2009

Integrated Resource Allocation and Scheduling in Bidirectional Flow Shop with Multi-Machine and COS Constraints

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

Zhengyi ZHAO
Singapore Management University

Shuzhi Sam Ge
National University of Singapore

DOI: https://doi.org/10.1109/TSMCC.2008.2007500

Citation

LAU, Hoong Chuin; ZHAO, Zhengyi; and Ge, Shuzhi Sam. Integrated Resource Allocation and Scheduling in Bidirectional Flow Shop with Multi-Machine and COS Constraints. (2009). *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*. 39, (2), 190-200. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/775

Integrated Resource Allocation and Scheduling in Bidirectional Flow Shop with Multi-Machine and COS Constraints

ZhengYi John ZHAO   Hoong Chuin LAU   Shuzhi Sam GE

Abstract—**An IP (Integer Programming) Model is proposed for integrated resource allocation and operation scheduling for a multiple job-agents system. Each agent handles a specific job-list in a bi-directional flowshop. For the individual agent scheduling problem, a formulation is proposed in continuous time domain and compared with an IP formulation in discrete time domain. Of particular interest is the formulation of the machine utilization function - both in continuous time and in discrete time. Fast heuristic methods are proposed with the relaxation of the machine capacity. For the integrated resource allocation and scheduling problem, a Linear Programming Relaxation (LPR) approach is applied to solve the global resource allocation and fast heuristic method is applied to solve each scheduling sub-problems. The proposed solution is compared experimentally with that from the IP solver by CPLEX.**

Index Terms—**Flow Shop Scheduling, Multiple Machine, Resource Allocation**

I. Introduction

A. Model abstraction

We are concerned with a system of multiple agents with each handling a job-list. In this paper, we deal specifically with the setting where all job-lists are bidirectional flow shops **(BiFSP)** under Critical Operation Sequencing **(COS)** constraints. This model is useful for a variety of logistic applications, such as container terminal operation, forward and reverse logistics and etc. TABLE I gives further details of the mapping between our model and possible applications.

TABLE I
Applications for BiFSP Model

Sample app.	Object to be Serviced	Pre-processing	Trans-portation	Post-processing	Critical operation
Port	Container	Un-stack from ship onto truck Quay Crane	Transport to yard	Stack from truck to yard by Yard Crane	Quay Crane Operation
Logis-tics	Items for delivery	Load from yard to truck	deliver to customer	Unload from truck to customer	Yard Operation in the port

More precisely, our 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;
(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 paper, 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, we call it a **Forward Flow Job** and otherwise, we call it 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.

Based on the above definition of BiFSP with multiple renewable machines and COS constraints, *the resource allocation problem* is to allocate the multiple renewable machines to contending job agents, while *scheduling* is to generate a schedule for each job agent with the allocated resources. Given the job release time, due time, and tardiness penalty (or priority), the objective is to minimize the weighted sum of makespan cost and tardiness penalty. The system overview and data flow is shown in Fig. 1.

One sample instance of our problem is described in TABLE II. In this example, there are 4 job lists and 3 types of machines. The operation on the 1^{st} machine has strict sequencing

Part of the paper and some preliminary results have appeared in IEEE conferences [13], [20].

This research is partially funded by the Singapore's A*STAR (Agency for Science and Technology Research) SERC TSRP GRANT P0520104 and IMSS TSRP 052 116 0075.

Z. J. ZHAO is a Research Engineer at School of Information Systems, Singapore Management University and currently a PhD candidate at the Electrical and Computer Engineering Department of the National University of Singapore. Email: zzytgx@yahoo.com.cn

Correspondence: A Prof H.C.LAU, School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902. Tel.: +65-68280229. Fax: +65-68280919 Email: hclau@smu.edu.sg

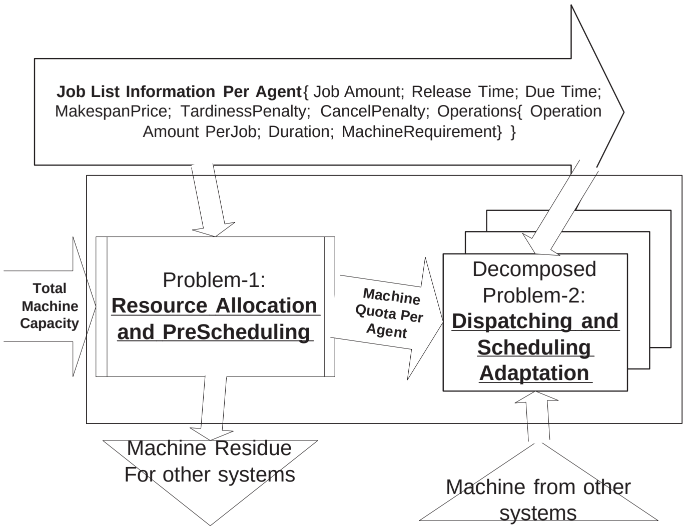

Fig. 1. System Overview: Resource Allocation and Scheduling for Multiple Agents

constraint, so the 1^{st} machine is regarded as the agent. The 2^{nd} type of machine is for transportation and the 3^{rd} type of machine is for postprocessing in forward jobs and preprocessing in reverse jobs. They are resources to be shared by all the job agents. The capacity for the 2^{nd} machine is 16 and that for the 3^{rd} machine is 8. The start time, due time of each job is in Job information of 4 agents part of TABLE II, together with the tardiness penalty W_d and makespan price factor W_m, which are metrics to denote the importance of job-lists. The job-lists of agents 1 / 2 /3 and 4 are shown in TABLE II. Every job-list has 5 forward jobs followed by 5 reverse jobs. It is clear from the job-list that a forward job takes operations on machines of 1^{st}, 2^{nd} and 3^{rd} types sequentially while a reverse job's operation takes place on machines 3^{rd}, 2^{nd} and 1^{st} types sequentially.

The above problem is abstracted from a real-world container terminal resource management problem, where there are 4 operating quay cranes (QC, machine type 1) operating in a berth, and prime movers (PM, 2^{nd} machine type) and yard cranes (YC, 3^{rd} machine type) are used to discharge containers (first 5 jobs) and load containers (next 5 jobs) between the vessel and the yard. This can also be a case for a 3PL workflow process involving loading, delivery and unloading.

