensitivity of different parameters for the speed factor function used to compute the variable step size. According to Eqn.(4.38) and Eqn.(4.40), there are two types of speed factor functions, i.e. *exp* and *atan*. The *exp* function has 2 parameters (offset parameter C and index parameter p), while *atan* has 1 parameter (index parameter p).

In total, we tested 10 different problem settings, the first 7 are of *exp* type and last 3 are of *atan* type. For the 7 *exp*-type settings, 1-4 have the same offset parameter value $C = 2$ but different index parameters $p \in \{1, 2, 3, 4\}$. 4-7 have the same index parameter $p = 4$ but different in offset parameter $C \in \{2.0, 1.8, 1.5, 1.0\}$. For the *atan*-type speed function, we test 3 cases, $p \in \{2.0, 3.0, 4.0\}$. The experimental results are shown in

Fig. 4.24 in appendix.

There are the following observations from the results:

- All above settings can achieve good final solutions.
- It seems that offset parameters in the *exp*-type function affect the convergence more than index parameters, although the the final solutions are almost the same.
- The above minor difference is thought to be related with this type of problem. The capacity is an integer, so is the net-demand.

In summary, all the parameter settings are effective to find high quality solutions. Although there is a minor difference in the exact behavior of different speed-function parameters, their final solutions are almost the same. This is good news as it gives evidence that shows that the approach is insensitive to the parameter values. It may be conjectured for the integer property of the problem.

4.11.6 Solution time comparison for larger problems

We further compare the solution time between LPR-Rounding-CH and an Auction Approach on large-scale problems. LPR means Linear Programming Relaxation, and LPR-Rounding-CH is a method proposed in my previous work [4]. CPLEX is not able to solve such problem instances with 4 agents that each agent has as more than 20 forward jobs and 20 reverse jobs. The resulting IP model *ResAlloc-MIP* for the problem has more than 56K integer variables, 113K constraints and 333K non-zeros in the matrix. CPLEX failed to obtain a feasible solution within 5 hours. Instead, both the solution quality and run time are compared between LPR-Rounding-CH and the auction's approach, which return feasible solutions within 20 minutes.

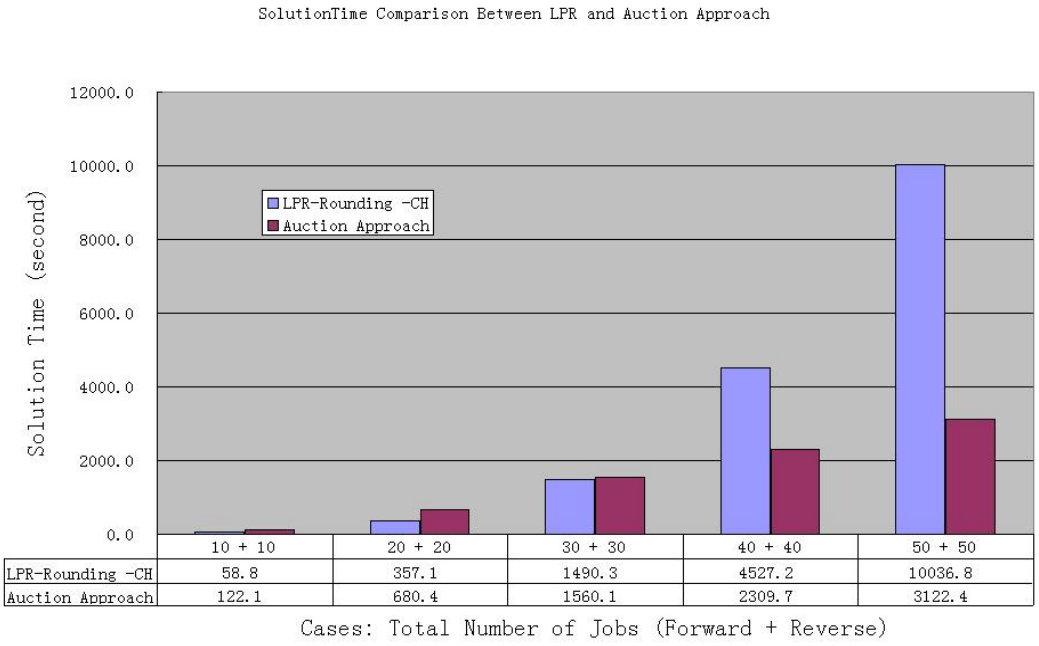


Fig. 4.12: Solution time comparison: auction v.s. LPR-Rounding-CH

Five problem instances have been generated as in [4]. We have [Number of forward jobs + Number of reverse jobs] as a pair, then [10 + 10] is the setting for the first instance, [20 + 20], [30 + 30], [40 + 40] and [50 + 50] for the subsequent instances respectively. The results are shown in Fig. 4.12. From the results, the solution by LPR-Rounding-CH is faster than the Auction Approach, when the problem size is smaller than [40 + 40] jobs. Beyond this size, auction is faster.

With respect to solution quality, auction's approach is better than LPR-Rounding-CH. It is comparable with the ResAlloc-MIP solved by CPLEX while LPR-Rounding-CH is generally worse except for some special cases [4].

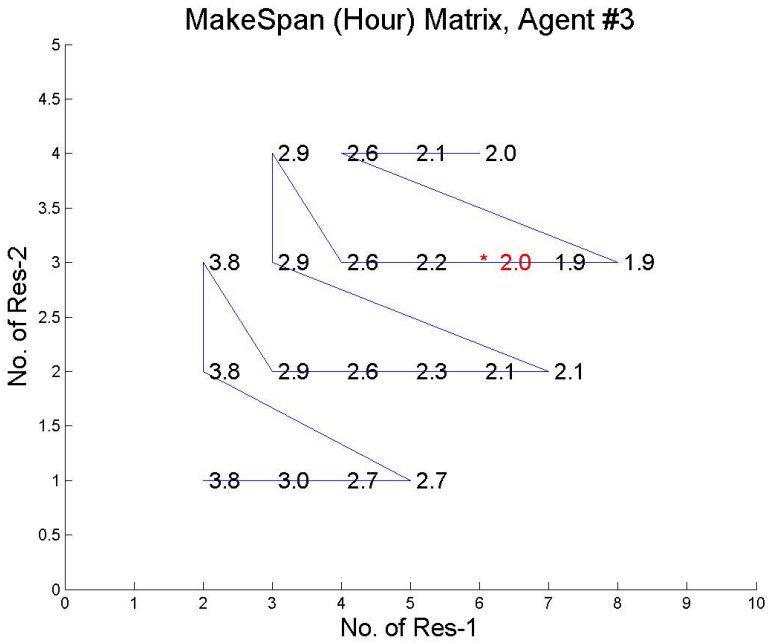
4.12 Conclusion

In this thesis, we present an adaptive tatonnement auction scheme that comprises two key ideas for price adjustment: the concept of utility pricing and variable step size, along

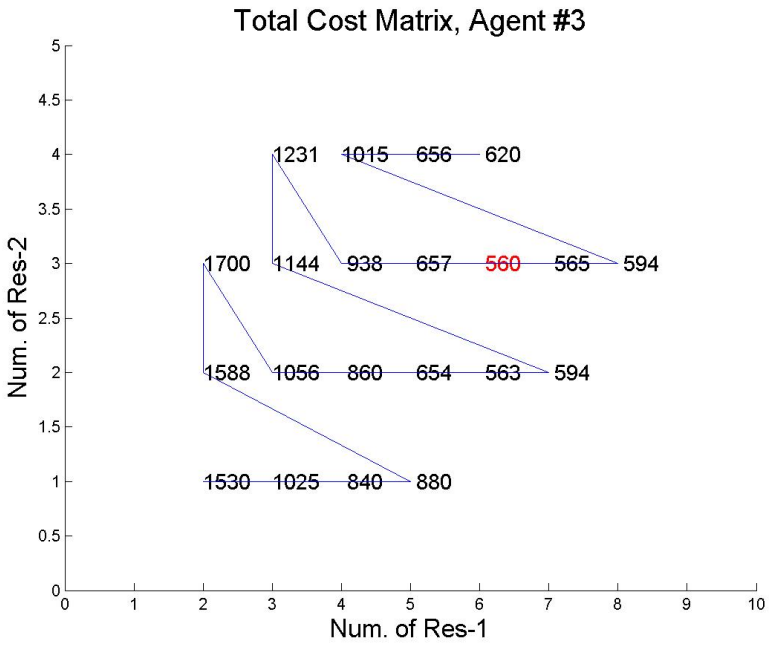
with an algorithm for performing bid generation. Combined with the pre-processing and post-processing steps, we demonstrate the power of the entire system in solving large-scale decentralized scheduling problems. On utility pricing, we demonstrated both analytically and experimentally that it has a better convergence property than conventional price adjustment schemes with bidding price only, and does not depend on initial prices, although the cost is an increased communication overhead between auctioneer and bidders. On variable step size, we show experimentally that it performs better than fixed step size with respect with solution quality and speed, and is insensitive to the underlying speed function parameters.

