

A Multi-Agent Model for Reactive Job Shop Scheduling

N. Liu, Mohamed A. Abdelrahman, and
Department of Electrical and Computer Eng.,
Tennessee Technological University,
Cookeville, TN 38505, USA

Srini Ramaswamy
Department of Computer Science,
Tennessee Technological University,
Cookeville, TN 38505, USA

Key Words: Reactive Job Shop Scheduling, Multi-Agent Systems, Scheduling Robustness

Abstract—Scheduling can be divided into two classes, namely, predictive and reactive scheduling. The former addresses problems with initially available jobs, deterministic processing times and available machines throughout the scheduling horizon; the latter has problems with random job arrivals, non-deterministic processing times and unpredictable events such as machine breakdowns. This paper presents a complete multiple agents' framework for reactive job shop scheduling. It provides a theoretical justification that this approach is actually a completely reactive scheduling approach combining real time decision making with predictive decision making and can resolve various disruptions as flexibly as dispatching rules. The justification also includes solution schemes for dynamic job arrivals, which make full use of available information for further scheduling robustness enhancement.

I. INTRODUCTION

Classical job shop scheduling involves a set of jobs and a set of machines. Each machine can handle at most one job at a time. Each job consists of a chain of operations to be processed in a specified sequence, on specified machines, and during an uninterrupted time period of given length. The purpose is to find a best schedule that is an allocation of the operations to time intervals on the machines and has the minimum duration required to complete all jobs. The total possible solutions for this problem with n jobs and m machines is $(n!)^m$. The problem is NP-hard and becomes even more complex when it includes other variables, such as dynamic situations.

Scheduling can be done in a static environment or in a dynamic environment. Scheduling done in a static environment is predictive scheduling, in which all jobs are available initially; process times are known and deterministic; and machines and other resources are available throughout the scheduling horizon. Scheduling done in a dynamic environment, such as in the common industrial setting, is reactive scheduling, in which jobs arrive at random over time; the processing times are not, in general, deterministic; there are always random and unpredictable events in the system, such as machine breakdowns and repairs, cancellation and due date changes of jobs [1]. Predictive scheduling usually produces schedules in advance in order to direct operations

and resource allocation. Unfortunately, in a dynamic environment, as soon as the schedule is released, it is immediately subject to random events, disruptions, which may make the initial schedule obsolete and result in the degeneration of the system performance [2].

Previous approaches to scheduling in the presence of uncertainties can be broadly classified into two groups. One group proposes job dispatching, which is completely reactive scheduling [3]; another group proposes control strategies that guide the recovery of the system from disruptions with consideration of a pre-computed schedule [2, 4, 5, 6]. In completely reactive scheduling, no schedule is generated in advance and decisions are made locally in real time. A frequently used reactive approach is priority dispatching rules, which dispatches jobs dynamically to an available machine according to their priorities to account for random disruptions as they occur. The priority dispatching rules are efficient but can bring poor long-term performance due to their local and myopic nature. However, all of traditional approaches are based on centralized models, in which a central computing unit performs all computations and views jobs and machines as passive symbols not as active entities like in certain decentralized models. This suggests that centralized approaches are inflexible, expensive, and slow to satisfy scheduling problems under disruptions [7].

Scheduling approaches based on decentralized models that view jobs and machines as active entities—agents are flexible and quick to satisfy real-world scheduling problems. Previous approaches are mainly on the combination of market-based mechanisms used by the multi-agent system, such as contract net protocol [8] or auction, with local decision schemes used by agents, such as dispatching rules, constraint satisfaction, and decision trees [9, 10, 11, 12]. Recently, the research has evolved into the combination of auction with Lagrangian relaxation [13, 14, 15, 16]. This progress provides the theoretical basis for bid computation, bid evaluation and price adjustment; considers optimization during distributed problem solving; and overcomes the performance unpredictability and schedules' poor quality of early approaches [16]. Auction/Lagrangian relaxation approaches have been used in quasi-distributed job shop environments [13, 14] and a holonic system [15]. A variant has been used in a distributed job shop environment [16] and a single machine with dynamic job arrivals [17].