The objective function is to minimize the sum of the weighted tardiness penalty and makespan cost. The makespan T_{MS} for such a BiFSP with COS constraints is the duration between the completion time of the last critical operation and the overall start time of the job-list. Tardiness is defined as $\max\{0, T_{MS} - D\}$, where D is the due time of the job-list. Note that this definition differs marginally from the literature [15]. For the container terminal case, once the last container is loaded onto the ship, the ship could leave, while the last container discharged from the ship may still be in the midst of transportation in the yard.

The solution is presented later at Section VII-A.

We like to comment that the resource allocation and schedule is a pre-plan and serves as a guideline for real-time dispatch. It does not consider dynamics such as machine breaking down, or operation time variation due to traffic congestion. Neither does it consider such cases as more resource available from other agents, more agents with job-lists joining the system in real

TABLE II
SAMPLE PROBLEM WITH 3 MACHINES, 4 AGENTS, EACH WITH 10 JOBS

Job-list of 4 agents

Job Id	$[p_{ij}, m_{ij}]$			Note
	Proc-1	Proc-2 Agent-1/2/3/4	Proc-3	
F-1	[1, 1]	[5/7/9/3, 2]	[2, 3]	Forward
F-2	[1, 1]	[9/8/9/10, 2]	[2, 3]	Forward
F-3	[1, 1]	[5/7/7/3, 2]	[2, 3]	Forward
F-4	[1, 1]	[9/8/7/10, 2]	[2, 3]	Forward
F-5	[1, 1]	[5/7/9/3, 2]	[2, 3]	Forward
R-1	[2, 3]	[5/7/9/3, 2]	[1, 1]	Reverse
R-2	[2, 3]	[9/8/9/10, 2]	[1, 1]	Reverse
R-3	[2, 3]	[5/7/9/3, 2]	[1, 1]	Reverse
R-4	[2, 3]	[9/8/9/10, 2]	[1, 1]	Reverse
R-5	[2, 3]	[5/7/9/3, 2]	[1, 1]	Reverse

Job information of 4 agents

Agent-Id	Start Time	Due Time	Makespan Price	Late Penalty
1	8:00	10:00	100	500
2	8:00	10:00	200	600
3	9:00	11:00	100	500
4	9:00	12:00	250	800

Machine capacity of resource pool

Machine type	2^{nd} transportation	3^{rd} post-processing
Capacity	16	8

Work load statistics for 4 agents

time. In practice, an "interface" time should be considered, i.e. in the container terminal operation, the prime mover must arrive at least slightly earlier than the crane carries the container to the lane. And in this model, the return time is not explicitly considered. The model simply assume that once a 2^{nd} machine finished its current job, it will be available immediately for the next job. However, it can be considered implicitly by enlarge the 2^{nd} machine operation time to twice as much, then it will cover the return time. Hence, a complete model to handle transportation related flow shop problem can be designed hierarchically as in Fig. 1:

- Stage-1 is a centralized problem for *Resource Allocation and PreScheduling*;
- Stage-2 is a decentralized problem for *Machine dispatching and Scheduling Adaptation*;

This paper will focus on the *Stage-1*.

The above model can be viewed as an extension to Bish's model [6], which considers only the vehicle dispatch problem by assuming infinite supply of yard cranes. It is similar with Decision problem-2 (D2) in [11], but is different in terms of the objective functions, which are to minimize the number of internal trucks in the yard and to maximize the utilization of internal trucks.

B. Literature review

Solutions for the classical job shop (flow shop) problems may be classified into 3 categories:

1) heuristics or branch-and-bound [1] [2] [3] [7] [8] and [17];
2) Lagrangian relaxation or auction approach [12] [13];
3) genetic algorithms [4] [9] [14] [16].

As for the Multi-Machine, Multi-Period problem studied in this paper, we contribute in the exact formulation and fast heuristic method benchmarked with standard solvers. Given that job

TABLE III
NOTATION FOR JOBS, PROCESSES MACHINES AND CONSTRAINTS

Symbol	Description
Notation for Jobs and Processes	
L	Total number of job-lists Total number of agents
N^l	Total number of jobs in job-list l $l \in \{1, 2, ..., L\}$
F	Total time frames (total number of time period)
o_i^l	Total number of operations for job i in job-list l
p_{ij}^l	Processing time of job i operation j in job-list l $i \in \{1, ..., N^l\}$ and $j \in \{1, ..., o_i^l\}$
D^l	Due time of job-list l
R^l	Release time of job-list l
W_d^l	Delay penalty of job list l
W_m^l	Makespan price per time of job list l
Notation for Machines	
K	Total number of machine types
k	Machine type index, $k \in \{1, ..., K\}$
$M_k(t)$	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 $k \in \{1, ..., K\}, t \in \{1, 2, ..., \tilde{T}\}$
$\mathcal{T}_f$	Partition of total time slots into F periods $1 \leq f \leq F$
$C_{k,\mathcal{T}_f}$	Capacity of machine k at time frame $\mathcal{T}_f$
$C_{k,\mathcal{T}_f}^M$	maximum capacity constraints of machine type k in time frame $\mathcal{T}_f$ for all agents
m_{ij}	Mapping from job i and operation j to machine type
Decision Variable	
X_{ijt}	Discrete decision variable for i^{th} job, j^{th} operation at time slot t
T_{ij}^s	Start time of operation j in job i
T_{ij}^e	Completion time of operation j in job i
T_{MS}	Makespan completion time of last critical operation

scheduling is hard computationally, it is even harder to consider resource allocation and operational scheduling jointly [5]. 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 paper gives an integrated model, based on Pritsker's 0-1 formulation [18].

II. Resource Allocation Problem Formulation-Centralized Approach Integer Programming(ResAlloc-IP)

TABLE III gives the notations in this paper. There are multiple job-lists $l \in \{1, 2, ..., L\}$, and all of them are sharing the same pool of resources. Assume each job-list is represented by an agent. And each job list has its own start time, due time, delay penalty, and makespan price. The problem is to allocate resources to each agent and then schedule each job-list with the allocated resources.