A salient point to conclude this thesis is that this research is not just computationally efficient and hence readily applicable in handling real time large-scale resource allocation problems, but along with other literature on decentralized/distributed resource scheduling, it enables large-scale resources to be managed in an inherently decentralized fashion through the use of auctions, something not too much in the classical auction literature.

4.13 APPENDIX for Chapter 4: Tables and Figures



(a) Makespan Matrix



(b) Sum Cost Matrix

Fig. 4.13: Makespan Matrix and Single Period Search

Iter.	Num. Res-1	Num. Res-2		Makespan	Total Cost	Notes for change direction or stop
1	2	1		3.83	1530.0	
2	3	1		3.00	1025.0	
3	4	1		2.67	840.0	Makespan not drop, so change to another dir
4	5	1		2.67	880.0	
5	2	2		3.75	1587.5	
6	2	3		3.75	1700.0	
7	3	2		2.92	1056.3	
8	4	2		2.58	860.0	
9	5	2		2.25	653.8	
10	6	2		2.08	562.5	Makespan not drop, so change to another dir
11	7	2		2.08	593.8	
12	3	3		2.92	1143.8	
13	3	4		2.92	1231.3	reach search boundary
14	4	3		2.58	937.5	
15	5	3		2.17	657.5	
16	6	3		2.00	560.0	
17	7	3		1.92	565.4	Makespan not drop, so change to another dir
18	8	3		1.92	594.2	
19	4	4		2.58	1015.0	
20	5	4		2.08	656.3	
21	5	4		2.08	656.3	
22	6	4		2.00	620.0	stop with a local minimum

Fig. 4.14: Tabular explanation of local search in the BidGen

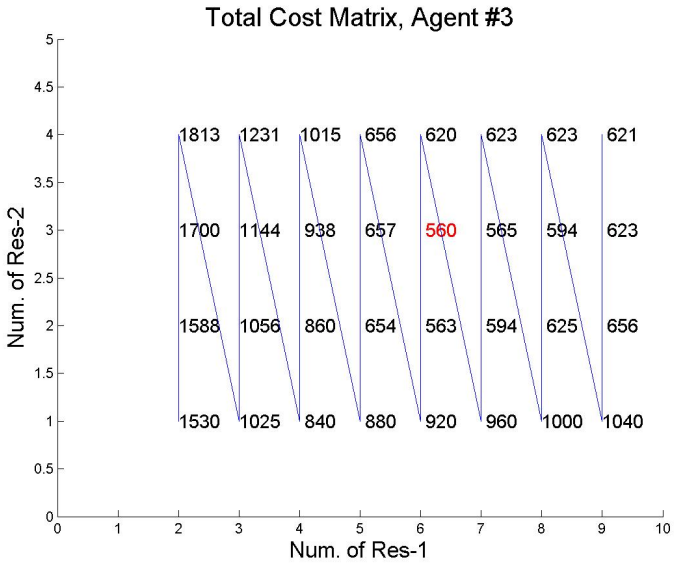
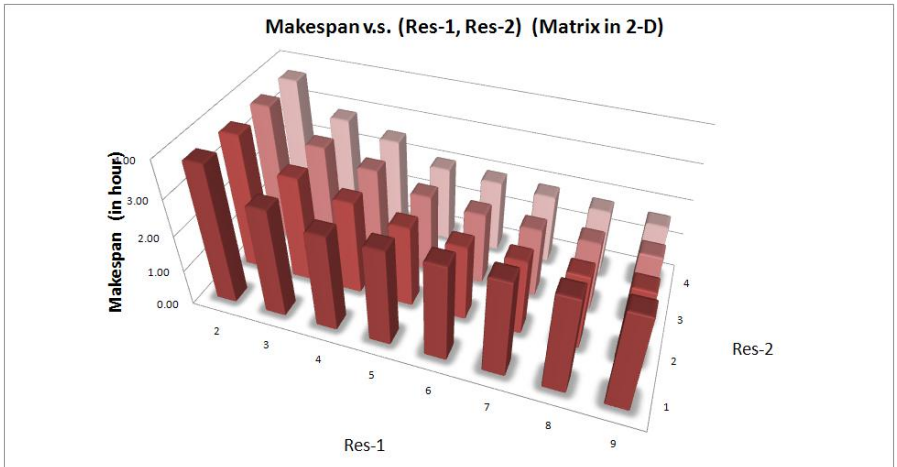


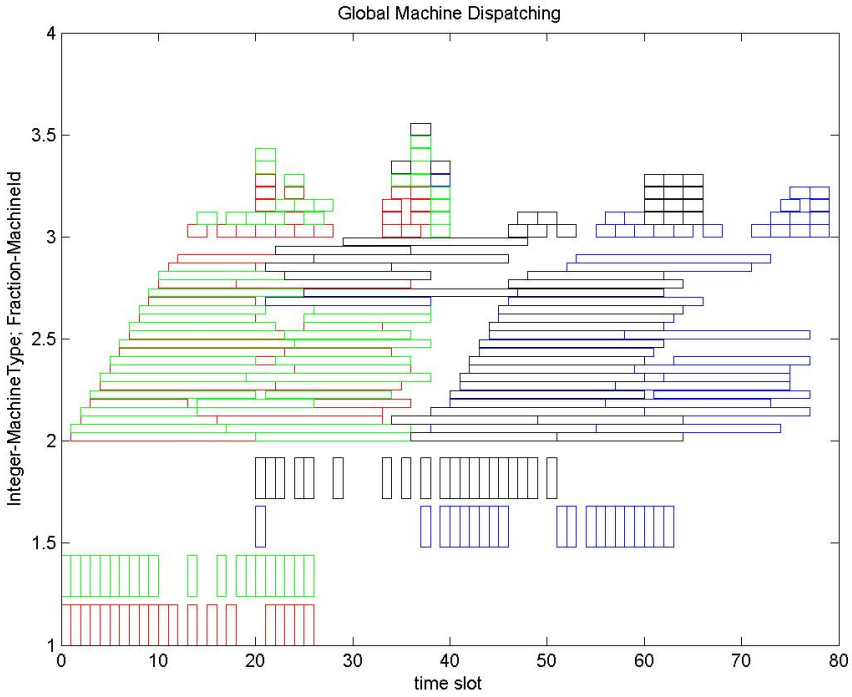
Fig. 4.15: Sum Cost Matrix in Complete Search



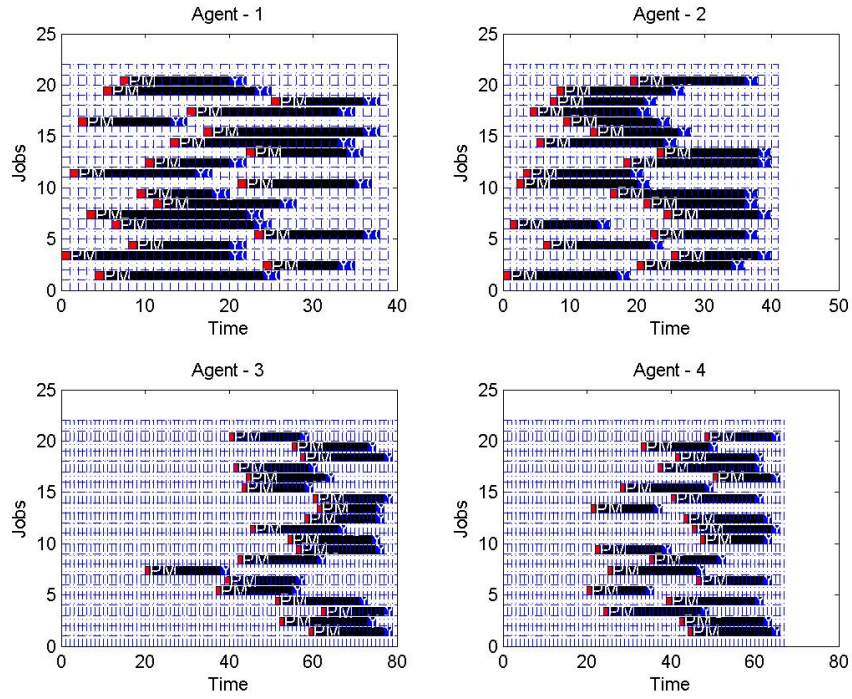
	2	3	4	5	6	7	8	9
1	3.83	3.00	2.67	2.67	2.67	2.67	2.67	2.67
2	3.75	2.92	2.58	2.25	2.08	2.08	2.08	2.08
3	3.75	2.92	2.58	2.17	2.00	1.92	1.92	1.92
4	3.75	2.92	2.58	2.08	2.00	1.92	1.83	1.75

