10-2011

Allocating Resources in Multiagent Flowshops with Adaptive Auctions

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

Zhengyi ZHAO
Singapore Management University

Sam Shuzhi Ge
National University of Singapore

Thong Heng LEE
National University of Singapore

Citation

LAU, Hoong Chuin; ZHAO, Zhengyi; Ge, Sam Shuzhi; and LEE, Thong Heng. Allocating Resources in Multiagent Flowshops with Adaptive Auctions. (2011). *IEEE Transactions on Automation Science and Engineering*. 8, (1), 732-743.
Available at: https://ink.library.smu.edu.sg/sis_research/1374

Allocating Resources in Multiagent Flowshops With Adaptive Auctions

Hoong Chuin Lau, Zhengyi John Zhao, Shuzhi Sam Ge, *Fellow, IEEE*, and Tong Heng Lee, *Member, IEEE*

***Abstract*—In this paper, we consider the problem of allocating machine resources among multiple agents, each of which is responsible to solve a flowshop scheduling problem. We present an iterated combinatorial auction mechanism in which bid generation is performed within each agent, while a price adjustment procedure is performed by a centralized auctioneer. While this approach is fairly well-studied in the literature, our primary innovation is in an adaptive price adjustment procedure, utilizing variable step-size inspired by adaptive PID-control theory coupled with utility pricing inspired by classical microeconomics. We compare with the conventional price adjustment scheme proposed in Fisher (1985), and show better convergence properties. Our secondary contribution is in a fast bid-generation procedure executed by the agents based on local search. Putting both these innovations together, we compare our approach against a classical integer programming model as well as conventional price adjustment schemes, and show drastic run time improvement with insignificant loss of global optimality.**

***Notes to Practitioners*—Decentralization and competition are major emerging themes in resource planning and scheduling. A supply chain, for example, is inherently decentralized, where decision makers are drawn from different companies which may compete with one another. Hence, a mechanism needs to be designed to ensure proper coordination among decision makers such that the overall system performance would be optimized. In this paper, we propose an enhanced iterative combinatorial auction approach based on general equilibrium to tackle a decentralized multimachine flow-shop scheduling problem. The major challenge of this work is in attaining price convergence quickly thereby achieving computational efficiency. To this end, we propose two ideas for rapid price convergence—through the use of utility pricing (from micro-economics) and variable step size (from control theory). We show that we can solve large-scale decentralized flow-shop problems with drastic improvement in computational performance with little compromise on solution quality.**

***Index Terms*—Auction, decentralized decision making, flow shop, resource allocation.**

Manuscript received January 30, 2010; revised November 08, 2010; accepted March 20, 2011. This paper was recommended for publication by Associate Editor A. Kalir and Editor Y. Narahari upon evaluation of the reviewers' comments. This work was supported in part by the A*STAR SERC TSRP under Grant P0520101 and Grant P0520104. A preliminary version of this paper was presented at the International Conference on Electronic Commerce, Innsbruck, Austria, 2008.

H. C. Lau is with the School of Information Systems, Singapore Management University, Singapore 178902, Singapore (e-mail: hclau@smu.edu.sg).

Z. J. Zhao, S. S. Ge, and T. H. Lee are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077, Singapore.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2011.2160536

I. Introduction

MARKET mechanisms, which have their roots in economics, have emerged as a powerful computational paradigm for multiagent decision making, particularly in a competitive environment. In e-Commerce, they have been deployed successfully in procurement (e.g., [3]) and sponsored search (e.g., [12]). In resource planning and scheduling, we witness the use of auctions in a variety of domains. For instance, in electricity markets, a pricing model is designed by Toczywski and Zoltowska [26] 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. Confessore *et al.* [5] consider a decentralized multiproject scheduling problem, where different projects are managed by local decision makers and a coordination mechanism is proposed to resolve shared resource allocation conflicts between different projects.

Despite the success of using markets in various applications, a number of inefficiencies might occur when markets are used in allocating resources. One major source of inefficiencies is the complementarity among resources, i.e., when multiple independently allocated resources are required in order to accomplish a single task, an agent might end up with only part of the required resources, and efficiency is lost as a result. This issue can be directly addressed by running a combinatorial auction. [6] provides a comprehensive treatment on the subject of combinatorial auctions where a number of other resource allocation applications (such as airspace system resources, truckload transportation, bus routes) have been presented.

In this paper, we consider the problem of multiagent resource allocation where agents seek to optimize their respective optimization goals by contending for resources from a common pool. More specifically, we are concerned with the following problem scenario:

1) there is a central pool of limited renewable resources that comprises multiple identical units of different machine types;
2) there are multiple automated software agents, each having a job list and is responsible to service jobs in its list by solving its respective scheduling problem; in this paper, we assume that each agent solves a generalized flowshop problem where each operation in a job is served by one unit of a distinct machine type.

The above problem scenario occurs in a variety of applications, such as container terminal operations (loading and discharge of containers), forward and reverse logistics (pickup and delivery over multiple transport modes), and (obviously) manufacturing. Where we differ from classical optimization prob-

lems is that there are multiple decision makers contending for resources for a common pool, and these agents are self-interested. They bid for the right to utilize certain combination of machine resources from the pool over certain time periods. In particular, we make use of a multiround combinatorial auction mechanism that is built on a general equilibrium framework [4], where the prices of machine resources over different periods will iterate over multiple rounds of the bidding process until a price equilibrium is achieved.

Our goal in this paper is not to propose a new combinatorial auction mechanism for resource allocation, as we witness abundance of contributions there (e.g., see the volume dedicated to Combinatorial Auction in [6]). Rather, our interest is to propose methods to boost the computational performance of such combinatorial auction mechanism that achieves global optimality (defined as the sum of agent objectives) as far as possible. More precisely, our contribution is as follows. First and foremost, we propose the a novel price adjustment procedure that addresses the convergence issue faced by conventional price adjustment schemes. This is done through an innovative combination of utility price (concept from micro-economics) and variable step size adjustment (concept from control theory). Second, we propose an efficient local search scheme for bid generation. Put together, our auction scheme is capable of producing very good solutions on large-scale problem instances with significantly improved runtime performance over conventional auction schemes for decentralized scheduling proposed in the literature, such as [18].

This paper proceeds as follows. We first position our contribution in the light of a brief surveyed literature. We then introduce our case problem and provide an overview of our proposed auction protocol. This is followed by detailed technical descriptions of the proposed bid generation and price adjustment strategies. An experimental study follows, and we conclude with some remarks.

II. Preliminaries

General equilibrium auction mechanisms have been widely used for coordination resource utilization among agents over multiple periods. Wellman *et al.* [29], for example, introduced auctions where prices derived through distributed bidding protocols are used to determine job schedules, and investigated the existence of equilibrium prices for some general classes of scheduling problems. Other works such as Gagliano *et al.* [10] and Kutanoglu and Wu [18] proposed similar ideas 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 adjusting price based on excess demands. Tatonnement has its roots in computational economics [28]. One major challenge in the computational efficiency of tatonnement and tatonnement-based auction mechanism is on the speed of convergence. The computational efficiency issue surrounding auctions has been raised (in [2], [4], [16]), and in particular, Fisher [8] proposed price adjustment process based on the subgradient search method.