Before formulation, one has to estimate a proper time unit $U_T \in R$, with which the continuous time domain is discretized to $\tilde{T} \in N$ time slots. Different from scheduling problem is that, this total time period $U_T\tilde{T}$ must hold all the operation time of L job-lists. It is from the earliest starting time to the latest completion time. There is a partition of total time horizon $\{1, 2, ..., \tilde{T}\}$ into F time frames, $\{\mathcal{T}_f : 1 \leq f \leq F\}$, such that $\bigcup_{f\geq 1}^{F} \mathcal{T}_f = \{1, 2, ..., \tilde{T}\}$ and $\mathcal{T}_{f1} \bigcap \mathcal{T}_{f2} = \phi, \forall f1 \neq f2$.

We list scheduling problem formulation both in discrete time domain and in continuous time domain. The comparison is for readers' clearly understanding the actual meaning of the constraints, which is useful for the proposition. Note that the integrated formulation for resource allocation problem is actually based on and very similar to the scheduling problem, except that machine quota $C_{k,\mathcal{T}_f}$ becomes a decision variable here. The formulation in continuous time domain could also be done similarly, but here we only focus on the discrete time domain, which is actually implemented in the final simulation in our experiments. Operational precedence constraint is represented by f_{OPRECE}, where equality means that there is no-wait between consecutive operations. Machine capacity constraint is represented by inequality g_{MACAP}. COS constraint is represented by inequality g_{COS}.

A. 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}^l for job-list agent l, $l \in \{1, 2, ..., L\}$.

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

Machine quota allocated to job-list l for each type k at time frame $\mathcal{T}_f$.

$$C_{k,\mathcal{T}_f}^l \in \mathcal{N}, l \in \{1, 2, ..., L\}, k \in \{1, 2, ..., K\}, f \in \{1, 2, ..., F\}$$

B. Decision parameters

- Partition of total time horizon $\{1, 2, ..., \tilde{T}\}$ into F time frames, $\{\mathcal{T}_f : 1 \leq f \leq F\}$, all the agents observe the same time partition;
- $p_{i,j}^l, l \in \{1, 2, ..., L\}, i \in \{1, 2, ..., N\}, j \in \{1, 2, ..., o_i\}$: 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_d^l: Delay penalty of job-list l;
- W_m^l: Makespan price per time for job-list l;
- $C_{k,\mathcal{T}_f}^M$: Maximum capacity constraints of machine type k in time frame $\mathcal{T}_f$;

C. Integrated minimization model in discrete time domain

The model is to minimize the sum of the weighted makespan cost and tardiness penalty (15) under the constraints { (16), (17), (18), (19), (20), (21), (22), (23), (24) }.

Constraints { (16), (17), (18), (19) } are just extended from that in scheduling problem to a system with multiple job agents. Constraint (20) states that l^{th} job-list's 1^{st} job's 1^{st} operation cannot start until the job-list is released. Constraint (21) states

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

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

$$f_{OPRECE} = \left\{ \begin{array}{ll} X_{i,j,t} - X_{i,j-1,t-p_{i,j-1}} & \text{if } t > p_{i,j-1} \\ X_{i,j,t} & \text{if } t \leq p_{i,j-1} \end{array} \right\} = 0, \forall i, j \in \{2,...,o_i\}. \tag{3}$$

$$g_{MACAP}(X) = \sum_{i,j:m_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > p_{ij} & (X_{ijt} - X_{i,j,t-p_{ij}}) \\ \text{if } t \leq p_{ij} & X_{ijt} \end{array} \right\} - C_{k,\mathcal{T}_f} \leq 0, \forall t \in \mathcal{T}_f, \forall k \in \{1,2,...,K\} \tag{4}$$

$$g_{COS} = \left\{ \begin{array}{ll} X_{i,j_i^*,t} - X_{i-1,j_{i-1}^*,t-p_{\{i-1,j_{i-1}^*\}}} & \text{if } t > p_{\{i-1,j_{i-1}^*\}} \\ X_{i,j_i^*,t} & \text{if } t \leq p_{\{i-1,j_{i-1}^*\}} \end{array} \right\} \leq 0, \forall i. \tag{5}$$

$$X_{ijt} \in \{0,1\} \qquad \forall i,j,t \in \{1,2,...,\tilde{T}\}. \tag{6}$$

$$\text{minimize} \quad W_m T_{MS} + W_d \max\{0, T_{MS} - D\} \tag{7}$$

$$\text{subject to: } f_{PREEM} = T_{ij}^s - T_{ij}^e + p_{ij} = 0, \forall i, j. \tag{8}$$

$$f_{OPRECE} = T_{i,j-1}^e - T_{ij}^s = 0, \forall i, j \geq 2. \tag{9}$$

$$g_{MACAP}(t) = M_k(t) - C_{k,\mathcal{T}_f} \leq 0, \forall t \in \mathcal{T}_f, k \in \{1,2,...,K\} \tag{10}$$

$$g_{COS} = T_{i,j_i^*}^e - T_{i+1,j_{i+1}^*}^s \leq 0, \forall i \in \{1,2,...,N-1\}. \tag{11}$$

$$T_{i,j_i^*}^e - T_{MS} \leq 0, \forall i \tag{12}$$

$$T_{ij}^s \geq 0 \tag{13}$$

$$T_{ij}^e \geq 0 \tag{14}$$

$$\textbf{ResAlloc-IP:} \text{ minimize} \sum_{l\geq 1}^{L} \left\{ W_d^l \cdot \sum_{t>D^l - p_{N,o_N}^l} (1 - X_{N,o_N,t}^l) + W_m^l \cdot \sum_t (1 - X_{N,o_N,t}^l) \right\} \tag{15}$$

$$\text{subject to: } X_{ijt}^l - X_{i,j,t+1}^l \leq 0, \qquad \forall l, i, j, t \in \{1,2,...,\tilde{T}-1\}. \tag{16}$$

$$f_{OPRECE}^l = \left\{ \begin{array}{ll} X_{i,j,t}^l - X_{i,j-1,t-p_{i,j-1}}^l & \text{if } t > p_{i,j-1}^l \\ X_{i,j,t}^l & \text{if } t \leq p_{i,j-1}^l \end{array} \right\} = 0, \forall l, i, j \in \{2,...,o_i\}. \tag{17}$$