(a) Monotonic, NOT Convex

Fig. 4.16: Detailed Study of Makespan Matrix

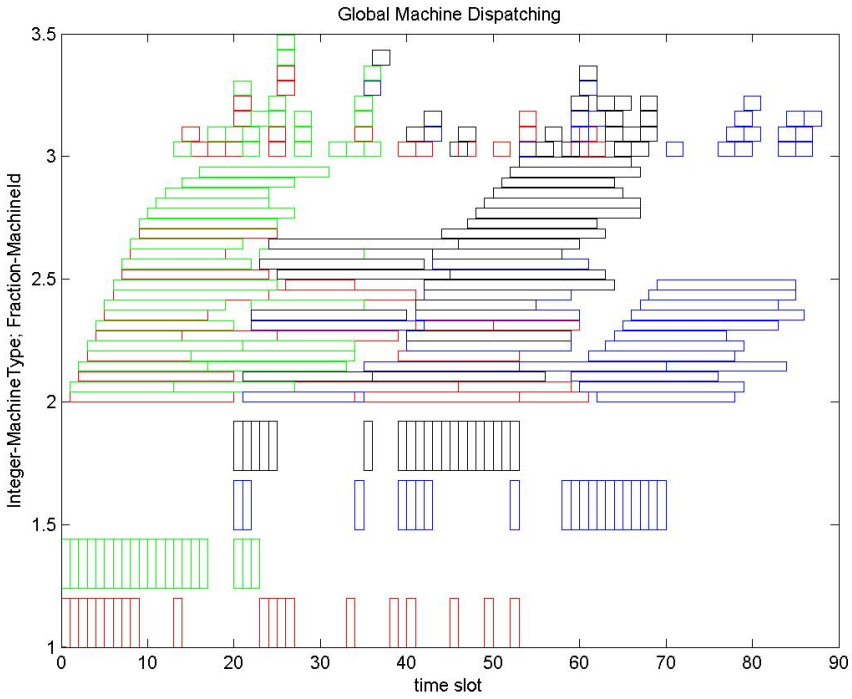


(a) Global Dispatching

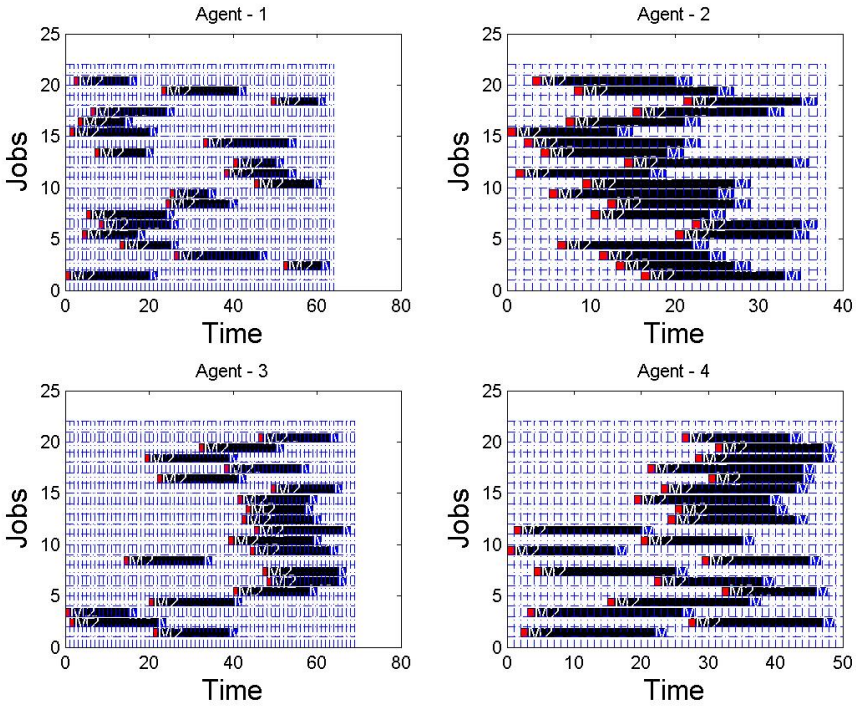


(b) Agent Job Schedule

Fig. 4.17: MIP Solution by CPLEX for 1st group in Sec.4.11.2

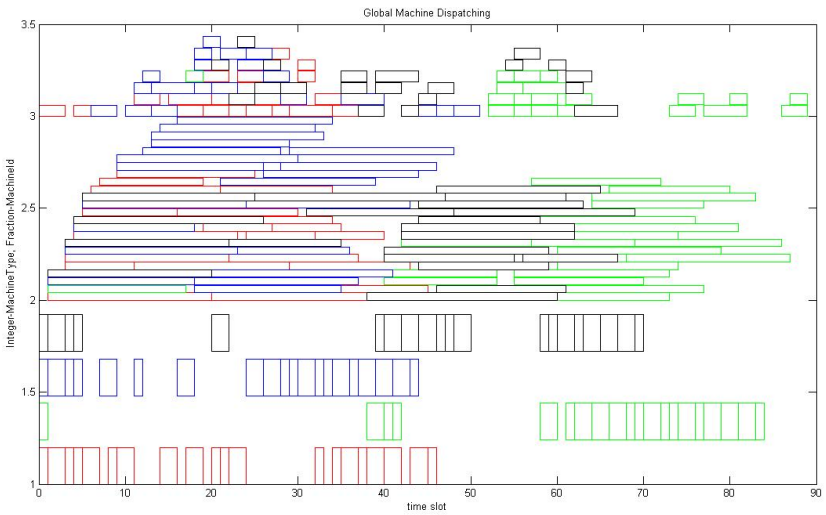


(a) Global Dispatching

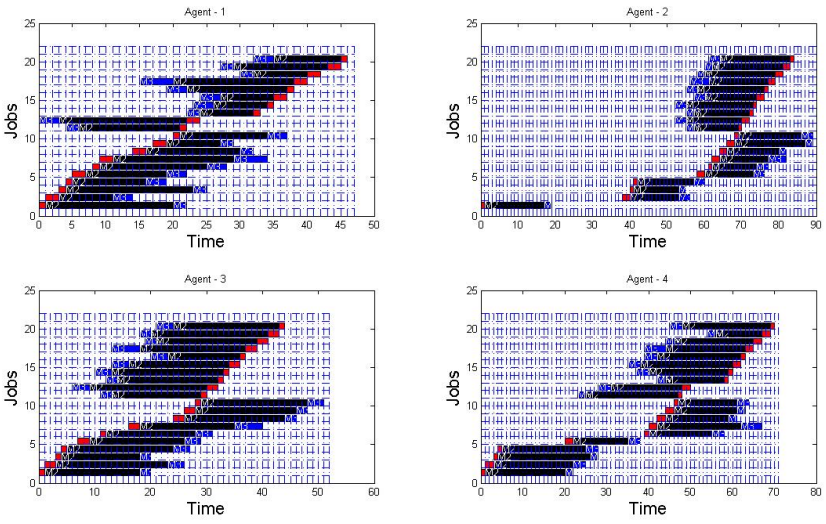


(b) Agent Job Schedule

Fig. 4.18: Auction-PAU-VSS for 1st group in Sec.4.11.2

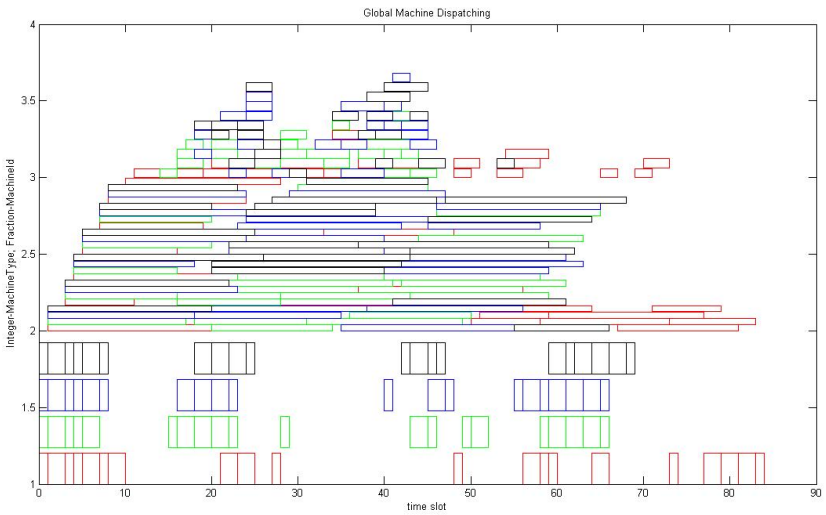


(a) Global Dispatching

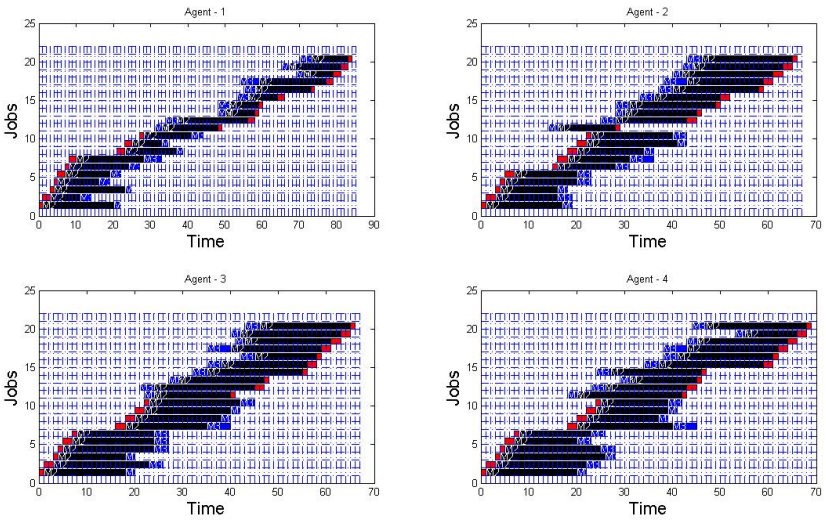


(b) Agent Job Schedule

Fig. 4.19: MIP Solution by CPLEX for 2nd group in Sec.4.11.2

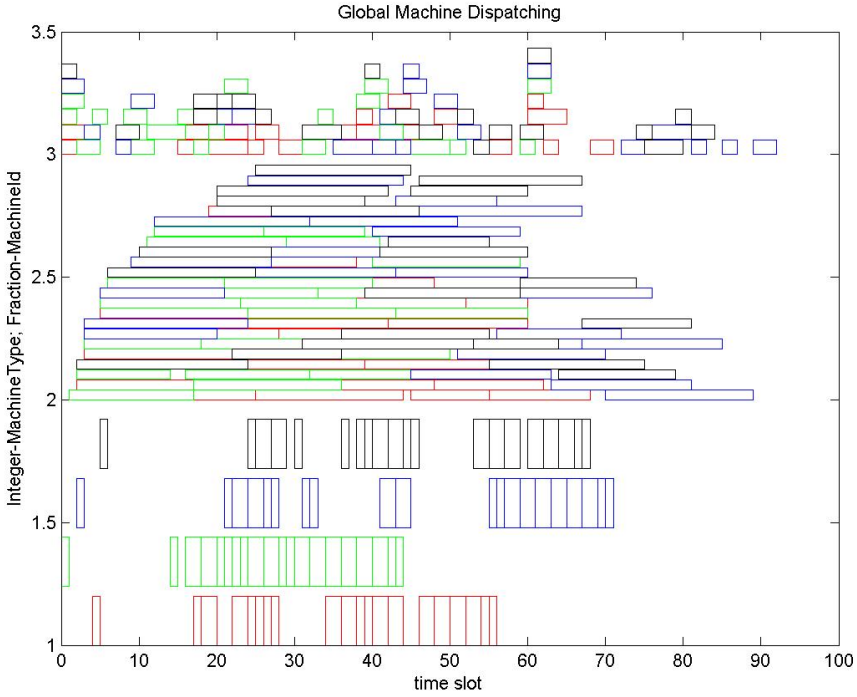


(a) Global Dispatching

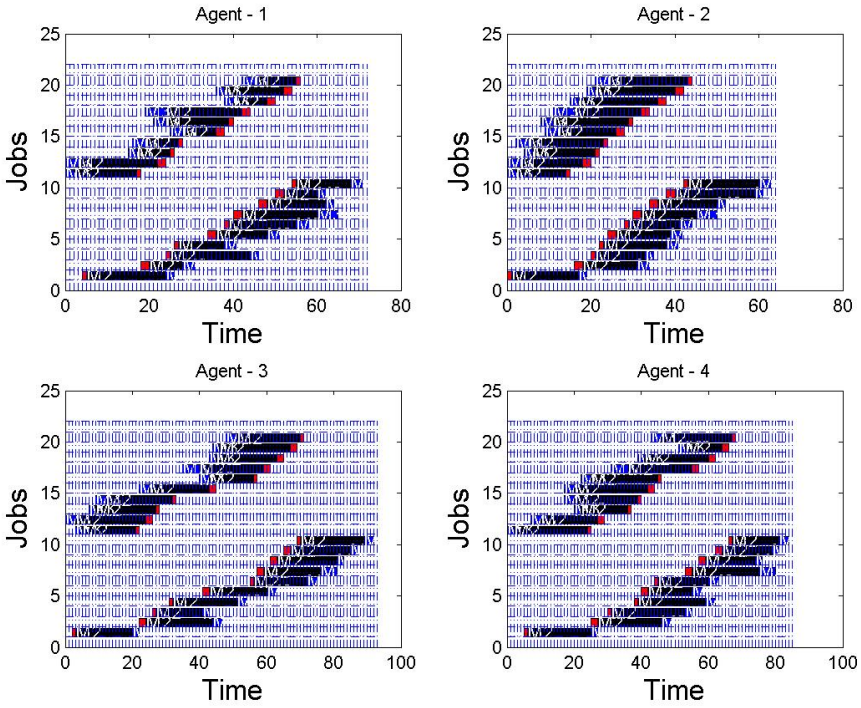


(b) Agent Job Schedule

Fig. 4.20: Auction-PAU-VSS for 2nd group in Sec.4.11.2

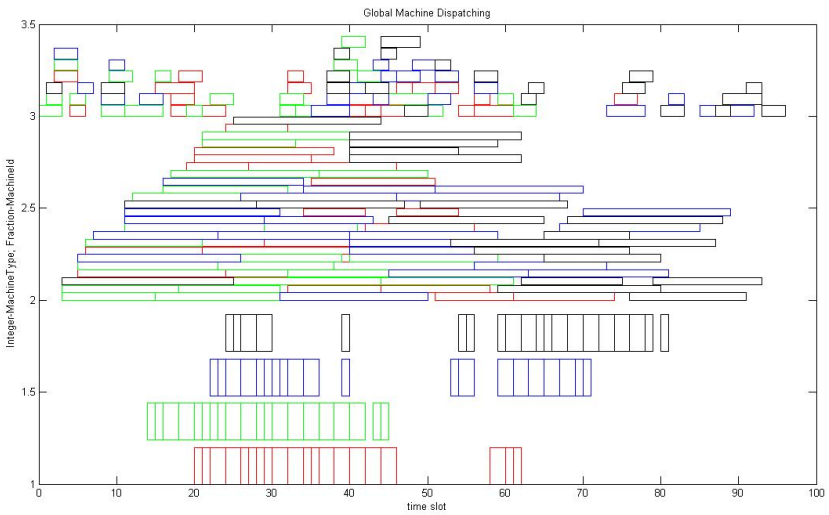


(a) Global Dispatching

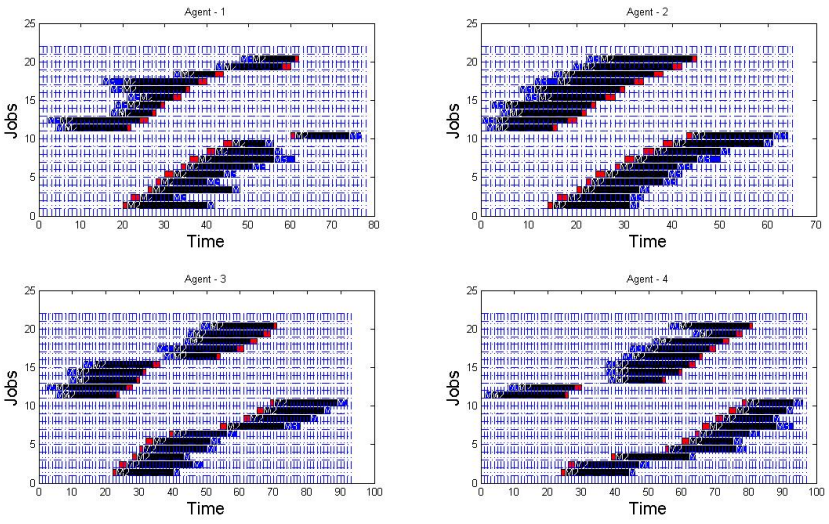


(b) Agent Job Schedule

Fig. 4.21: MIP Solution by CPLEX for 3rd group in Sec.4.11.2

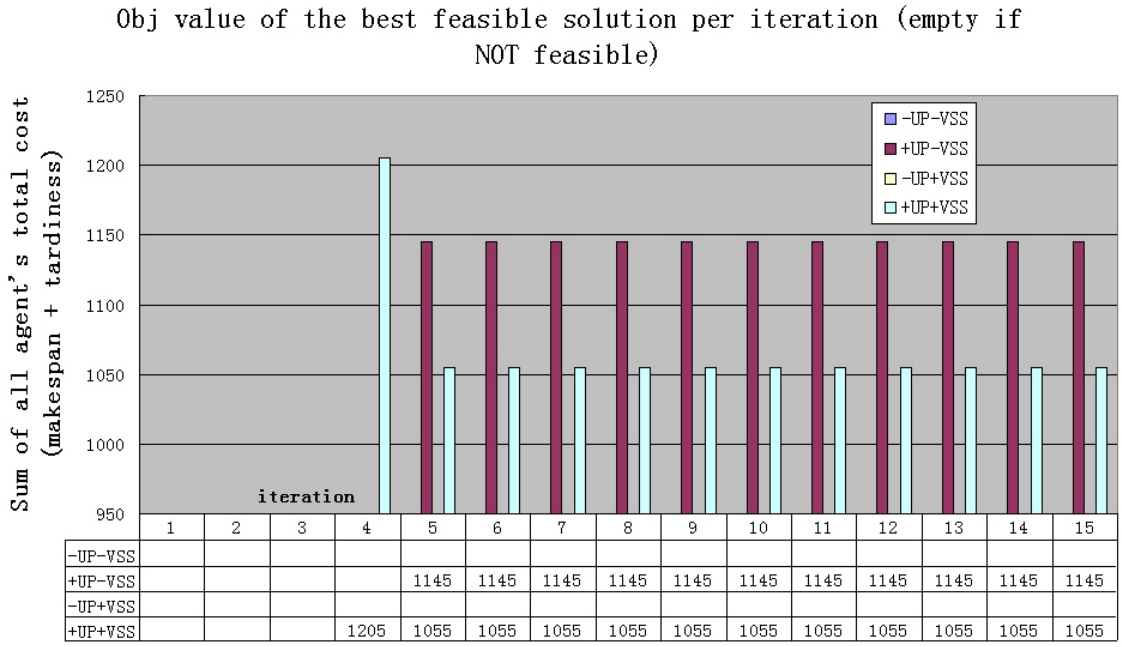


(a) Global Dispatching



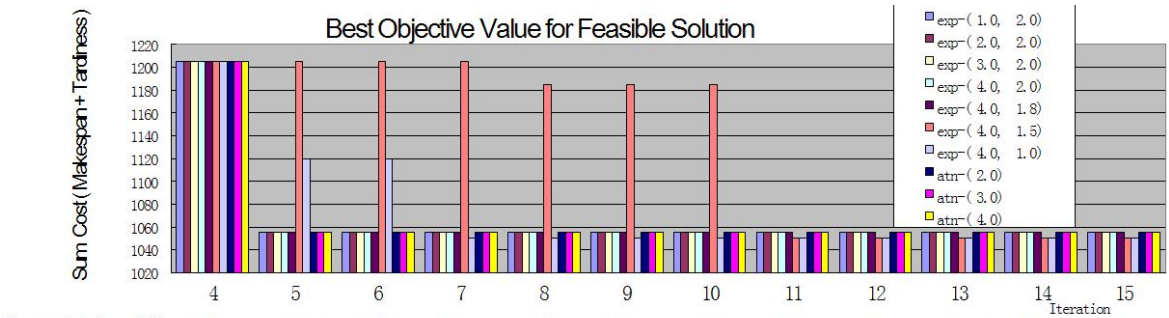
(b) Agent Job Schedule

Fig. 4.22: Auction-PAU-VSS for 3rd group in Sec.4.11.2



iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-UP-VSS															
+UP-VSS					1145	1145	1145	1145	1145	1145	1145	1145	1145	1145	1145
-UP+VSS															
+UP+VSS				1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055

Fig. 4.23: Effect of utility price and variable step size in price adjustment



	4	5	6	7	8	9	10	11	12	13	14	15
exp-(1.0, 2.0)	1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055
exp-(2.0, 2.0)	1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055
exp-(3.0, 2.0)	1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055
exp-(4.0, 2.0)	1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055