The goal of this research is to explore a methodology that uses distributed problem solvers — multiple agents to effectively and efficiently schedule job shops in a dynamic environment with minimum global information and without master/slave relationships between agents, and considers not only real time decision-making but also optimization. This paper presents a complete multiple agents' framework for reactive job shop scheduling, which extends the variant in [16] with a rolling time horizon procedure. It provides a theoretical justification that this approach is actually a completely reactive scheduling approach combining real time decision making with predictive decision making and can resolve various disruptions as flexibly as dispatching rules. The justification also includes solution schemes for dynamic job arrivals, which make full use of available information for further scheduling robustness enhancement.

The paper is organized as follows. Section II presents the mathematical formulation of job shop scheduling and resolution of its Lagrangian relaxation problem. Section III describes the complete multiple agents' framework. Section IV provides solution schemes for dynamic job arrivals. Section V is a summary.

II. PROBLEM FORMULATION AND RESOLUTION

An integer programming formulation is a common way to represent a scheduling problem. The following discrete-time, integer-programming formulation of a deterministic job shop scheduling problem with weighted tardiness objective is due to [16].

First, the following variables will be defined, where operation j of job i is referred to as operation (i, j) .

T	time horizon of scheduling
N	total number of jobs
M	total number of machines
r_i	ready time of job i
p_{ij}	processing time of operation (i, j)
d_i	due date of job i
w_i	weight of job i
C_{ij}	completion time of operation (i, j)
T_i	tardiness of job i , $T_i = \max \{0, C_{ij} - d_i\}$
O_i	total number of operations of job i
X_{ijt}	0-1 interger variable equals one if operation (i, j) is completed at time t
Y_{ijm}	0-1 integer variable equals one if operation (i, j) is processed on machine m
λ_{mt}	Lagrange multiplier
UB	upper bound of Lagrangian relaxation problem
LB_r	lower bound of iteration r of Lagrangian relaxation problem
Sr	step size of iteration r
SG^r_{mt}	subgradient of iteration r
α_r	subgradient multiplier of iteration r , $0 < \alpha_r \leq 2$

Then, the integer programming formulation of the job shop problem is defined as follows:

$$(P) \quad \min \sum_{i=1}^N \sum_{t=1}^T w_i T_i X_{iO_t}$$

s.t.

$$\sum_{t=1}^T X_{ijt} = 1, \quad \forall i, j \quad (1)$$

$$\sum_{t=1}^T t X_{ijt} + p_{i,j+1} \leq \sum_{t=1}^T t X_{i,j+1,t}, \quad \forall i, j \quad (2)$$

$$\sum_{i=1}^N \sum_{j=1}^{O_i} X_{ijt} Y_{ijm} + \sum_{i=1}^N \sum_{j=1}^{O_i} \sum_{t'=t-1}^{t-p_{ij}+1} X_{ijt'} Y_{ijm} \leq 1, \quad \forall m, t \quad (3)$$

$$\sum_{t=1}^T t X_{it} \geq p_{i1} + r_i, \quad \forall i \quad (4)$$

$$X_{ijt} \in \{0, 1\}, \quad \forall i, j, t$$

$$Y_{ijm} \in \{0, 1\}, \quad \forall i, j, m.$$

where constraints (2) are operation precedence constraints; constraints (3) are machine capacity constraints; and constraints (4) are job ready time constraints.

When the machine capacity constraints (3) are relaxed by using Lagrange multipliers λ_{mt} , the following Lagrangian relaxation problem is obtained:

$$(P_\lambda) \quad \min \sum_{i=1}^N \sum_{t=1}^T \left[w_i T_i X_{iO_t} + \sum_{m=1}^M \lambda_{mt} \sum_{j=1}^{O_i} \left(X_{ijt} Y_{ijm} + \sum_{t'=t-1}^{t-p_{ij}+1} X_{ijt'} Y_{ijm} \right) \right] - \sum_{t=1}^T \sum_{m=1}^M \lambda_{mt}$$

s.t. (1), (2), and (4).