$$g_{MACAP}^l(X) = \sum_{i,j:m_{ij}=k} \left\{ \begin{array}{ll} \text{if } t > p_{ij}^l & (X_{ijt}^l - X_{i,j,t-p_{ij}}^l) \\ \text{if } t \leq p_{ij}^l & X_{ijt}^l \end{array} \right\} - C_{k,\mathcal{T}_f}^l \leq 0, \forall l, t \in \mathcal{T}_f, \forall k \in \{1,2,...,K\} \tag{18}$$

$$g_{COS}^l = \left\{ \begin{array}{ll} X_{i,j_i^*,t}^l - X_{i-1,j_{i-1}^*,t-p_{\{i-1,j_{i-1}^*\}}}^l & \text{if } t > p_{\{i-1,j_{i-1}^*\}}^l \\ X_{i,j_i^*,t}^l & \text{if } t \leq p_{\{i-1,j_{i-1}^*\}}^l \end{array} \right\} \leq 0, \forall i, l. \tag{19}$$

$$X_{11,R^l-1}^l = 0, \forall l. \tag{20}$$

$$X_{ij,\tilde{T}}^l = 1, \forall l, i, j. \tag{21}$$

$$\sum_{l\geq 1}^{L} C_{k,\mathcal{T}_f}^l \leq C_{k,\mathcal{T}_f}^M, \forall \mathcal{T}_f, k. \tag{22}$$

$$X_{ijt}^l \in \{0,1\} \qquad \forall l, i, j, t \in \{1,2,...,\tilde{T}\}. \tag{23}$$

$$C_{k,\mathcal{T}_f}^l \in \mathcal{N}, \qquad \forall k, \mathcal{T}_f \tag{24}$$

that all jobs must be finished at the end. Constraint (22) states the sum of machine quotas by all job-lists should be within the capacity limit. This constraint links all L scheduling problems together. So by relaxing this constraints, the problem could be decentralized.

III. Comparison with Continuous Time Domain Formulation

- T_{ij}^s: start time of operation j in job i;
- T_{ij}^e: completion time of operation j in job i;
- T_{MS}, Makespan: completion time of last critical operation.

In continuous time domain, the decision variables are just $\{T_{ij}^s, T_{ij}^e : i, j\}$ and T_{MS}. Although the formulation cannot be solved by any available solvers, a feasible solution can be given by the heuristics. Especially, the construction of machine utilization is useful in decentralized approach [13] to solve the resource allocation problems when different agents have different time slot units. *Machine Capacity* constraint is represented by Inequality (10) with the δ function.

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

where in continuous time domain $t \in R$, $\delta(t-T)$ is a positive

infinite pulse at time point defined by parameter T:
$\delta(t-T)=\begin{cases} 0 & \forall t\in(-\infty,T)\cup(T,\infty) \\ \infty & \text{if } t=T \end{cases}$
And the energy or area of the function $\delta(t-T)$ with axis t is unit step up at time point $t=T$.
$\int_{-\infty}^{t}\delta(t-T)\,dt=\begin{cases} 1 & \text{if } t>T, \\ 0 & \text{if } t<T. \end{cases}$

The above 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, then scheduling problem can be solved by the heuristic methods.

It is clear that $g_{MACAP}(t)$ in continuous time formulation is replaced with $g_{MACAP}(X)$ in discrete time domain. From (27) to (30), we can see that a set of scheduling $\left\{(T_{ij}^s,T_{ij}^e)|i=1,2,...,N;j=1,2,...,o_i\right\}$ will be the parameter of function $g_{MACAP}(t)$, while in discrete time domain $\left\{X_{ijt}\in\{0,1\}|i=1,2,...,N;j=1,2,...,o_i;t=1,2,...,\tilde{T}\right\}$ with constraints by (2) is corresponding to a set of scheduling.

Critical Operation Sequencing is formulated by Inequality (11), noting j_i^* is a critical operation for each job and it has strict precedence dependency among jobs.

IV. Implementation of Machine Utilization Function

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.

A. Implementation in continuous time domain

The above formulation makes use of the $\delta(t)$ function, which is only available in close form formulas. It is equivalent to the following window functions.

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

In terms of implementation, a look-up table is generated given a set of schedule $\{(T_{ij}^s,T_{ij}^e)|i=1,...,N;j=1,...,o_i\}$. 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 $\mathcal{T}_k^{se}$ is constructed which contains all such cell matrix related with $m_{ij}=k$.

$$\mathcal{T}_k^{se}=\left\{\left[\begin{pmatrix} T_{ij}^s \\ 1 \end{pmatrix},\begin{pmatrix} T_{ij}^e \\ -1 \end{pmatrix}\right]:\right\} \quad 1\le i\le N, 1\le j\le o_i \text{ and } m_{ij}=k \tag{25}$$

The above 2-by-2 matrix can also be viewed as a 2-by-2 mapping, from time to pulse. The start time T_{ij}^s maps to a unit positive pulse 1, while the end time T_{ij}^e maps to a unit negative pulse -1.
Considering all the machine types $k\in\{1,2,...,K\}$, the total number of above cell matrices is $\sum_{i\ge1}^{i\le N}o_i$.
Specially for flow shop, there is $o_i=K,\forall 1\le i\le N$, 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, we form a mapping matrix of 2 by N 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.

$$\mathcal{T}_k=\left[\begin{array}{cccc} \mathcal{T}_k^{se}(1) & \mathcal{T}_k^{se}(2) & \cdots & \mathcal{T}_k^{se}(N) \end{array}\right] \tag{26}$$

Note that $\mathcal{T}_k$ 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:

$$[T_k^{ord},Index] = sort(\mathcal{T}_k(1,:)) \tag{27}$$
$$\mathcal{T}_k^{ord}(1,:) = T_k^{ord}. \tag{28}$$
$$\mathcal{T}_k^{ord}(2,:) = \mathcal{T}_k(2,Index) \tag{29}$$

Then,

$$M_k(t)=\sum_{j:\mathcal{T}_k^{ord}(1,j)\le t}\mathcal{T}_k^{ord}(2,j) \tag{30}$$

Hence, (25) to (30) give the complete implementation of machine utilization function in continuous time domain. For online programming where numerical error might happens for improper rounding, an infinitesimal value ϵ could be added in (25), which will be replace by (31).