To our knowledge, there is rich research opportunity in boosting computational performance through better price adjustment strategies in a tatonnement process. In much of the research works listed above, prices are often determined primarily and simplistically by the supply-demand gap, or by the profit gap between buyer and seller. In this paper, we carefully study how the standard scheme can be augmented in two ways. First, we propose that bidders be given the opportunity to also reveal to the auctioneer their *marginal utility* values, and together with the market (i.e., aggregate demand and supply), we can achieve faster convergence and better quality solutions. Intuitively, we are able to achieve fast price equilibrium since 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. We will assume that agents truthfully report the marginal utility values. We propose a price adjustment strategy called price adjustment with utility pricing (termed as *PA-U*), which we compare with the conventional price adjustment strategy based on bid prices alone (*PA-B*). Second, we propose an adaptive scheme where the step size is adjusted from one iteration to the next, which 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 we will call an "adaptive auction."

With the improved price adjustment strategies as well as improved search algorithms proposed in this paper, we are able to tackle far larger and more complex scheduling instances compared with known experimental results. As an illustration, Kutanoglu and Wu [18] dealt with $3 * 3$ flowshop (as well as other small jobshop) instances, while we experiment with four agents each solving a $20 * 3$ generalized flowshop problem requesting resources from a pool of 44 machines.

A. Problem Definition

We are concerned with the problem of allocating machine resources from a common resource pool to multiple agents using an auction approach. To guarantee feasibility, each agent is first endowed with a baseline resource allocation. Each agent needs to solve a generalized flowshop problem [30], where jobs comprise multiple operations requiring distinct machine types. The goal is to seek an allocation that minimizes the global objective function, which is the sum of the agent objectives.

Table I contains details for notations used in this paper.

All agents compete for the utilization of resources from a common pool of K different types of machines available over $\mathcal{T}$ periods, and the resource capacity is denoted as $S_{k\tau} : k = 1, \cdots K, \tau = 1, \cdots, \mathcal{T}$. Each agent objective is to minimize the weighted sum of total makespan and tardiness cost (abbrev. makespan-tardiness cost or MTC).

Being an auction approach, each machine at each period will be priced. Let $p^r_{k\tau}$ denote the bid price of for machine k at period τ in auction iteration r. According to this price vector, each agent will submit a resource bid denoted $\mathbf{U}^l$. Let $\mathcal{M}(\mathbf{U}^l)$ denote the resulting makespan, and tardiness is defined as $\max\{0, \mathcal{M}(\mathbf{U}^l) + R^l - D^l\}$.[1] The makespan-tardiness cost, denoted MTC^l is computed as $W^l_m \cdot \mathcal{M}(\mathbf{U}^l) + W^l_d \cdot \max\{\mathcal{M}(\mathbf{U}^l) + R^l - D^l, 0\}$. The

[1]Note that we measure makespan with respect to the release time. In other literature, this is called flow-time.

TABLE I
LIST OF NOTATIONS

Symbol	Description
Agents	
L	Total number of agents (equiv. 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_m^l	Makespan penalty coefficient for job list l
W_d^l	Tardiness penalty coefficient for job list l
MTC^l	Makespan-Tardiness Cost of agent l defined as sum of weighted makespan and tardiness costs
$\mathcal{RC}^l$	Resource cost incurred by agent l
$\mathcal{SC}^l$	Sum Cost for agent l = $MTC^l + RC^l$
Resources	
K	Total number of machine types
k	Machine type index, $k \in \{1, ..., K\}$
τ	Time period index, $\tau \in \{1, ..., \mathcal{T}\}$
$S_{k\tau}$	Overall supply of resource k at period τ
$D_{k\tau}$	Overall demand for resource k at period τ
B_k^l	Base-line allocation for agent l of machine type k
Bids	
$\mathbf{U}^l$	Bid by agent l, expressed as $\{U_{1,1}^l, \cdots, U_{K,1}^l, \cdots U_{1,\mathcal{T}}^l, \cdots, U_{K,\mathcal{T}}^l\}$
$U_{k,\tau}^l$	Utilization by agent l of machine type k at period τ
$\mathcal{U}^l$	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 $\mathcal{U}^l$
$u_{k\tau}^l$	Utility price of agent l for machine k at period τ
$p_{k\tau}^r$	Bid price for machine k at period τ in 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

resource cost incurred for that bid, denoted $\mathcal{RC}^l$, is computed as $\sum_\tau \sum_k p_{k\tau} U_{k\tau}^l$. Finally, the total sum cost associated with the bid, denoted $\mathcal{SC}^l$, is simply $MTC^l + \mathcal{RC}^l$.

The other notations in the table will be explained as the paper proceeds.

As an example, we provide a sample problem instance, comprising four agents (or simply four job-lists), each with ten jobs on two machines types. This sample problem is abstracted from a real-life problem scenario in container terminal operations, which has the following features.

- A job comprises three flow-shop operations and no wait is allowed between consecutive operations.
- The three operations are performed by a quay crane, a truck and a yard crane, respectively, and nonpreemptively.
- Each job has a release time and due time. All jobs must be completed, and a tardiness penalty will be incurred if it is completed after its due time.
- A quay crane is assumed as the proxy agent for a given job-list, while trucks and yard cranes are common machine resources.

The basic input data for the above sample instance and the detailed job-list data are given in Tables II and III. For this instance, we take one hour as one time period, and feasible solutions produced by various approaches are found in Table V (which is shown later in Section IX, Experimental Results.

TABLE II
SAMPLE PROBLEM FOR FOUR AGENTS AND TWO MACHINES

Agent -ID	Release Time	Due Time	Makespan Price	Tardiness 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

$S_{k\tau}$ $\forall\tau$	Machine 1	Machine 2
	16	8

B_k^l $1 \le l \le 4$	Machine 1	Machine 2
	2	1

TABLE III
PROCESSING TIME FOR THE FOUR 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]

III. SYSTEM ARCHITECTURE

In this section, we present the overall system architecture and our proposed auction protocol.

We propose a simple system architecture comprising three modules.

- System initialization, which will perform initialization and trigger the auction until one of the stopping criteria has been reached (details below).
- Auction protocol, which includes the bid-generation and price adjustment procedures (overview below). Bid generation will be discussed in detail in Section IV, while price adjustment, in Section V. For the latter, our ideas of utility prices and variable step-size will be separately discussed in Sections VI and VII, respectively.
- 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 VIII.

A. Initialization and Stopping Criteria

In the initialization step, we perform preprocessing 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 [23]. Second, we determine the upper bound resource capacities that will be required by the agents by solving a centralized flowshop problem with infinite resources, using the algorithm proposed in [30]. 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 IV).

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 in 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.

B. Overview of Auction Protocol

Our proposed auction protocol comprises mainly two components: (a) Bid Generation and (b) Price Adjustment. The entire auction process will proceed in rounds. Within each round, the bidder agents will perform Bid Generation to generate their respective bids; 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.

On the bidder end, each agent l has to decide on its bid in response to the prevailing resource prices. This is called the Bid Generation (*BidGen*) problem which, in our context, is the multimachine flowshop 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 bidder.
- Announce to start a new round of BidGen, or stop the auction according to the stopping criterion.