exp-(4.0, 1.8)	1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055
exp-(4.0, 1.5)	1205	1205	1205	1205	1185	1185	1185	1050	1050	1050	1050	1050
exp-(4.0, 1.0)	1205	1120	1120	1050	1050	1050	1050	1050	1050	1050	1050	1050
atn-(2.0)	1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055
atn-(3.0)	1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055
atn-(4.0)	1205	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055	1055

Fig. 4.24: Comparisons on different parameter in variable step size in price adjustment

Chapter 5

Conclusion and Further Research Directions

For the classical shop scheduling, we apply Pritsker's approach to get a 0-1 IP formulation. This formulation can be adjusted to problems with constraints either no-wait or infinite-wait-buffer. This formulation is further extended to handle (1) multi-machine problems (job-shop or flow-shop), (2)multi-period problems, (3) COS constraints, (4) resource allocation problems.

For general multi-machine shop scheduling problem, a continuous time method is proposed to construct the machine utilization. This method is fundamental for the whole thesis, it is used in the following aspects:

- to compare the 2 optimal criterion for multi-machine job-shop problems;
- to build machine capacity relaxation-based heuristics (CH and RH) for flow-shop, bi-directional flow-shop with or without COS constraints;
- to construct the auction's approach in resource allocation among several scheduling

agents;

For the multi-machine job shop problem with infinite wait buffer, we proposed a heuristic method to get a fast feasible solution. The solution quality is compared with CPLEX solution with IP formulation. This heuristic is extendable to handle dynamic batch-job arrival.