$$\mathcal{T}_k^{se}=\left\{\left[\begin{pmatrix} T_{ij}^s+\epsilon \\ 1 \end{pmatrix},\begin{pmatrix} T_{ij}^e \\ -1 \end{pmatrix}\right]:\right\} \quad 1\le i\le N, 1\le j\le o_i \text{ and } m_{ij}=k \tag{31}$$

B. Implementation of machine 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). The machine utilization function could be directly constructed by the first part of (4)

$$M_{kt}(X)=\sum_{i,j:m_{ij}=k}\left\{\begin{array}{ll} \text{if } t>p_{ij} & (X_{ijt}-X_{i,j,t-p_{ij}}) \\ \text{if } t\le p_{ij} & X_{ijt} \end{array}\right\}, \quad \forall k\in\{1,2,...,K\} \tag{32}$$

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

$$T_{ij}^s = t^*-1 \Leftrightarrow X_{ij,t^*}=1 \text{ and } X_{ij,t^*-1}=0 \tag{33}$$
$$T_{ij}^e = T_{ij}^s+p_{ij} \tag{34}$$

Then equations {(31), (26), (27), (28), (29), (30)} 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.

C. Solution equivalence between discrete time formulation and continuous time formulation

Proposition: The solution to Continuous Time Formulation is equivalent to the Discrete Time Formulation, under the following conditions:

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

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

$$\begin{array}{ccc} X_{i,j,t^*-1} == 0 & \cap & X_{i,j,t^*} == 1 \\ & \Updownarrow & \\ T^s_{ij} = t^* - 1, & & i^{th} \text{ job's } j^{th} \text{ operation starts at time slot } t^*. \end{array}$$

According to Eqn. (2) 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

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

Hence, the 1^{st} term $W_m T_{MS}$ in continuous time domain objective function (7) is related with the first term of discrete time domain objective function (1) 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 2^{nd} term $W_d max\{0, T_{MS} - D\}$ in continuous time domain objective function (7) is related with the 2^{nd} term of discrete time domain objective function (1) 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 (1) 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 (1) is equivalent to minimize function (7). ■

V. Scheduling Heuristic Methods Based on Machine Capacity Constraint Relaxation

The basic concept of the proposed heuristic methods are very much similar to those in [7], while ours are different in following aspects:

- There is no-wait within consecutive tasks of a job;
- Our heuristic methods are suitable for both continuous time (i.e. $p_{ij} \in \mathcal{R}^+$) and discrete time problems (i.e. $p_{ij} \in \mathcal{Z}^+$);
- Ours are suitable for both constant machine capacity and period-dependant capacity.

A. 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 {(8), (9), (11), (12), (13), (14) } in continuous time domain, or constraints {(1), (2),(3), (5), (6) } in discrete time domain.

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

CH3: *Construct machine utilization function* according to equations {(31), (26), (27), (28), (29), (30) } in continuous time domain.

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

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 {(31), (26), (27), (28), (29), (30)} in continuous time domain.

CH4: *loop*

B. Repair heuristic method(RH)

RH1: Construct an initial schedule with infinite resource capacity (i.e. considering only the constraints {(8), (9), (11), (12), (13), (14) } in continuous time domain, or constraints {(2), (3), (5), (6)} 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 {(31), (26), (27), (28), (29), (30)} in continuous time domain.

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

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

RH2-b2: $G = M_{kt} - C_{k,\mathcal{T}_f}$;

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 following jobs according to the constraints {(8), (9), (11)};

RH2-b6: Update T_{MS};

RH2-b7: Construct the *machine utilization function* according to equations {(31), (26), (27), (28), (29), (30) } 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 IV-B). Heuristic RH is usually used in Lagrangian relaxation approach to repair the feasibility and get an upper bound estimation of makespan [20].

VI. Resource Allocation Solution by Linear Programming Relaxation

When the problem size grows larger, solving the Integer Problem becomes intractable, even with commercial solvers. In this section, we present an approach based on the combination of Linear Programming Relaxation (**LPR**) and scheduling heuristic (CH), which yields an average solution quality within 120% optimality in less than 2% of run time compared with running ResAlloc-IP model on the CPLEX solver.

A. Formulation of linear programming relaxation

The LPR formulation is simply to relax the integer constraints thereby resulting in a linear program. The formulation is as follows.

ResAlloc-LPR

minimize	(15)
subject to:	Inequality(16)
	Equality(17)
	Inequality(18)
	Inequality(19)
	Equality(20)
	Equality(21)
	Inequality(22)
	$0 \leq X_{ijt}^{l} \leq 1, \forall l, i, j, t \in \{1, 2, ..., \tilde{T}\}.$
	$C_{k,\mathcal{T}_f}^{l} \geq 1, \forall k, \mathcal{T}_f.$

With sparse matrix technique and interior point methods, the linear problem could be solved in polynomial time, i.e. much faster than the integer problem (**ResAlloc-IP**).

The complete solution procedure (*Heuristic LPR-Rounding*) is described as follows:

B. Heuristic LPR-Rounding

1) Formulating (**ResAlloc-LPR**) and solve it.
2) Get the solution of $C_{k,\mathcal{T}_f}^{l}$, which should be floating real numbers. And round them to the nearest integer.
3) Calculate the overall machine usage by $\sum_{l\geq 1}^{L} C_{k,\mathcal{T}_f}^{l}$.
4) *for* every machine type $k \in \{1, ..., K\}$ and every time frame $\mathcal{T}_f : 1 \leq f \leq F$
5) *if* there is any violation in machine type k, calculate this excess demand $ED_{k,\mathcal{T}_f} = \sum_{l\geq 1}^{L} C_{k,\mathcal{T}_f}^{l} - C_{k,\mathcal{T}_f}^{M}$.
   a) Calculate the priority of all the job agents at time frame $\mathcal{T}_f$, according to [19].
   b) find the job agent with the lowest priority and simply reduce its demand by $ED_{k,\mathcal{T}_f}$
6) *end if*
7) *loop for*
8) With the feasible allocation solution $\{C_{k,\mathcal{T}_f}^{l} : 1 \leq k \leq K, 1 \leq f \leq F\}$, solve the scheduling problem. It can be done either by solving a smaller ScheGen-IP problem instance or by applying a fast heuristic method (which could be either CH or RH in Section V).