Kutanoglu and Wu [18] observed that the above iterative combinatorial auction protocol is analogous to Lagrangian relaxation [8] 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 subproblems 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 tatonnement price adjustment scheme. This corresponds to an iteration in solving the Lagrangian dual master problem (by using the subgradient 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.

Similarly, this analogy carries over nicely to other deterministic scheduling and resource allocation problems so long as they are able to follow the same Lagrangian-based decomposition procedure. In our context, we follow the same idea of dualizing the resource capacity constraints as [18].

IV. 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 the novelty of our approach lies in the notion of a makespan matrix. We then extend our approach to solve the multiperiod 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.

Recall that the agent objective is to minimize the makespan-tardiness cost MTC. Given that resources are priced, the problem of generating an optimal bid in response to the resource prices is essentially a search procedure to find a bid that minimizes the minimum total cost $\mathcal{SC}$. Let SchGen (Schedule Generation) denote the function that returns this value by calling a single agent scheduling problem using resources specified in a given bid. For this paper, our focus is not to investigate algorithms for computing SchGen (since there are already well-studied algorithms for various underlying scheduling problems), but rather we treat it as a black box $\mathcal{A}$.

One critical concern in bid generation is to be able to determine 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.

Hence, 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 derived from

$$\mathcal{M}(\mathbf{U}^l) \triangleq SchGen(\mathcal{L}^l, \mathbf{U}^l, \mathcal{A}). \tag{1}$$

Over the entire bid space $\mathcal{U}^l$, given that the job-list information is fixed throughout the auction process, we define the **Makespan Matrix** for an agent l as $\mathcal{M}(\mathcal{U}^l)$. Similarly, we can derive the **Sum Cost Matrix** by summing the respective MTC value and prevailing $\mathcal{RC}$ value.

Strictly speaking, the term *matrix* can be used in the simple case when $K = 2$, $\mathcal{T} = 1$. For higher dimensions, it will become a hypercube. 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.

If the BidGen problem is single-period (i.e., when $\mathcal{T} = 1$), bids are made on resources at a constant level throughout the time horizon. We propose a local search method that begins with an initial bid (say 2 units of resource 1 and 1 unit of resource 2, for the sample problem). It searches along either dimension of the sum cost matrix until it reaches a 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). Extrapolating to more than 2 dimensions, our local search can be seen a hill-climbing neighborhood search in the sum cost matrix space—it always increases one resource type by one unit, and continues doing so until no further improvement is possible; it then repeats the search with another resource type. Note that as the search proceeds, undefined matrix (and hence sum cost) values become defined, and are stored (for future search).

Next, we extend the single-period algorithm to handle the multiperiod BidGen problem. It is an improvement search over $MaxIterations$ until no further improvement is obtained from one iteration to the next. For simplicity in notation, we omit the subscript l (agent index), and hence $U^l_{k\tau}$ is simplified to be $U_{k\tau}$, while keeping in mind this BidGen is done for each agent. Let U^* denote the best-so-far bid (in terms of the objective value). The following pseudocode describes our approach:

1) based on the current resource prices p, compute the sum cost matrix and initialize $U^*_{k\tau}$;
2) for $(iter = 1 \ldots MaxIterations)$ until no further improvement:
   - for $(\tau = 1 \ldots \mathcal{T})$
     - perform single-period search for period τ, with bids for other periods τ' fixed as $U^*_{k\tau'}$;
     - update makespan matrix.
   - if an improved solution can be found then update U^*.

We like to remark that although we propose a local search procedure for Bid Generation, Bertsimas [1] shows that the local optima obtained is indeed a global optima under the assumption that the objective function (i.e., sum cost $\mathcal{SC}^l$) is convex with respect to the search space ($\mathbf{U}^l$).

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

$$u^l_{k\tau} \triangleq \frac{\partial MTC^l}{\partial U^l_{k\tau}} = \begin{cases} W^l_m \left| \frac{\partial \mathcal{M}}{\partial U^l_{k\tau}} \right|, & \text{if } \mathcal{M} < D^l - R^l \\ \left(W^l_m + W^l_d\right) \left| \frac{\partial \mathcal{M}}{\partial U^l_{k\tau}} \right|, & \text{if } \mathcal{M} \geq D^l - R^l. \end{cases} \tag{2}$$

Equation (2) gives the formal definition of utility price. It is the makespan penalty W^l_m (or $W^l_m + W^l_d$ if there is tardiness) times $\partial \mathcal{M}/\partial \mathbf{U}^l_{k\tau}$, 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 penalty W^l_m is (virtual) dollars per hour. The unit $\partial \mathcal{M}/\partial U^l_{k\tau}$ 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

$$u^l_{k\tau}\big|_{\mathbf{U}^*} = MTC^l\left((\cdots, {U^*}_{k-1,\tau}, {U^*}_{k\tau} - 1, {U^*}_{k+1,\tau}, \cdots, {U^*}_{K\mathcal{T}})\right) - MTC^l(\mathbf{U}^*). \tag{3}$$

The utility price will be used in the price adjustment strategy presented in the next section.

V. Overview of Price Adjustment Strategy

In this section, we present an overview of our contributions in the light of literature on price adjustment. This is followed by the subsequent two sections that present our contributions in detail.

The classical price adjustment procedure is given as follows.

1) Collect bids from all agents and compute aggregate demand, as (4).
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 (5), which is dependent on the step size and the demand-supply gap and must be non-negative

$$D_{k\tau} = \sum_{l=1}^{L} U^l_{k\tau} \tag{4}$$

$$p^{r+1}_{k\tau} = \max\left\{0, p^r_{k\tau} + s^r \cdot (D_{k\tau} - S_{k\tau})\right\}. \tag{5}$$

Fisher [8] 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 and Wu [18] follows the same strategy. For the highly intensive computation due to the NP-hardness of the underlying scheduling problem, recent research (e.g., [5] and [20]) turn to such methods that do not explicitly calculate $\mathcal{LB}$ and $\mathcal{UB}$.

In this paper, building on recent works, we propose a price adjustment scheme that departs from the classical price adjustment scheme as follows.

- Instead of $\mathcal{LB}$ and $\mathcal{UB}$, we use a simpler term *Average Price* as the numerator of the step size formula (see next section for details).
- We propose a new technique to compute the step size that combine two ideas from micro-economics and control theory to help us achieve equilibrium quickly.

Our first idea is to incorporate what is termed as the utility price *instead of bid price* in the price adjustment formula. We find that although the conventional scheme based on bid prices is straightforward 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 VI.

Our second idea is to use a variable (instead of fixed) step size s^r that changes from one iteration r to the next. A continuous function has been designed for obtaining a feasible solution quickly. The function is defined within $(-\infty, \infty)$ and the value is within (0, 2) based on the theoretical result in [15]. While the idea of a variable step size is analogous to adaptive control, we depart from it in the following aspects.