The relaxed problem can be decomposed into the following independent job-level subproblems for the given set of multipliers:

$$(P_{\lambda_i}) \quad \min \sum_{t=1}^T \left[w_i T_i X_{iO_t} + \sum_{m=1}^M \lambda_{mt} \sum_{j=1}^{O_i} \left(X_{ijt} Y_{ijm} + \sum_{t'=t-1}^{t-p_{ij}+1} X_{ijt'} Y_{ijm} \right) \right]$$

s.t.

$$\sum_{t=1}^T X_{ijt} = 1, \quad \forall j \quad (5)$$

$$\sum_{t=1}^T t X_{ijt} + p_{i,j+1} \leq \sum_{t=1}^T t X_{i,j+1,t}, \quad \forall j \quad (6)$$

$$\sum_{t=1}^T t X_{it} \geq p_{i1} + r_i \quad (7)$$

$$X_{ijt} \in \{0, 1\}, \quad \forall j, t$$

$$Y_{ijm} \in \{0, 1\}, \quad \forall j, m.$$

Then, the dual problem is obtained as follows, where $V(P)$ denotes the value of optimal solution of problem P :

$$(P_D) \quad \max \left\{ \min \left[\sum_{i=1}^N V(P_{\lambda_i}) - \sum_{t=1}^T \sum_{m=1}^M \lambda_{mt} \right] \right\}$$

s.t. $\lambda \geq 0$.

The dual problem can be solved by subgradient search algorithm, which iteratively determine optimal λ_{mt} given X_{ijt} according to the following formulas:

$$\lambda_{mt}^{r+1} = \max\{0, \lambda_{mt}^r + S_r \cdot SG_{mt}^r\}$$

$$SG_{mt}^r = \left\{ \sum_{i=1}^N \sum_{j=1}^{O_i} X_{ijt} Y_{ijm} + \sum_{i=1}^N \sum_{j=1}^{O_i} \sum_{t'=t-1}^{t-p_{ij}+1} X_{ijt'} Y_{ijm} - 1 \right\},$$

$$S_r = \alpha_r \cdot \left[\frac{UB - LB_r}{\sum_{m=1}^M \sum_{t=1}^T (SG_{mt}^r)^2} \right],$$

$$LB_r = \sum_{i=1}^N V(P_{\lambda_i}) - \sum_{t=1}^T \sum_{m=1}^M \lambda_{mt},$$

where stopping criteria could be one of the following: $UB - LB_r < \epsilon$; α_r becomes very small; or $r >$ Maximum number of iterations.

III. THE COMPLETE MULTI-AGENT FRAMEWORK

When a Lagrange multiplier λ_{mt} denotes the price of using a time slot t on machine m , the subgradient search procedure can be transferred into a combinatorial auction, in which each job and machine is an agent; an auctioneer proposes prices for using each time slot of machines to maximize their profits; and bidders construct their bids of the wanted time slots to minimize their costs (see the left architecture in Figure 1).

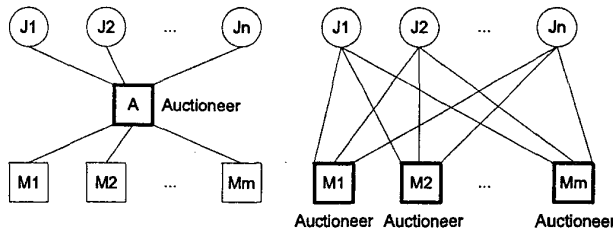


Figure 1. System architecture

For a given set of prices λ_{mt} , every job agent, i.e. a bidder, constructs its bids X_{ijt} from its job-level independent subproblem $P_{i,t}$, which is obtained from the decomposition of Lagrangian relaxed problem P_i . The coordination of bidders and the auctioneer is performed through the iterative adjustment of these prices according to $\lambda_{mt}^{r+1} = \max\{0, \lambda_{mt}^r + S_r \cdot SG_{mt}^r\}$ based on the degrees of machine capacity constraint violation and following the market economy concept, i.e. increasing prices for over-utilized time slots and reducing prices for under-utilized slots.