For the multi-machine Bi-directional flow shop with COS constraints, 2 heuristics (CH and RH)are proposed and compared with existing greedy method and CPLEX solution. Although CH and RH give better solution than Greedy, the computational time is substantially longer. But if compared with the ScheGen-IP solution, the run time for all heuristic methods are relatively much shorter. The heuristic method works fine for pure **Forward Flow Jobs** with small variance in processing time, when it could achieve real optimal solutions.

The COS constraints not only has meaningful applications, but also provides a new point theoretically. Even the job-sequence has been fixed, the heuristic solutions still make a lot of difference from the optimal one. This may serves as complementary methods for genetic algorithm. Optimal solution needs both optimal sequencing and optimal start time. For the integrated resource allocation among several flow-shop scheduling agents, in multi-machine multi-period environment. we propose an adaptive tatonnement auction scheme that comprises two key ideas for price adjustment: the concept of utility pricing and variable step size, along with an algorithm for performing bid generation. Combined with the pre-processing and post-processing steps, the power of the entire system is demonstrated in solving large-scale decentralized scheduling problems. On utility pricing, we demonstrated both analytically and experimentally that it has a better convergence property than conventional price adjustment schemes with bidding price only,

and does not depend on initial prices, although the cost is an increased communication overhead between auctioneer and bidders. On variable step size, we show experimentally that it performs better than fixed step size with respect with solution quality and speed, and is insensitive to the underlying speed function parameters.