- The energy function in adaptive control is usually symmetrical around some special point. For example, both the Lyapunov function $V_{\bar{q}_c} = \|\bar{q} - \bar{q}_c\|^2$ by Fua *et al.* [9], and the trench function $f(d) = \sqrt{1+d^2} - 1$ by Ge and Fua [13] are symmetric. In our case, the price adjustment function is modeled after the Lagrangian dual master problem (as described in the Systems Architecture section), and is asymmetric.
- In adaptive control, convergence is defined as local and/or global control error (setpoint—feedback) approaching zero, whereas in our case, underlying the price adjustment process is an optimization problem and hence convergence is defined as prices approaching some unknown values (since the optimal price vector is yet unknown).
- In adaptive control, the error could approach zero in either directions (positive or negative) symmetrically, while in our case the price must always be positive.

Despite the above major differences, in this paper, we seek to apply adaptive control concepts in the price adjustment process to speed up convergence. Details of our proposed variable step size technique will be explained in Section VII.

VI. Price Adjustment With Utility Price

In the following, we present two price adjustment strategies—one based on the conventional bid prices, while the other using our proposed utility prices. These two approaches will be compared experimentally in Section IX.

A. Price Adjustment With bid Price (PA-B)

The detailed formulation is as follows.

1) Calculate the aggregate demand as (4).
2) Step size
$$s^r = \alpha \cdot = \frac{\frac{\sum_k \sum_t p^\tau_{k\tau} D_{k\tau}}{\sum_k \sum_t D_{k\tau}}}{\sqrt{\frac{\sum_{k=1}^{K} \sum_{\tau=1}^{T^r} (D_{k\tau} - S_{k\tau})^2}{K \cdot T^r}}}.$$
3) Calculate new price for next iteration as (5).

B. Price Adjustment With Utility Price (PA-U)

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).
2) Calculate average utility price $u_{k\tau} = \sum_l^L u^l_{k\tau} U^l_{k\tau} / \sum_l^L U^l_{k\tau}$, for $\forall 1 \le k \le K, 1 \le \tau \le \mathcal{T}$.
3) Step size
$$s^r = \alpha \cdot = \frac{\frac{\sum_k \sum_t u_{k\tau} D_{k\tau}}{\sum_k \sum_t D_{k\tau}}}{\sqrt{\frac{\sum_{k=1}^{K} \sum_{\tau=1}^{T^r} (D_{k\tau} - S_{k\tau})^2}{K \cdot T^r}}}.$$
4) Calculate new price for next iteration as (5).

C. Comparing PA-B and PA-U

The major differences between PA-B and PA-U are the following:

1) The convergence of PA-B depends on the bid price, while PA-U does not. During the auction process, once there is a round r^* such that $p^{r^*}_{k\tau} = 0, \forall k\tau$, then $p^r_{k\tau} \equiv 0, \forall r \ge r^*, \forall k\tau$, which means the price thereafter will always be 0, whether or not a feasible solution has been reached.
2) The rate of convergence of PA-B depends on the initial price, while PA-U is insensitive to it. The closer the initial bid price is to zero, the slower the convergence for PA-B.

These differences are precisely needed to speed up convergence. In the following, we provide an analytical insight on why we believe that the use of utility prices is a superior approach than those that use conventional bid prices in speeding up convergence.

It has been observed experimentally that the poor convergence of conventional price adjustment schemes such as PA-B occurs when the bid price falls to a zero value. Hence, to improve convergence, one remedy is to ensure that the utility price is NOT zero. In the following, we present properties under which this property holds.

Consider a particular agent l whose bid $\mathbf{U}^*$ is represented by a vector $[U^*_{1,1}, U^*_{2,1}, \ldots, U^*_{k\tau}, \ldots, U^*_{K,\mathcal{T}}]$.

Lemma 6.1: If the makespan matrix $\mathcal{M}(\mathcal{U}^l)$ satisfies the following properties.

— *Monotonicity*: For any time τ and any machine k, the makespan value is monotonically nonincreasing with machine utilization.
— *Convexity*: For any time τ and any machine k, the makespan value is convex with respect to machine utilization.
— *BackDiff*: The calculation of utility price is based on backward difference, given in (3).

Then, once $\exists \tau, k$ such that $u_{k\tau}|_{U^*} = 0, u_{k\tau}|_U \equiv 0$, for all U such that $U_{k\tau} \ge U^*_{k\tau}$.

Proof: And $u_{k\tau}|_{U^*_k} = 0$ means that

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

Due to convexity

$$\mathcal{M}(\ldots, U^*_{k\tau} + 1, \ldots) + \mathcal{M}(\ldots, U^*_{k\tau} - 1, \ldots) \ge 2\mathcal{M}(\ldots, U^*_{k\tau}, \ldots)$$

then

$$\mathcal{M}(\ldots, U^*_{k\tau} + 1, \ldots) \ge \mathcal{M}(\ldots, U^*_{k\tau}, \ldots).$$

Because of monotonic non-increasing property

$$\mathcal{M}(\ldots, U^*_{k\tau} + 1, \ldots) \le \mathcal{M}(\ldots, U^*_{k\tau}, \ldots).$$

Then, there will be

$$\mathcal{M}\left(\ldots, U^*_{k\tau}+1, \ldots\right) = \mathcal{M}\left(\ldots, U^*_{k\tau}, \ldots\right).$$

It is further extended as

$$\mathcal{M}(\ldots, U_{k\tau}, \ldots) \equiv \mathcal{M}\left(\ldots, U^*_{k\tau}, \ldots\right), \quad \forall U_{k\tau} \geq U^*_{k\tau}.$$

In other words, $MTC^l(U)$ will not change for $U_{k\tau} \geq U^*_{k\tau}$. Then, by backward difference, $u_{k\tau}|_{U_k} \equiv 0$, for all U such that $U_{k\tau} \geq U^*_{k\tau}$. **Q.E.D.**

In particular, for resource-constrained environment where the total number of operations to be performed on a particular machine k is always greater than the baseline allocation. i.e., for any agent l, we have $\sum_{m_{ij}=k}^{1\leq i\leq N^l, 1\leq j\leq o^l_i} 1 > B^l_k$, a nonzero utility price can be guaranteed. We have the following corollary.

Proposition 6.2: In a *competitive* environment, if the makespan matrix $\mathcal{M}(\mathcal{U}^l)$ satisfies the above 3 properties {*Monotonicity, Convexity, BackDiff*}, *it never happens that* $u_{k\tau}|_{U^*_k} = 0, \forall \tau, k$ *except at the upper bound machine capacity*.

Proof: Following Lemma 6.1, in a resource-constrained environment, for any k and τ, we have the following strict inequality:

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

Hence, $u^l_{k\tau}|_{B^l} > 0$. **Q.E.D.**■

Note that the property of monotonicity should be always true, since adding resource should never increase the schedule's makespan. Unfortunately, the property of convexity cannot be enforced generally. Nonconvexity arises because the schedule requires 2 or more resources that are interdependent.

VII. Price Adjustment With Variable Step Size

Given that the nice property of convexity presented in the previous section may not always hold true in practice, we may still get very small values for utility prices, which will in turn impact the speed of convergence. In this section, we address this shortcoming by proposing the idea of variable step size in price adjustment.