The LPR-Rounding heuristic method serves to decompose the overall problem into subproblems to be solved with one scheduling heuristic methods. If the underlying scheduling heuristic method is CH (resp. RH), we name the overall algorithm as *LPR-Rounding-CH* (resp. *LPR-Rounding-RH*); if the scheduling heuristic method is greedy algorithm, [6], we name it *LPR-Rounding-Greedy*.

VII. Experimental Results

We first provide a detailed comparison of our approach against existing approaches on the sample problem provided in Section I. Then, we present comprehensive experimental results for both the scheduling problem as well as the integrated resource allocation and scheduling problem. The benchmark is done both in run time and solution quality. All the experiments were performed on a Pentium IV CPU with 3GHz and 1GByte memory. We chose CPLEX 10.0 for solving IP problems and MOSEK (from http://www.mosek.com) for linear problems.

A. Comparison on sample problem

On the sample problem in Section I, one period is one hour. We compare the ResAlloc-IP model, Auction Approach [13] (a distributed algorithm based on Lagrangian Relaxation) and the LPR-Rounding-CH approach. The comparison is done on run time, total cost and average makespan for 4 agents and the result is in TABLE IV. We can see that LPR-Rounding-CH is comparatively nearer to optimal and the run time is fastest of all approaches.

Fig. 2 gives the operation scheduling for ResAlloc-IP solution. Lower part of TABLE IV presents the resource allocation for each agents for ResAlloc-IP solution.

B. Comparison of scheduling solutions, heuristic methods vs ScheGen-IP

Next, we present experiments' result to compare the makespan derived from the heuristic methods *CH*, *RH*, Greedy algorithm in [6] and ScheGen-IP model. The common setting is as following:

TABLE IV
SOLUTION COMPARISON FOR SAMPLE PROBLEM IN TABLE II

Total cost and solution time comparison

	Cost per agent				Total cost	Solution time (sec)
	1	2	3	4		
ResAlloc-IP	175.0	266.7	450.0	750.0	1641.7	203.4
Auction Approach	500.0	800.0	450.0	604.2	2354.2	69.6
LPR-Rounding-CH	450.0	266.7	450.0	729.2	1895.8	6.2

Makespan comparison

	Makespan per agent (hour)				Average makespan
	1	2	3	4	
ResAlloc-IP	1.8	1.3	2.4	3.0	2.1
Auction Approach	2.5	2.5	2.4	2.4	2.4
LPR-Rounding-CH	2.4	1.3	2.4	2.9	2.3

Solution by ResAlloc-IP

Agent-Id	Bid (Machine 2, Machine 3)		
	1^{st} Period	2^{nd} Period	3^{rd} Period
	8:00 - 9:00	9:00 - 10:00	10:00-11:00
1	(6, 2)	(2, 2)	(0, 0)
2	(9, 2)	(0, 2)	(0, 0)
3	(1, 2)	(9, 2)	(0, 3)
4	(0, 0)	(4, 2)	(5, 2)
subtotal	(16, 6)	(15, 8)	(5, 5)

Solution by LPR-Rounding-CH

Agent-Id	Bid (Machine 2, Machine 3)		
	1^{st} Period	2^{nd} Period	3^{rd} Period
	8:00 - 9:00	9:00 - 10:00	10:00-11:00
1	(4, 2)	(3, 2)	(1, 0)
2	(9, 2)	(0, 2)	(0, 0)
3	(2, 2)	(8, 2)	(1, 2)
4	(0, 0)	(4, 2)	(3, 1)
subtotal	(15, 6)	(15, 8)	(5, 3)

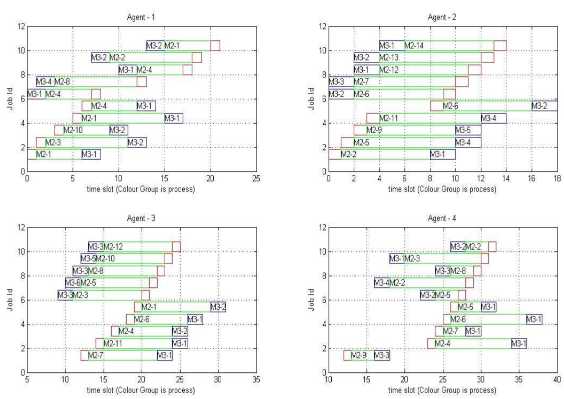

Fig. 2. MIP Solution for Sample Problem – agent's scheduling

- 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;
- 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. 3-(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*

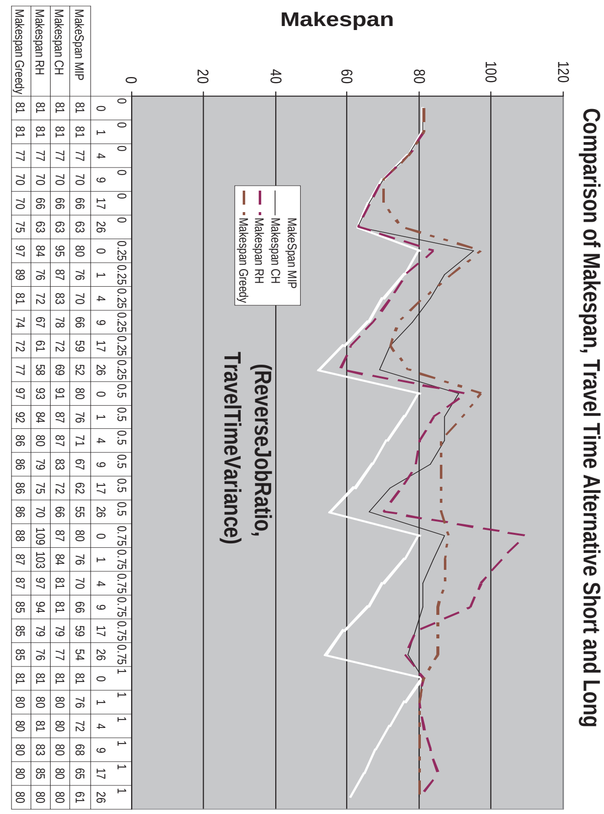

Fig. 3. Comparison on 25 job lists between heuristic methods and ScheGen-IP model