For further research, we can see the possibility to handle more real-time dynamic problems, because of the computationally efficiency of the proposed scheduling methods. Especially for the dynamic problems related with communication and data transmission, there are still a lot of works to be carried out. With respect to the auction's approach, more work is needed to handle decentralized/distributed resource scheduling and allocation in dynamic situations. Dynamic situations may be as complicated as incomplete communication, unreliable or damaged data and etc.

That is why we would rather take this thesis as a gate to a wide research area, than declare any best solutions.

5.1 Research Grant and Publications During PhD Candidate

The research for my PhD thesis is partially funded by the A*STAR SERC TSRP Grant numbers P0520101 and P0520104.

Journal Papers

1. Z. J. Zhao, H.C.Lau and S. S. GE, "Integrated Resource Allocation and Scheduling in a Bidirectional flow-shop with Multimachine and COS Constraints", *IEEE Trans. on Systems, Man and Cybernetics - Part C: Applications and Reviews*, Vol.39, No.2, pp.190-200, March 2009.

2. H.C.Lau, Z. J. Zhao, S. S. GE and T.H.Lee, "Allocating resources in multi-agent flow-shops with adaptive auctions", Accepted by *IEEE Trans. Automation Science and Engineering* , 2010.

Patent

1. Z.J. ZHAO, "A Multi-media Transmission Device Based on Power Line Communication" *Chinese Patent*, No. ZL 2007 2 0007199.6., 2008

Conference Papers

1. Z.J. ZHAO and S.S.GE, "High Performance Motion Control Command Generator and Its Application in a 3-link Plane Manipulator", *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation*, pp.3942-3947, 2007.

2. Z.J. ZHAO, T.Y.LEONG, S.S.GE and H.C.LAU, "Bidirectional Flow Shop Scheduling with Multi-Machine Capacity and Critical Operation Sequencing", *Proceedings of 22nd IEEE International Symposium on Intelligent Control*, pp.446-451, 2007.

3. Z.J. ZHAO, J.SUN and S.S.GE, "High Performance Quadratic Classifier and the Application On PenDigits Recognition", *Proceedings of the 46th IEEE Conference on Decision and Control*, pp.3072-3077, 2007.

4. Z.J. ZHAO, J. KIM, M. LUO, H. C. LAU, S. S. GE and J.B. ZHANG "A Heuristic Method for Job-Shop Scheduling with an Infinite Wait Buffer" , *3rd IEEE International Conf. Cybernetics and Intelligent Systems*, pp.1198 - 1203, 2008

5. Lau H. C., Z. J. ZHAO, S.S.GE and T.H.Lee, "Utility Pricing Auction for Multi-period Resource Allocation in Multi-Machine Flow Shop Problems", *Proceedings of ACM-International Conference on Electronic Commerce*, 2008

6. I.R. WIOR, Z.J. ZHAO, M. LUO, J.B. ZHANG, S. S. GE and H.C. LAU "Conceptual framework of a dynamic resource allocation test bed and its practical realization with ProModel, *Proceedings of 2009 IEEE International Symposium on Intelligent Control, Part of 2009 IEEE Multi-conference on Systems and Control"*, pp.1613-1618, 2009

7. Z.J. ZHAO and G.N.WANG "High Order Smooth Motion Command Generation for FFC-Digital Controller", *Proceedings of International Conference on Measuring Technology and Mechatronics Automation, IEEE ICMTMA*, pp.337-340, 2011.

Bibliography

[1] Ayten Turkcan, M. Selim Akturk and Robert H. Storer "Predictive / reactive scheduling with controllable processing times and earliness-tardiness penalties", *IIE Transactions*, Vol.41, pp.1080-1095, 2009.

[2] Toczywski E., I. Zoltowska, "A new pricing scheme for a multi-period pool-based electricity auction", *European Journal of Operational Research*, Vol.197, pp.1051-1062, March 2009.

[3] Wojciech B., Mariusz M., "A fast hybrid tabu search algorithm for the no-wait job shop problem", *Computers & Industrial Engineering* , Vol.56, 1502C1509, 2009.

[4] Z. J. Zhao, H.C.Lau and S. S. GE, "Integrated Resource Allocation and Scheduling in a Bidirectional Flowshop with Multimachine and COS Constraints", *IEEE Trans. on Systems, Man and Cybernetics - Part C: Applications and Reviews*, Vol.39, No.2, pp.190-200, March 2009.

[5] Dipti Srinivasan and Lily Rachmawati, "Efficient Fuzzy Evolutionary Algorithm-Based Approach for Solving the Student Project Allocation Problem", *IEEE Trans. on Education*, Vol.51, No.4, pp.439-447, Nov. 2008.

[6] Heidemarie B., A. Herms, Marc M., Thomas Tautenhahn, Jan Tusch, Frank Werner, "Heuristic constructive algorithms for open shop scheduling to minimize mean flow time", *European Journal of Operational Research,* Vol.189, pp.856-870, 2008

[7] Bibo Yang, Joseph Geunes. "PredictiveCreactive scheduling on a single resource with uncertain future jobs", *European Journal of Operational Research,* 189, pp.1267-1283, 2008.

[8] Kedad-Sidhoum, S., Solis, Y.R. and Sourd, F. "Lower bounds for the earliness-tardiness scheduling problem on parallel-machines with distinct due dates", *European Journal of Operational Research,* 189, pp.1305-1316, 2008.

[9] Lau H. C., Z. J. ZHAO, et al., "Utility Pricing Auction for Multi-period Resource Allocation in Multi-Machine Flow Shop Problems", *Proceedings of ACM-International Conference on Electronic Commerce*, 2008

[10] Ruslan Sadykov, "A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates", *European Journal of Operational Research,* Vol.189, pp1284-1304, 2008

[11] Samani H.A., Fua C., Chai T., Ge S.S., Hang C.C. "Multi-modal Task Apportionment in dynamic multi-factor systems", *Proceedinngs, Chinese Control and Decision Conference, CCDC2008*, pp1692-1697, 2008

[12] P.Ballal, F.Lewis, Mireles Jr. and K.Sreenath "Deadlock Avoidance for Free Choice Multi-Reentrant Flow lines: Critical Siphons & Critical Subsystems" *Proceedings, 18th Mediterranean Conference on Control and Automation*, T08-002,

2007

[13] B.B. Li and L. Wang, "A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling", *IEEE Trans. Syst., Man, Cybern. B*, Vol.37, No.3, pp. 576-691, 2007.

[14] Bonsang Koo, M.Fischer, J.Kunz, "A formal identification and re-sequencing process for developing sequencing alternatives in CPM schedules", *Automation in Construction* Vol.17, pp.75-89, 2007.

[15] Confessore G., S. Giordani, S. Rismondo, "A market-based multi-agent system model for decentralized multi-project scheduling", *Annals of Operations Research*, Vol.150, No.1, pp.115-135, 2007.

[16] C. Fua, S. S. GE, K. D. Do, and K. Lim "Multirobot Formations Based on the Queue-Formation Scheme With Limited Communication", *Ieee Transactions On Robotics,* Vol. 23, No. 6, 2007.

[17] Lau H. C. et al., "Multi-period combinatorial auction mechanism for distributed resource allocation and scheduling", *Proceedings of IAT2007, IEEE/ACM/WIC International Conference on Intelligent Agent Technology*, 2007, pp. 407-411.

[18] Nozha Zribi, I. Kacem, A. ElKamel and P. Borne, "Assignment and scheduling in flexible job-shops by hierarchical optimization", *IEEE Trans. Syst., Man, Cybern. C*, Vol.37, No.4, pp.652-661, 2007.

[19] Shabtay, D. and Steiner, G. "A survey of scheduling with controllable processing times", *Discrete Applied Mathematics,* Vol.155, pp.1643-1666, 2007.

[20] Shabtay, D., Kaspi, M. and Steiner, G. "The no-wait two-machine flow-shop scheduling problem with convex resource-dependent processing times", *IIE Transactions,* Vol.39, pp.539-557, 2007.

[21] Van de Vander S., E.Demeulemeester and W.Herroelen, "A classification of predictive-reactive project scheduling procedures", *Journal of Scheduling*, 10, pp195-207, 2007

[22] Z. J. Zhao et al., "Bidirectional flow shop scheduling with multi-machine capacity and critical operation sequencing", *Proceedings ISIC'2007, International Symposium on Intelligent Control*, 2007, pp.446-451.