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, we choose a larger factor 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 we already have feasible solutions), we choose a smaller factor to fine tune the optimality.
- The standard deviation is considered—In an optimal solution, $std^{T^r-1}_{t=1}(D_{kt})$ should be as small as possible.

We propose that the step-size s^r given in (5) to be replaced by two components $\alpha^r \cdot \beta^r_k$, where α^r intuitively is the utility price component. As for β^r_k, we further break into $\beta^r_{std}(k) \cdot \beta^r_{speed}(k)$ representing the factor of standard deviation $\beta^r_{std}(k)$ and the speed factor $\beta^r_{speed}(k)$, which are dependent on specific machine type k. $\beta^r_{std}(k)$ is derived from the standard deviation of machine utilization for each type, and $\beta^r_{speed}(k)$ is a function of maximum net demand. They are defined as follows:

$$x_k = \mathcal{N}^M_k = \max_t\{D_{kt} - S_{kt}\} \tag{6}$$

$$\beta_{std}(k) = \max\left\{1, sign(x_k) \cdot std^{T^r-1}_{t=1}(D_{kt})\right\} \tag{7}$$

$$\beta_{speed}(k) = f_{speed}(x_k) = \begin{cases} \exp\left(-x_k^2\right) & \text{if } x_k < 0 \\ 2 - \exp\left(-x_k^2\right) & \text{if } x_k \geq 0 \end{cases}. \tag{8}$$

Details of our consideration are given as follows:

- The standard deviation is measured 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 resources in the last period.
- The factor $\beta_{std}(k)$ is only considered when the solution is NOT feasible, i.e., $sign(\mathcal{N}^M_k) > 0$, for some specific machine type k. When it is a feasible solution, $\beta^r_{std}(k) = 1$ since $sign(\mathcal{N}^M_k) < 0$ $std^{T^r-1}_{t=1}(D_{kt}) > 0$ always.
- The speed factor is a function of the maximum net demand, and the formulation given in (8) is just one possible implementation. More options are further studied later.
- The speed factor function is continuous up to at least the second order. They are all equal to 1 when $x_k = 0$, which means a feasible solution with the 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 pth order continuous.
— FuncSpeed-2: $f(0) = 1$, as explained 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 [8]. Furthermore, in [8], it has been shown that the following two properties for the step-size s^r are sufficient but NOT necessary conditions for achieving convergence:

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

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

A. 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 $C \in (0, 2]$

$$f_1(x) = \begin{cases} \exp\left(-|x|^p\right), & x < 0 \\ 1 + \frac{C}{2}\left(1 - \exp\left(-|x|^p\right)\right), & x \geq 0 \end{cases}. \tag{9}$$

Above function is p-*th* order continuous since it can be verified that $f'_1(0^-) = f'_1(0^+) = 0$.

$$f'_1(x) = \begin{cases} p \cdot (-x)^{p-1} \cdot \exp\left(-|x|^p\right), & x < 0 \\ \frac{p \cdot C}{2} x^{p-1} \cdot \exp(-x^p), & x \geq 0 \end{cases}. \tag{10}$$

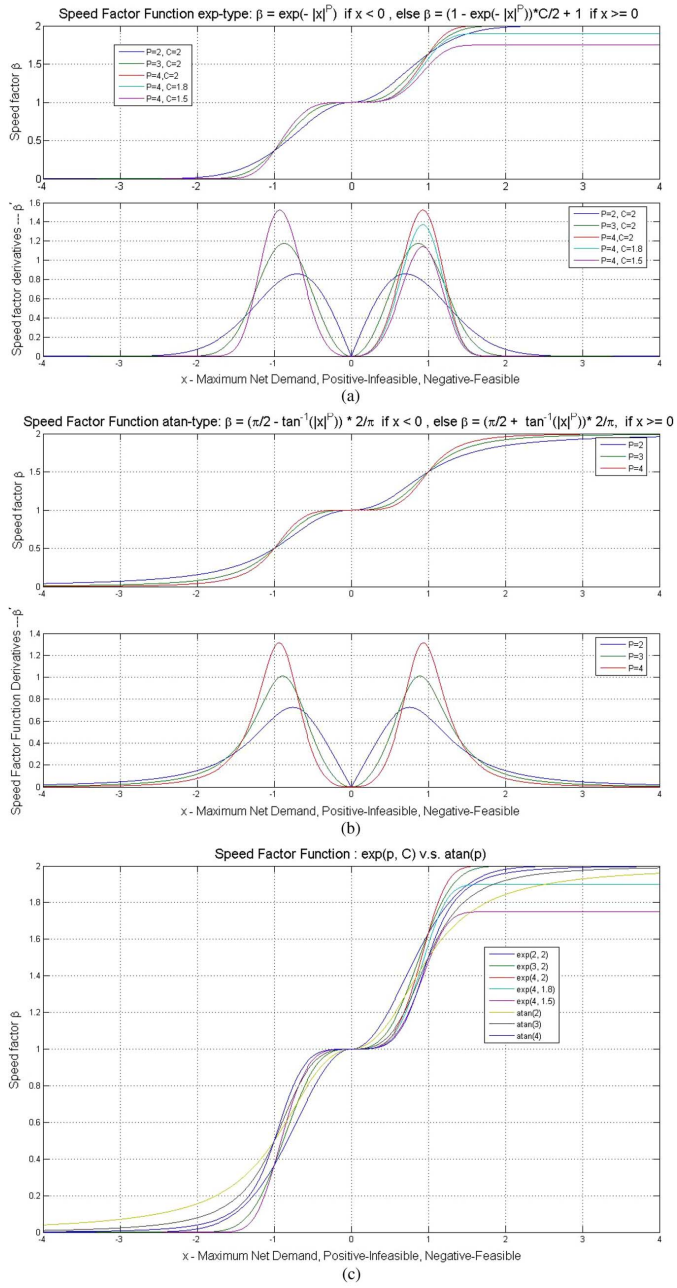


Fig. 1. Comparison of speed factors. (a) Speed Factor exp-type, $f_1(x)$ and $f_1'(x)$. (b) Speed Factor atan-type, $f_2(x)$ and $f_2'(x)$. (c) Speed Factor in one plot.

Besides the continuous property, there are $f_1(-\infty) = 0^+$ and $f_1(\infty) = (C/2) + 1$. Since $C \in (0, 2]$, there is $(C/2) + 1 \in (1, 2]$. 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. 1(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. $p \geq 2$, $p \in \mathcal{N}$

$$f_2(x) = \begin{cases} 1 - \frac{2}{\pi} \cdot \tan^{-1}\left(|x|^p\right), & x < 0 \\ 1 + \frac{x}{\pi} \cdot \tan^{-1}\left(|x|^p\right), & x \geq 0 \end{cases}. \tag{11}$$

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

$$f_2'(x) = \begin{cases} \frac{2p}{\pi} \frac{(-x)^{p-1}}{1+x^{2p}}, & x < 0 \\ \frac{2p}{\pi} \frac{x^{p-1}}{1+x^{2p}}, & x \geq 0 \end{cases}. \tag{12}$$

Further, $f_2(-\infty) = 0^+$ and $f_2(\infty) = 2^-$. Comparisons of some $f_2(x)$ with different parameters p are shown in Fig. 1(b).