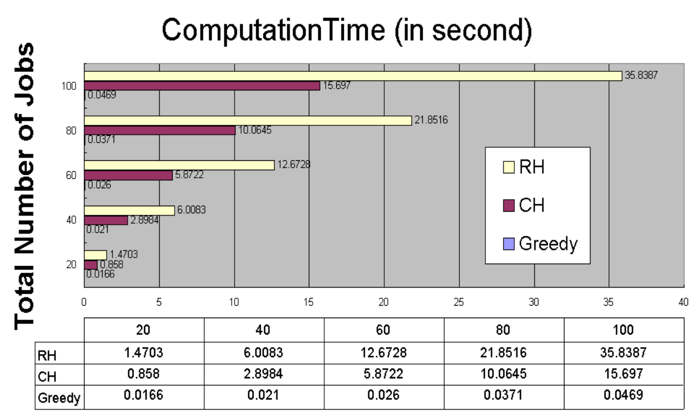

Fig. 4. Solution time comparison between scheduling heuristics and ScheGen-IP model

Flow Jobs with small variance in processing time, 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. VII-B. 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.

TABLE V
PROBLEM SETTING AND SOLUTION COMPARISONS FOR FIRST 5 GROUPS OF EXPERIMENTS

Grp No.	Problem Settings						Solution Comparison			
	For	Rev	Total mach. type	Processing time on machine 1st - 2nd - 3rd - 4th - 5th	Machine capacity 1st - 2nd - 3rd - 4th - 5th	Job Sequence	Average Optimality v.s. CPLEX (Obj. Value) v.s. CPLEX in 1 hour: LPR -Grd.	LPR -CH	Obj.	S.able p.cent
									CPLEX Comp. Time	
1	10	10	3	1 - Var - 2 - NA - NA	4 - 24 - 16 - NA - NA	For → Rev	139.8%	118.5%	114.3%	50.0 %
2	20	0	3	1 - Var - 2 - NA - NA	4 - 24 - 16 - NA - NA	For → Rev	114%	108%	107.0%	100.0%
3	10	10	3	[1,2] - Var - [1,5] - NA - NA	4 - 24 - 16 - NA - NA	For → Rev	137%	128%	106.0%	60.0 %
4	10	10	5	[1,2] - Var - [1,5] - [6,9] - [5,8]	4 - 24 - 16 - 16 - 16	For → Rev	199%	165%	101.4%	50.0 %
5	10	10	4	[1,2] - Var - [1,5] - NA - NA	4 - 24 - 16 - NA - NA	Mixed	600%	118%	103.8%	100.0%

C. Comparison of integrated allocation and scheduling solutions

Final is the experiments' result on the integrated allocation and scheduling problem. We performed 6 groups of experiments. The first five groups focus on the optimality comparison with our IP model, and the last one focuses on run time comparison between LPR-Rounding-CH and an Auction Approach proposed in [13], since the IP model is unable to return solutions due to the large-scale nature of the problem instances.

The design of our experiments is similar to that in [9], while details are shown in TABLE. V.

The first 5 groups of experiments focus on comparison of optimality. Precisely, the common setting among these groups is listed as follows, while the differences are shown in TABLE. V

- Each group consists 10 problem instance.
- In each problem instance, there are 4 agents;
- One period is equal to 40 TimeSlots;
- All 4 agents start their jobs at the same time;
- Tardiness penalty and makespan price are the same for every agent, makespan price is 100 per period, tardiness penalty is 500 per period.

In TABLE. V, the notation $[a, b]$ means uniform distribution between a and b, while *Var* means a variable processing time under uniform distribution of some range. The ranges are different for different cases within each group of experiments. *For* → *Rev* means forward jobs are always proceeding reverse jobs in one job-list. A summary of our results is shown at *Solution Comparison* columns in TABLE. V.

Further notes about this table is following:

- Solution comparison is done between LPR-Rounding-CH and LPR-Rounding-Greedy. The percentage is their solution objective value divided by ResAlloc-IP solution objective value and ResAlloc-IP is solved by CPLEX. Because the problem is large, we just obtain the best feasible solution within 1 hour.
- We tried the LP solver by both MOSEK and SEDUMI. Although MOSEK is around two times faster than SEDUMI (solved within 2 minutes), their results are almost the same.
- The problems with 3 machine , i.e. group-{1, 2, 3, 5}, can be solved by MOSEK in 1 minute. While 5 machine problems, i.e. group-4, need about 2 minutes.
- The table mainly shows the comparison between LPR-Rounding-CH and LPR-Rounding-Greedy [6].
- Besides comparing LPR-Rounding-CH and LPR-Rounding-Greedy, we further compare them with CPLEX solution within comparable times, i.e. 1 minute for group-{1, 2, 3, 5} or 2 minutes for group-4.

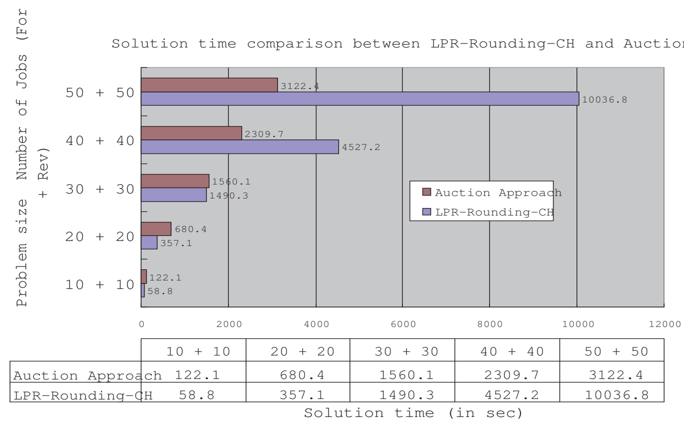

Fig. 5. Comparison on solution time between auction's approach and ResAlloc-IP Solution

From the five groups of experiments, we can see that solutions by LPR-Rounding-CH perform consistently better than that by LPR-Rounding-Greedy. Specially, the Greedy algorithm seems to be more sensitive to variation of the COS constraints, while CH heuristic method is less sensitive (or more robust).