[23] V.Giordano, P.Ballal, F.Lewis, B.Turchiano and J.B.Zhang "Supervisory Control of Mobile Sensor Networks: Math Formulation, Simulation and Implementation", *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, Vol.36, No.4, pp806-819, 2006

[24] Herrmann J. W. *Handbook of Production Scheduling*, Chap-1, Springer US, 2006.

[25] A. G. Kristina Lerman1, Chris Jones and M. J. Matari, "Analysis of Dynamic Task Allocation in Multi-Robot Systems", *International Journal of Robotics Research,* pp. 225-242, March 2006.

[26] Liu J., W.C. Zhong and L.C. Jiao, "A multiagent evolutionary algorithm for constraint satisfaction problems", *IEEE Trans. Syst., Man, Cybern. B*, Vol.36, No.1, pp.54-73, 2006.

[27] Shabtay, D. and Kaspi, M. "Parallel-machine scheduling with a convex resource consumption function", *European Journal of Operational Research,* 173, pp.92-107, 2006

[28] Schuster, C. "No-wait job shop scheduling: Tabu search and complexity of subproblems", *Mathematical Methods of Operations Research,* Vol.63, No.3, pp.473C491, 2006

[29] P. Vansteenwegen and D. Van Oudheusden "Developing railway timetables which guarantee a better service", *European Journal of Operational Research,* 173, pp.337C350, 2006

[30] S. Van De Vonder, E. Demeulemeester, W. Herroelen and R. Leus "The trade-off between stability and makespan in resource-constrained project scheduling", *International Journal of Production Research,* Vol. 44, No. 2, pp.215-236, 2006.

[31] Wong, T.N., Leung, C.W., Mak, K.L., and Fung R.Y.K., "An agent-based negotiation approach to integrate process planning and scheduling", *International Journal of Production Research,* Vol. 44, No. 7, pp. 1331-1351, 2006.

[32] Ada Che, C. Chu, "Multi-Degree cyclic scheduling of two robots in a no-wait flowshop", *IEEE Transactions on Automation Science and Engineering*, Vol.2, No.2, pp. 173-183, 2005.

[33] Aytug, H., Lawley, M.A., Mckay, K., Mohan, S. and Uzsoy, R., "Executing production schedules in the face of uncertainties: a review and some future directions", *European Journal of Operations Research,* Vol. 161, pp. 86-110, 2005

[34] Bish E. K. et al., "Dispatching vehicles in a mega container terminal", *OR Spectrum,* Vol.27, No.4, pp. 491-506, 2005.

[35] C. *Oğuz* and M. F. Ercan, "A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks", *Journal of Scheduling*, vol. 8, no. 4, pp. 323–351.

[36] C. Fua and S. S. GE, "COBOS: Cooperative Back-Off Adaptive Scheme for Multi-Robot Task Allocation", *IEEE Transactions on Robotics,* Vol.21, No.6, pp.1168-1178, 2005

[37] S. S. GE and C.H. Fua, "Queues and Artificial Potential Trenches for Multi-Robot Formations", *IEEE Trans. Robotics,* VOL. 21, NO. 3, pp.646-656, 2005.

[38] Katta G. Murty et al., "HongKong International Terminal Gains Elastic Capacity Using a Data-Intensive Decision-Support System", *Interfaces*, Vol.35, No.1, pp. 61-75, 2005.

[39] Sourd,F. "Earliness-tardiness scheduling with setup considerations", *Computers & Operations Research,* 32, pp.1849-1865, 2005.

[40] L. Chaimowicz, V. Kumar, and M. F. M. Campos "A Paradigm for Dynamic Coordination of Multiple Robots", *Autonomous Robots,* vol. 17, pp. 7-21, 2004.

[41] Cicirello, V. and Smith, S. F., "Wasp-based agents for distributed factory coordination", *Journal of Autonomous Agents and Multi-Agent Systems,* Vol. 8, No.3, pp. 237-266, 2004

[42] Kutanoglu E., S. David Wu, "Improving Scheduling Robustness via Preprocessing and Dynamic Adaptation", *IIE Transactions*, vol.36, pp.1107-1124, 2004.

[43] Shabtay,D. and Kaspi,M. "Minimizing the totalweighted flowtime in a single machine with controllable processing times", *Computers &Operations Research,* 31, pp.2279-2289, 2004.

[44] Bish E. K., "A multiple-crane-constrained scheduling problem in a container terminal", *European Journal of Operational Research* Vol.144, pp. 83-107, 2003.

[45] Sabuncuoglu I. and O.B.Kizilisik, "Reactive scheduling in a dynamic and stochastic FMS environment", *International Journal of Production Research*, Vol.41, No.17, pp4211-4231, 2003

[46] Schuster, C. and Framinan, J. "Approximative procedures for no-wait job shop scheduling", *Operations Research Letters,* Vol.31, No.3, pp.308C318, 2003

[47] Turkcan, A., Akturk, M.S. and Storer, R.H. "Non-identical parallel CNC machine scheduling", *International Journal of Production Research,* Vol.41, No.10, pp.2143-2168, 2003

[48] Vieira, G.E., Herrmann, J.W. and Lin, E. "Rescheduling manufacturing systems: a framework of strategies, policies and methods", *Journal of Scheduling,* Vol.6, No.1, pp.39-62, 2003.

[49] Brucker P., "Scheduling and constraint propagation", *Discrete Applied Mathematics*, Vol.123, pp. 227-256, 2002

[50] B. P. Gerkey and M. J. Matari, "Sold!: Auction Methods for Multirobot Coordination", *IEEE Transactions on Robotics and Automation,* vol. 18, No. 5, pp. 758-768, October 2002.

[51] Kacem I., S. Hammadi, P. Borne, "Pareto-optimality Approach for Flexible Job-shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic", *Mathematics and Computers in Simulation*, Vol.60, pp. 245-276, 2002.

[52] Imed Kacem, S. Hammadi, P. Borne, "Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems", *IEEE Trans. Syst., Man, Cybern. C*, Vol.32, No.1, pp. 1-13, 2002.

[53] Peter B., "Scheduling and constraint propagation", *Discrete Applied Mathematics*, Vol.123, pp. 227-256, 2002.

[54] Sun, J., and Xue, D. "A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources", *Computers in Industry,* Vol.46, No.2, pp.189-207, 2001.

[55] Tharumarajah, A. "Survey of resource allocation methods for distributed manufacturing systems", *Production Planning and Control,* Vol.12, No.1, pp.58-68, 2001.

[56] Wellman M. P., W. E.Walsh, P. R. Wurman and J.K. MacKie-Mason. "Auction protocols for decentralized scheduling", *Games and Economic Behavior*, 35, 2001.

[57] Zhou, H., Feng, Y., and Han, L. "The hybrid heuristic genetic algorithm for job shop scheduling", *Computers and Industrial Engineering,* Vol.40, No.3, pp.191-200, 2001.

[58] Sabuncuoglu I., M.Bayiz "Analysis of reactive scheduling problems in a job shop environment", *European Journal of Operational Research*, 126, pp567-586, 2000

[59] Vieira G. E., Herrmann J. W. and Lin E. "Analytical models to predict the performance of a single-machine system under periodic and event-driven rescheduling strategies", *International Journal of Production Research*, Vol.38 No.8, pp1899-1915, 2000.

[60] Bierwirth C. and DC Mattfeld "Production scheduling and rescheduling with genetic algorithms", *Evolutionary computation* Vol.7, No.1, pp1-17, 1999

[61] David Wu S., E.S. Byeon, R. H. Storer, "A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness", *Operations Research*, Vol. 47, No.1, 113-124, 1999.

[62] Kloeden P. E., E. Platen. "Numerical Solution of Stochastic Differential Equations". Edition 3, Springer, 1999.

[63] E. Kutanoglu and S. D. Wu. "On combinatorial auction and Lagrangean relaxation for distributed resource scheduling", *IIE Transactions*, 31, 1999.