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

B. Integrated Price Adjustment

Combining both utility price and variable step size concepts, we arrive at the following price adjustment formulas listed in (13)–(15).

The step-size s^r in (5) is now replaced by two components α^r and β_k^r.

- Equation (13) shows that α^r is the average of utility price over *Root Mean Square* (RMS) net demand.
- Equation (14) shows that β_k^r which augments the standard deviation factor by considering variable step size.
- Equation (15) updates the price vector for the next round.

Again, the reader will note that our formulation does not calculate the lower bound $\mathcal{LB}$ and upper bound $\mathcal{UB}$, as in [8] and [18], since they are computationally expensive

$$\begin{aligned} \alpha^r &= \frac{AVE_{kt}(UtilityPrice)}{RMS_{kt}(Demand - Supply)} \\ &= \frac{\frac{\sum_k \sum_t u_{kt} D_{kt}}{\sum_k \sum_t D_{kt}}}{\sqrt{\frac{\sum_k^K \sum_t^{T^r} (D_{kt} - S_{kt})^2}{K \cdot T^r}}} \end{aligned} \tag{13}$$

$$\begin{aligned} \beta_k^r &= \beta_{std} \cdot \beta_{speed} \\ &= \max\left\{1, sign\left(\mathcal{N}_k^M\right) \cdot std_{t=1}^{T^r - 1}(D_{kt})\right\} \\ &\quad \cdot \begin{cases} \exp\left(-\mathcal{N}_k^{M^2}\right), & \text{if } \mathcal{N}_k^M < 0 \\ 2 - \exp\left(-\mathcal{N}_k^{M^2}\right), & \text{if } \mathcal{N}_k^M \geq 0 \end{cases} \end{aligned} \tag{14}$$

$$p_{kt}^{r+1} = \max\left\{0, p_{kt}^r + \alpha^r \beta_k^r \cdot (D_{kt} - S_{kt})\right\}. \tag{15}$$

VIII. Resource Re-Allocation Strategy

Although our proposed adaptive auction scheme can improve the speed of convergence and achieve feasibility, it often results as a local optimal solution. On careful examination, we discover that it is possible to obtain better solutions through reallocation of resources that has not been fully utilized after the auction process is completed. We perform a postprocessing phase with the following resource reallocation strategy. We apply a simple greedy algorithm based on makespan-tardiness cost as follows

We consider unallocated resources in each period $\tau = 1, \ldots, \mathcal{T}$. For each non-fully utilized resource, we simply allocate the spare units to the agent l such that:

- MTC^l is the largest in value;
- its job-list release time is on or after period τ.

After each reallocation, the agent will reconstruct its schedule and recalculate its objective value (MTC). Notice that the objective value cannot be worse, since additional resources have been allocated; moreover after reallocation, it is also possible

TABLE IV
ILLUSTRATION OF EFFECT OF RESOURCE-REALLOCATION

Before Reallocation							
Agent -id	(Res-1, Res-2) per Hour					Makespan (hour)	Total Cost
	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00		
1	(3, 1)	(1, 1)	(15, 3)	(6, 4)		3.5	350
2	(13, 2)	(9, 5)	(1, 1)	(2, 2)	(1, 2)	4.35	435
3		(3, 1)	(1, 2)	(9, 2)	(10, 5)	3.85	385
4		(10, 2)	(7, 4)	(7, 2)		3	300
	Free Resources per Hour						
	(8,13)	(1,7)	(0,6)	(0,6)	(13,9)		
After Reallocation							
Agent -id	(Res-1, Res-2) per Hour					Makespan (hour)	Total Cost
	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00		
1	(5, 2)	(1, 2)	(14, 3)	(5, 4)		3.35	335
2	(15, 2)	(11, 6)	(1, 2)			2.15	215
3		(2, 1)	(1, 1)	(11, 2)	(12, 5)	3.8	380
4		(10, 2)	(7, 4)	(7, 2)		3	300
	Free Resources per Hour						
	(4,12)	(0,5)	(1,6)	(1,8)	(12,11)		

that the resources that agents bidded for earlier may not be utilized and hence they can be returned to the pool. The above procedure is repeated until none of the agent objective can improve further.

An example of the effect of reallocation for the sample instance is given in Table IV.

IX. EXPERIMENTAL RESULTS

First, we benchmark our approach against a MIP (mixed integer programming) model called **ResAlloc-MIP** whose objective function is the sum of agent objectives. Details of the MIP formulation can be found in [30]. We present detailed results on the sample problem instance given in Section II, followed by a summary of results on a number of other problem instances. This is followed by experimental results specific to price adjustment—an experimental comparison on different price adjustment strategies, and different variable step size parameters.

A. Comparison on Sample Problem Instance

Here, we provide a detailed commentary of a comparison of the MIP model with both PA-B and PA-U strategies. Table V gives the allocation for each agent given by the MIP, PA-B, and PA-U solutions, respectively. Table VI shows the comparison of the total cost and average makespan for four agents. We can see that the solution produced by PA-U is nearer to optimal, and the run time is comparable with that of PA-B.

Another observation is that the run-time performance of PA-B is dependent on the initial price settings, as shown in Table VII. In all the 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 PA-B is comparable with that of PA-U. 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 we have an infeasible solution. Under the PA-U strategy, we can achieve a feasible solution within three iterations, while it takes about 30 iterations for the PA-B strategy. When the initial price is zero, PA-B fails to find a feasible solution while PA-U stills achieves a feasible solution within a short time. This result is consistent with our proposition.

TABLE V
MULTIPERIOD RESOURCE ALLOCATION, SOLUTION COMPARISON MIP, PA-B, AND PA-U

Agent -Id	Algo-rithm	Allocation (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)
	PA-B	(5, 2)	(4, 2)	(1, 1)	(0, 0)
	PA-U	(7, 2)	(4, 2)	(0, 0)	(0, 0)
2	MIP	(7, 2)	(4, 2)	(0, 0)	(0, 0)
	PA-B	(8, 3)	(3, 3)	(0, 0)	(0, 0)
	PA-U	(7, 3)	(3, 2)	(1, 1)	(0, 0)
3	MIP	(0, 0)	(3, 2)	(7, 4)	(0, 0)
	PA-B	(0, 0)	(4, 2)	(4, 1)	(1, 2)
	PA-U	(0, 0)	(5, 2)	(5, 4)	(0, 0)
4	MIP	(0, 0)	(5, 1)	(5, 2)	(0, 0)
	PA-B	(0, 0)	(3, 1)	(2, 2)	(0, 0)
	PA-U	(0, 0)	(4, 1)	(5, 3)	(0, 0)
Sub -total	MIP	(13, 4)	(15, 7)	(12, 6)	(0, 0)
	PA-B	(13, 5)	(16, 8)	(11, 6)	(1, 2)
	PA-U	(14, 5)	(16, 7)	(11, 8)	(0, 0)

TABLE VI
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
PA-B	2.1	2.0	2.3	1.6	2.0	1445.8	53.8
PA-U	2.0	2.1	1.9	2.0	2.0	1358.3	32.7

TABLE VII
COMPARISON FOR DIFFERENT INITIAL PRICES

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

B. Comparison With MIP Model

We next compare the solution quality of our approach with the ResAlloc-MIP model. We generated ten problem instances to compare on both the solution quality and run time performance. These instances are extracted from a real-world resource allocation problem from a container terminal. We like to remark that for the MIP model, the CPLEX solver cannot even find a feasible solution within 6 h for large-scale problem instances. In our experimental setup, we consider medium-sized hard problem instances with four agents, each having a 20×3 standard flowshop scheduling problem (i.e., a job-list with 20 jobs on 3 machines), and the following settings:

- A discrete-time domain where one period is equal to 40 time units.