We found that such problem *BiFSP* with 3 machine and 10 forward and 10 reverse jobs is just, sort of, solvable boundary for CPLEX. Which means, some can be solved while some cannot. Further, CPLEX just cannot give a solution for 20 forward and 20 reverse jobs.

For the 6th group of experiments, we focus on solution time comparison between LPR-Rounding-CH and an Auction Approach on large-scale problems. We are unable to use CPLEX to solve them when each agent has such problem instances as more than 20 forward jobs and 20 reverse jobs. The resulting IP model *ResAlloc-IP* 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, we compare both the solution quality and run time between LPR-Rounding-CH and an auction approach, which return feasible solutions within 20 minutes.

Five problem instances have been generated, the common setting is similar as previous five groups, while the difference among 5 instances is the total number of jobs in the job-lists. 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. 5. From the results, we see that 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.

VIII. Conclusion and Further Research Directions

This paper offers a new perspective to a new variant of the BiFSP with multiple machine capacity and COS constraints. We rather benchmark the proposed solution approach against existing approaches (IP and Greedy) than declare any best solutions. Continuous time domain formulation is more useful in resource allocation scenario than in schedule generation, for different agents may have different time slot units. The theoretical meaning for COS constraints is that optimal sequencing may not lead to global optimal scheduling.

IX. Acknowledgement

We appreciate the reviewers and the editors' suggestions, which greatly help to improve our work.

References

[1] A. 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.
[2] J. Adams, E. Balas, D.Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34, pp. 391-401, 1988.
[3] M. Bartusch, R.H. Mohring and F.J. Radermacher, Scheduling project networks with resource constraints and time windows, in *Annals of Operations Research*, Vol.16, pp.-201-240, 1988
[4] B.B. Li and L. Wang, A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling, in *IEEE Trans. Syst., Man, Cybern. B*, Vol.37, No.3, pp. 576-691, 2007.
[5] E. K. Bish, A multiple-crane-constrained scheduling problem in a container terminal, *European Journal of Operational Research* Vol.144, pp. 83-107, 2003.
[6] E. K. Bish et al., Dispatching vehicles in a mega container terminal, OR Spectrum, Vol.27, No.4, pp. 491-506, 2005.
[7] P. Brucker , Scheduling and constraint propagation, in *Discrete Applied Mathematics*, Vol.123, pp. 227-256, 2002
[8] S. David Wu, 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.
[9] I. Kacem, 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.
[10] J.R. Jackson, An Extension of Johnson's Results on Job Lot Scheduling, *Naval Research Logistics Quarterly*, 3, pp201-203, 1956.
[11] K.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.
[12] E. Kutanoglu, S. David Wu, Improving Scheduling Robustness via Preprocessing and Dynamic Adaptation, in *IIE Transactions*, vol.36, pp.1107-1124, 2004.
[13] H. C. Lau et al., Multi-period combinatorial auction mechanism for distributed resource allocation and scheduling, in *Proceedings of IAT2007, IEEE/ACM/WIC International Conference on Intelligent Agent Technology*, 2007, pp. 407-411.
[14] J. Liu, 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.
[15] M. Drozdowski, Scheduling multiprocessor tasks - An overview, *European Journal of Operational Research*, Vol.94, pp 215-230, 1996.
[16] N. 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.
[17] B.Peter, Scheduling and constraint propagation, *Discrete Applied Mathematics*, Vol.123, pp. 227-256, 2002.
[18] A. Pritsker, 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.
[19] A.P.J. Vepsalainen, and T.E. Morton, Priority rules for job shops with weighted tardiness costs, in *Management Science*, Vol.33, No.8, 1035 - 1047.
[20] Z. J. Zhao et al., Bidirectional flow shop scheduling with multi-machine capacity and critical operation sequencing, in *Proceedings ISIC'2007, International Symposium on Intelligent Control*, 2007, pp.446-451.

ZhengYi John ZHAO (S97-ME01-MS02)BEng, MEng from Tsinghua China, MS from SMA-HPCES Singapore. He has 4 years working experience for motion control scheduling and production planning in ASM-PT(*Advanced Semiconductor Material Pacific Tech.*), R&D Dept. in Singapore. He has 2 years working experience for port-transportation scheduling in Singapore Management University as a research engineer. He is now a Ph.D. Candidate at Dept. of Electrical & Computer Engineering, NUS(*National University of Singapore*). His research interest includes control, automation and scheduling.

Hoong Chuin LAU (DEng, MSc, BSc) is Associate Professor of Information Systems at the Singapore Management University. He holds a concurrent appointment as Director of Defense Logistics at The Logistics Institute Asia Pacific, National University of Singapore. His research interests are situated at the intersection of artificial intelligence and operations research, with application to decision support and optimization in large-scale transportation, logistics and supply chain management. He has published more than 80 research papers in journals and international conferences and was awarded the Lee Kwan Yew research fellowship in 2008. He serves actively in program committees of international conferences, including the IEEE/WIC/ACM International Conference on Intelligent Agents Technology, and International Conference on Planning and Scheduling. His research contributions have also turned into innovative tools and systems for the logistics industry, which have been deployed in the Singapore Ministry of Defense, PSA, and Land Transport Authority. He was awarded the National Innovation and Quality Circles Star Award in 2006.

Shuzhi Sam GE (S90-M92-SM99-F06), PhD, DIC, BSc, PEng, is founding Director of Social Robotics Lab of Interactive Digital Media Institute, and Professor of the Department of Electrical and Computer Engineering, the National University of Singapore. He has (co)-authored three books: Adaptive Neural Network Control of Robotic Manipulators (World Scientific, 1998), Stable Adaptive Neural Network Control (Kluwer, 2001) and Switched Linear Systems: Control and Design (Springer-Verlag, 2005), edited a book: Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications (Taylor and Francis, 2006), and over 300 international journal and conference papers. He is the founding Editor-in-Chief, International Journal of Social Robotics, Springer. He has served/been serving as an Associate Editor for a number of flagship journals including the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, the IEEE TRANSACTIONS ON NEURAL NETWORKS and Automatica. He also serves as a book Editor of the Taylor & Francis Automation and Control Engineering Series. His current research interests include social robotics, multimedia fusion, adaptive control, intelligent systems and artificial intelligence.