[64] I. Sabuncuoglu, M. Bayiz "Job shop scheduling with beam search", *European Journal of Operational Research* , Vol.118, No.2, pp390-412, 1999

[65] M. Selim Akturk, Elif Gorgulu "Match-up scheduling under a machine breakdown", *European Journal of Operational Research* , Vol.112, No.1, pp81-97, 1999

[66] Bartusch M., R.H. Mohring and F.J. Radermacher, "Scheduling project networks with resource constraints and time windows", *Annals of Operations Research*, Vol.16, pp.201-240, 1988

[67] Byeon E.S., Wu, S.D., Storer, R.H. "Decomposition heuristics for robust job-shop scheduling", *IEEE Transactions on Robotics and Automation*, Vol.14, No.2, pp303-313, 1998

[68] Cheng J. Q. and M. P. Wellman. "The Walas algorithm: A convergent distributed implementation or general equilibrium outcomes", *Computational Economics*, 12, 1998.

[69] Mehta, S.V. and Uzsoy, R.M. "Predictable scheduling of a job shop subject to breakdowns", *IEEE Transactions on Robotics and Automation*, Vol.14, No.3, pp365-378, 1998

[70] L. E. Parker "ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation", *IEEE Transactions on Robotics and Automation,* vol. 14, no. 2, pp. 220-240, April 1998.

[71] Bikhchandani S. and J. W. Mamer. "Competitive equilibrium in an exchange economy with indivisibilities", *Journal of Economic Theory*, 74(2):385–413, 1997.

[72] Gaines J. G., T. J. Lyons. "Variable Step Size Control in the Numerical Solution of Stochastic Differential Equations", *SIAM Journal On Applied Mathematics*, 57:5:1455-1484, 1997.

[73] Levner E., V. Kats, and V. E. Levit, "An improved algorithm for cyclic scheduling in a robotic cell", *European Journal of Operational Research*, Vol. 97, No.3, pp. 500-508, March 1997.

[74] Gagliano R., M. Fraser and M. Schaefer. "Auction allocation of computing resources", *Communication of the ACM*, 38(6):88–102, 1995.

[75] Wu, H.H. and Li, R.K. "A new rescheduling method for computer based scheduling systems", *International Journal of Production Research*, Vol. 33 Issue 8, p2097-2110, 1995

[76] Kim, Min Hee, Kim, Yeong-Dae. "Simulation-based real-time scheduling in a flexible manufacturing system", *Journal of Manufacturing Systems,* Vol. 13, No,2, pp. 85-94, 1994.

[77] Bengu G., "A simulation-based scheduler for flexible flowlines", *International Journal of Production Research*, Vol.32, No.2, pp321-344, 1994

[78] Leon V. J., Wu S. D. and Storer, R. H. "Robustness measures and robust scheduling for job shops", *IIE Transactions* Vol.26, No.5, pp. 32-43, 1994

[79] I. M. Ovacikt; R. Uzsoy "Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times", *International Journal of Production Research,* Vol. 32, No.6, pp1243 - 1263, 1994

[80] Li R.K., Shyu Y.T. and S. Adiga "A heuristic rescheduling algorithm for computer-based production scheduling systems", *International Journal of Production Research,* Vol.31, No.8, pp1815-1826, 1993.

[81] Matsuura H., Tsubone H.; Kanezashi, M. "Sequencing, dispatching and switching in a dynamic manufacturing environment", *International Journal of Production Research,* Vol. 31, No.7, pp1671 - 1688, 1993

[82] Laura K. Church and Reha Uzsoy "Analysis of periodic and event-driven rescheduling policies in dynamic shops", *International Journal of Computer Integrated Manufacturing* , Vol.5, No.3, pp153 - 163, 1992.

[83] Applegate D., W. Cook. "A computational study of the job-shop scheduling problem", *ORSA J. Computing* 3, pp. 149-156. 1991.

[84] Bean, James C., Birge, John R., Mittenthal, John, Noon, Charles E, "Matchup Scheduling With Multiple Resources, Release Dates And Disruptions", *Operations Research* Vol.39, No.3, pp470-484, 1991.

[85] Kiran, Ali S., Alptekin, Sema, Kaplan, A. Celal, "Tardiness heuristic for scheduling Flexible Manufacturing Systems", *Production Planning & Control,* Vol. 2, No.3, pp228-241, 1991.

[86] Nof, Shimon Y., Grant, F. Hank. "Adaptive/predictive scheduling: review and a general framework", *Production Planning & Control,* Vol. 2, No.4, pp298 - 313, 1991.

[87] Dutta, A., "Reacting to scheduling exceptions in FMS environments", *IIE Transactions* Vol.22, No.4, pp.300-314, 1990.

[88] Carlier J. and E. Pinson, "An algorithm for solving the job-shop problem", *Management Science* Vol. 35, pp. 164-176, 1989.

[89] S.Y. David Wu; Richard A. Wysk "An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing", *International Journal of Production Research,* Vol. 27, No.9, pp1603 - 1623, 1989

[90] Adams J., E. Balas, D.Zawack. "The shifting bottleneck procedure for job shop scheduling", *Management Science*, 34, pp. 391-401, 1988.

[91] Dimitris Bertsima. J. N. T. "Introduction to Linear Optimization", *Massachusetts, U.S.A.: Athena Scientific, Belmost,* 1988.

[92] Vepsalainen A.P.J., and T.E. Morton, "Priority rules for job shops with weighted tardiness costs", *Management Science*, Vol.33, No.8, 1035 - 1047.

[93] Fisher M. L. "An application oriented guide to Lagrangian relaxation", *Interfaces*, 15(2):pp.10–21, 1985.

[94] M. Yamamoto, S. Y. Nof, "Scheduling/rescheduling in the manufacturing operating system environment", *International Journal of Production Research,* Vol. 23, No.4, pp705 - 722, 1985

[95] P. Joyce. "The Walarsian tatonnement mechanism and information", *RAND Journal of Economics*, 15(3):pp.416–425, 1984.

[96] Panwalkar S. S. and C. R. Woollam, "Ordered flow shop problems with no in-process waiting: further results", *The Journal of the Operational Research Society*, Vol.31, No.11, pp.1039-1043, 1980.

[97] Farn, C.K., Muhleman, A.P., "The dynamic aspects of a production scheduling problem", *International Journal of Production Research,* Vol.17, No.1, pp15 - 21, 1979

[98] Garey M.R.and D.S.Johnson, *Computer and intractability, a guide to the theory of NP-completeness.* Freeman, San Francisco, 1979.

[99] Graham R.L., E.L. Lawler, J.K.Lenstra, A.H.G. Rinnooy Kan, "Optimization And Approximation in Deterministic Sequencing And Scheduling: a survey", *Annuals of Discrete Mathematics*, pp.287-326, 1979.

[100] Gonzalez T. and S. Sahni, "Flowshop and jobshop Schedules: Complexity and Approximation", *Operations Research*, Vol.26, pp. 36-52, 1978.

[101] Nelson R.T., Holloway C.A. and Wong R.M. "Centralized scheduling and priority implementation heuristics for a dynamic job shop model", *AIIE Transactions* Vol.9, No.1, 1977

[102] Rinnooy A.H.G. Kan, *Machine scheduling problems: classification, complexity and computations*, Nijhoff, The Hague, 1976.

[103] Bestwick P.F. and K.G.Lockyer, "Glossary of Terms for the General Scheduling Problem", *Operational Research Quarterly*, Vol.26, No.3, pp.564-565, 1975.

[104] Baker K.R., *Introduction to sequencing and scheduling.* John Wiley, New York, 1974.

[105] Held M., P. Wolfe, H. P. Crowder "Validation of Subgradient Optimization", *Mathematical Programming*, 6:62-88, 1974.

[106] Holloway, C. A., Nelson and R. T., "Job Shop Scheduling With Due Dates And Variable Processing Times", *Management Science* Vol. 20 No.9, pp1264-1275, 1974.

[107] Goyal S.K., "A Note on the Paper: on the Flowshop Sequencing Problem with no wait in process", *Opertional Research*, Q.24, pp130-133, 1973.

[108] Reddi S.S.and C.V. Ramamoorthy, "On the flowshop sequencing problem with no wait in process", *Operational Research*, Q.23, pp323-331, 1972.

[109] Pritsker A., L. Watters, and Ph. Wolfe, "Multiproject scheduling with limited resources: a zero-one programming approach", *Management Science: Theory*, Vol.16, No.1, pp. 93-108, 1969.

[110] Muth J.F. and G.L. Thompson (eds.) *Industrial scheduling*, Prentice-Hall, Englewood Cliffs, NJ. 1963.

[111] Jackson J.R., "An Extension of Johnson's Results on Job Lot Scheduling", *Naval Research Logistics Quarterly*, 3, pp201-203, 1956.

[112] Walras L. *Elements of pure economics.* Homewood, Irwin, 1954.