TABLE VIII
SOLUTION COMPARISON

2 agents start 8:00 and the other 2 agents start at 9:00							
	SumCost (Makespan + Tardiness)			PA-U over CPLEX		CPLEX 2 hours	PA-U & VSS
	PA-U & VSS	CPLEX (hour)		CPLEX (hour)		Duality	RunTime
		2	12	2	12	Gap	(sec)
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				99%	121%		4% of 2 hours

- Two agents start their job-lists at 8:00 and the other 2 start at 9:00 (i.e., 20 time-slot later), and each job comprises three operations.
- The tardiness and makespan penalties are equal for all agents—500 and 100 per period, respectively.
- There are 4 machines of type 1, 16 machines of type 2, and 24 machines of type 3.
- 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 our 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.

In terms of computation time, we let the CPLEX solver run for 2 h 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 PA-U auction for the same ten problem instances, and the comparison is given in Table VIII.

We like the readers to take note of the following points.

- The above problem instances are exactly the same as those presented in [21]. The current solution is done by auction with both PA-U and Variable Step-Size (*VSS*), while our preliminary result in [21] was achieved with PA-U and a fixed step size.
- Our previous results were obtained using CPLEX 10.0, while we used CPLEX 10.1.1 to obtain results in this paper.
- By comparison of the total cost (i.e., the smaller objective function the better), our solution is slightly better than the ResAlloc-IP solution obtained by CPLEX in 2 h, but worse than that obtained in 12 h.
- By comparison of the solution time, solution by our approach with PA-U and VSS is much faster. It can achieve price equilibrium within 10 minutes while the comparable results by CPLEX within 2 h.

The reason why PA-U is better than PA-B has been explained both theoretically and empirically. On the other hand, the theory behind why VSS outperforms fixed-step-size is a subject matter for future research. In the following, we will show this point empirically.

C. 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 four 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.

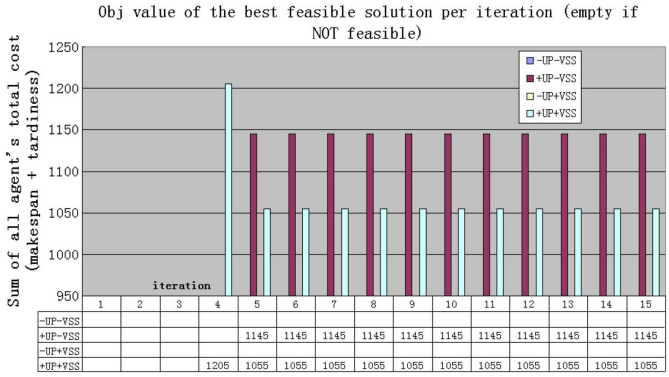


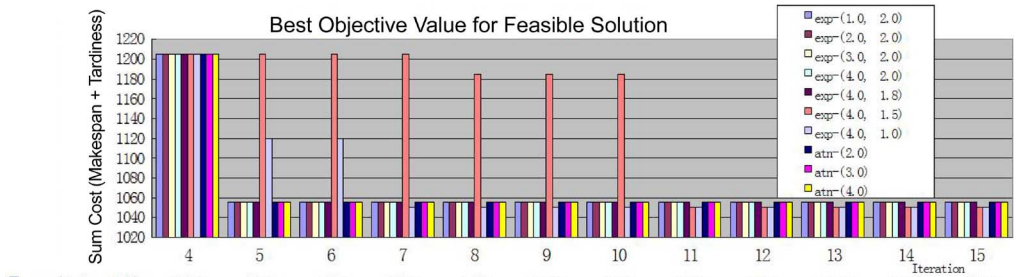
Fig. 2. Empirical study of the effect of utility price and variable step size in price adjustment.

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 we terminate after a maximum of 15 iterations. The result is shown in Fig. 2.

From the experiment result shown in Fig. 2, we have the following observations.

- For the two 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 IX-A. This point is clear since we observe the formula in (15). 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), we can see that a feasible solution is found at the fifth 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), we can see that a feasible solution is found at the fourth iteration. The objective function value 1205 of this solution is near optimal but not the best one. Subsequently, at the fifth 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 that combines the concepts of utility price and variable step size proposed in this paper. The performance of such a combined setting +UP+VSS exceeds those of all other settings.



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. 3. Sensitivity of different parameters in speed factor function.

D. 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 (9) and (11), there are two types of speed factor functions, i.e., $\exp$ and $atan$. The $\exp$ function has two parameters (offset parameter C and index parameter p), while $atan$ has one parameter (index parameter p).

In total, we tested ten different problem settings, the first seven 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 three cases, $p \in \{2.0, 3.0, 4.0\}$. The experimental results are shown in Fig. 3.

We observe the following 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 final solutions are almost the same.

In summary, we find 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 our approach is insensitive to the parameter values.

X. Conclusion

In this paper, we propose an adaptive auction scheme that contains two novel ideas for price adjustment: the concepts of utility pricing and variable step size, along with an efficient algorithm for performing bid generation using the idea of makespan matrix. Combined with the preprocessing and postprocessing steps, we demonstrate the power of the entire system in solving large-scale problem instances. On utility pricing, we demonstrated experimentally that it has a better convergence property than a conventional price adjustment scheme based on bid price, and furthermore does not depend on the values of the initial prices—although the price we pay is an increased communication overhead between the auctioneer and bidders. Analytically, we have also shown that our approach converges to global optimality albeit under restricted condition. On variable step size, we show experimentally that it performs better than fixed step size with respect to solution quality and speed, and is insensitive to the underlying speed function parameters.

Although our discussion centred around the flowshop scheduling problem, we believe our auction scheme is generic in the sense that the bid generation can be readily replaced by another algorithm to solve another decentralized resource allocation problem with little or no changes to the underlying price adjustment process.

A salient point to conclude this paper 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 we do not see too much in the classical auction literature.

References

[1] D. Bertsimas, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1988.

[2] S. Bikhchandani and J. W. Mamer, "Competitive equilibrium in an exchange economy with indivisibilities," *J. Econ. Theory*, vol. 74, no. 2, pp. 385–413, 1997.

[3] T. S. Chandrashekar, Y. Narahari, C. H. Rosa, D. M. Kulkarni, J. D. Tew, and P. Dayama, "Auction-based mechanisms for electronic procurement," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 3, pp. 297–321, Jul. 2007.

[4] J. Q. Cheng and M. P. Wellman, "The Walas algorithm: A convergent distributed implementation or general equilibrium outcomes," *Comput. Econ.*, p. 12, 1998.

[5] G. Confessore, S. Giordani, and S. Rismondo, "A market-based multi-agent system model for decentralized multi-project scheduling," *Ann. Oper. Res.*, vol. 150, no. 1, pp. 115–135, 2007.

[6] P. Cramton, Y. Shoham, and R. Steinberg, *Combinatorial Auctions*. Cambridge, MA: MIT Press, 2006.

[7] M. Drozdowski, "Scheduling multiprocessor tasks—An overview," *Eur. J. Oper. Res.*, p. 94, 1996.

[8] M. L. Fisher, "An application oriented guide to Lagrangian relaxation," *Interfaces*, vol. 15, no. 2, pp. 10–21, 1985.

[9] C. H. Fua, S. S. Ge, K. K. Do, and K. W. Lim, "Multirobot formations based on the queue-formation scheme with limited communication," *IEEE Trans. Robotics*, vol. 23, no. 6, pp. 1160–1169, 2007.

[10] R. Gagliano, M. Fraser, and M. Schaefer, "Auction allocation of computing resources," *Commun. ACM*, vol. 38, no. 6, pp. 88–102, 1995.

[11] J. G. Gaines and T. J. Lyons, "Variable step size control in the numerical solution of stochastic differential equations," *SIAM J. Appl. Math.*, vol. 57, no. 5, pp. 1455–1484, 1997.

[12] D. Garg and Y. Narahari, "An optimal mechanism for sponsored search auctions and comparison with other mechanisms," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 4, pp. 641–657, Oct. 2009.

[13] 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.

[14] R. L. Graham and E. L. Lawler, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Discrete Math.*, pp. 287–326, 1979.

[15] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Math. Programming*, vol. 6, pp. 62–88, 1974.

[16] P. Joyce, "The Walarsian tatonnement mechanism and information," *RAND J. Econ.*, vol. 15, no. 3, pp. 416–425, 1984.

[17] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, 3rd ed. New York: Springer, 1999.

[18] E. Kutanoglu and S. D. Wu, "On combinatorial auction and Lagrangean relaxation for distributed resource scheduling," *IIE Transactions*, p. 31, 1999.

[19] E. Kutanoglu and S. D. Wu, "Improving scheduling robustness via preprocessing and dynamic adaptation," *IIE Transactions*, vol. 36, pp. 1107–1124, 2004.

[20] H. C. Lau *et al.*, "Multi-period combinatorial auction mechanism for distributed resource allocation and scheduling," in *Proc. Int. Conf. Intell. Agent Technol.*, 2007, pp. 407–411.

[21] H. C. Lau *et al.*, "Utility pricing auction for multi-period resource allocation in multi-machine flow shop problems," in *Proc. Int. Conf. Electron. Commerce*, 2008, (doi>10.1145/1409540.1409547).

[22] P. L. Lorentziadis, "Pricing in multiple-item procurement auctions with a common to all items fixed cost," *Eur. J. Oper. Res.*, vol. 190, pp. 790–797, 2008.

[23] C. Oğuz and M. F. Ercan, "A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks," *J. Scheduling*, vol. 8, no. 4, pp. 323–351, 2005.

[24] A. B. Pritsker, L. J. Watters, and P. Wolfe, "Multi-project scheduling with limited resources: A zero-one programming approach," *Manage. Sci.: Theory*, vol. 16, no. 1, pp. 93–108, 1969.

[25] F. Riane, A. Artiba, and S. E. Elmaghraby, "A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan," *Eur. J. Oper. Res.*, vol. 109, pp. 321–329, 1998.

[26] E. Toczywski and I. Zoltowska, "A new pricing scheme for a multi-period pool-based electricity auction," *Eur. J. Oper. Res.*, vol. 197, no. 3, pp. 1051–1062, 2008.

[27] A. P. Vepsalainen and T. E. Morton, "Priority rules for job shops with weighted tardiness costs," *Manage. Sci.*, vol. 33, no. 8, pp. 1035–1047, 1987.

[28] L. Walras, *Elements of Pure Economics*. Homewood, IL: Richard Irwin, 1954.

[29] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason, "Auction protocols for decentralized scheduling," *Games and Economic Behavior*, vol. 35, no. 1–2, pp. 271–303, 2001.

[30] 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. Syst., Man, Cybern.—Part C: Appl. Rev.*, vol. 39, no. 2, pp. 190–200, 2009.

Hoong Chuin Lau received the B.Sc. and M.Sc. degrees from the University of Minnesota, Minneapolis, in 1988 and the D.Eng. degree from the Tokyo Institute of Technology, Japan, in 1996.

He is an Associate Professor of Information Systems at the Singapore Management University (SMU) and a Director of The Logistics Institute Asia Pacific, National University of Singapore. He has published more than 100 research papers in journals and conferences. He is active in the development of a number of innovative tools and systems for industry, with applications to practical problems in transportation, logistics, supply chain management, defense, and e-Commerce. His research in the combined areas of artificial intelligence and operations research has been widely applied to decision support and optimization.

Dr. Lau received the Singapore National Innovation and Quality Circles Star Award in 2006 in recognition of his work and was awarded the Lee Kwan Yew Research Fellowship at SMU in 2008.

ZhengYi John Zhao received the B.Eng. and M.Eng. degrees from Tsinghua University, Beijing, China, in 1997 and 2001, respectively, the M.S. degree from SMA-HPCES, Singapore, in 2002, and the Ph.D. degree from the National University of Singapore.

He has six years experience in motion control scheduling and production planning at ASM-PT (*Advanced Semiconductor Material Pacific Tech.*), R&D Department, Singapore. He has two years working experience at the Singapore Management University as a Research Engineer. His research interest includes control, automation, and scheduling.

Shuzhi Sam Ge (S'90–M'92–SM'99–F'06) received the B.Sc. and Ph.D. degrees from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 1986 and the University of London, U.K., in 1993, respectively.

He is founding Director of the Social Robotics Laboratory of Interactive Digital Media Institute, and Professor of the Department of Electrical and Computer Engineering, National University of Singapore. He has (co)-authored three books: *Adaptive Neural Network Control of Robotic Manipulators*, *Stable Adaptive Neural Network Control and Switched Linear Systems*, and *Control and Design*. He has edited a book *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*. He has over 300 international journal and conference papers. He is the founding Editor-in-Chief of the *International Journal of Social Robotics*, Springer. He has served/been serving as an Associate Editor for a number of flagship journals. 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.

Tong Heng Lee (M'90) received the B.A. degree with (First Class Honors) in the engineering tripos from Cambridge University, Cambridge, U.K., in 1980 and the Ph.D. degree from Yale University, New Haven, CT, in 1987.

He is a Professor in the Department of Electrical and Computer Engineering at the National University of Singapore, as well as the current Head of the Control Engineering Section. His research interests are in the areas of adaptive systems, knowledge-based control, and intelligent mechatronics. He currently holds Associate Editor appointments in *Automatica*, the IEEE TRANSACTIONS IN SYSTEMS, MAN AND CYBERNETICS, *Control Engineering Practice* (an IFAC Journal), the *International Journal of Systems Science* (Taylor and Francis, London), and the *Mechatronics Journal* (Oxford, Pergamon Press).