The auction/Lagrangian relaxation approaches have two very different types of multi-agent system architecture (Figure 1). In the left architecture [13, 14, 15], there is only one auctioneer representing all machine agents, in which

scheduling can only be considered in a static environment. In the right architecture [16], there are multiple auctioneers; every machine agent becomes an auctioneer when it needs, in which scheduling can be done in a dynamic environment. This is a real decentralized market organization [18]. The complete multi-agent framework uses this architecture.

The common shortcoming of combinatorial auction with Lagrangean relaxation approaches is the use of a constant time horizon T . The length of time horizon has to be long enough to cover the whole problem size while not to make the problem computationally intractable. It has been shown that increasing the length of time horizon increases not only computational times but also network communication delays when being implemented in a distributed network. Rolling time horizon procedures have been applied to a wide range of dynamic optimization problems [19]. It optimizes the system in a limited time horizon instead of globally optimizing the system. The complete multi-agent framework is based on a rolling time horizon procedure.

Additional variables are defined as follows:

- L length of rolling time horizon
- t_c time of current decision point
- N_c current total number of job agents
- M_c current total number of machine agents

The complete multi-agent framework is described as follows (Figure 2):

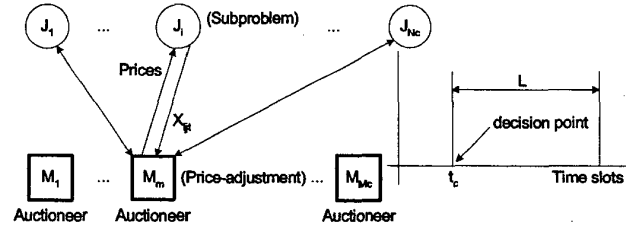


Figure 2. Complete multi-agent framework

(Subproblem) =

$$\min \sum_{t=t_c}^{t_c+L} \left[w_i T_i X_{i0t} + \sum_{m=1}^{M_c} \lambda_{mt} \sum_{j=1}^{O_i} \left(X_{ijt} Y_{ijm} + \sum_{t'=t-1}^{t-p_{ij}+1} X_{ijt'} Y_{ijm} \right) \right]$$

s.t.

$$\sum_{t=t_c}^{t_c+L} X_{ijt} = 1, \quad \forall j \quad (8)$$

$$\sum_{t=t_c}^{t_c+L} t X_{ijt} + p_{i,j+1} \leq \sum_{t=t_c}^{t_c+L} t X_{i,j+1,t}, \quad \forall j \quad (9)$$

$$\sum_{t=t_c}^{t_c+L} t X_{i1t} \geq p_{i1} + r_i \quad (10)$$

(Price-adjustment) = $\lambda_{mt}^{r+1} = \max\{0, \lambda_{mt}^r + S_r \cdot SG_{mt}^r\}$

$$SG_{mt}^r = \left\{ \sum_{i=1}^{N_c} \sum_{j=1}^{O_i} X_{ijt} Y_{ijm} + \sum_{i=1}^{N_c} \sum_{j=1}^{O_i} \sum_{t'=t-1}^{t-p_{ij}+1} X_{ijt'} Y_{ijm} - 1 \right\},$$

$$S_r = \alpha_r \cdot \left[\frac{UB - LB_r}{\sum_{m=1}^{M_c} \sum_{t=t_c}^{t_c+L} (SG_{mt}^r)^2} \right]$$

$$LB_r = \sum_{i=1}^{N_c} V(P_{\lambda_i}) - \sum_{t=t_c}^{t_c+L} \sum_{m=1}^{M_c} \lambda_{mt},$$

where m is from 1 to M_c and t is from t_c to t_c+L .

Each machine agent is also an auctioneer and each job agent is a bidder. When a machine becomes available, it is at decision points t_c to select an operation of a job to process. At every decision point t_c , the machine agent becomes an auctioneer and hosts a combinatorial auction with proposed prices λ_{mt} of using each time slot of all machines. All bidders construct their bids of the wanted time slots X_{ijt} according to (*Subproblem*). Coordination of bidders and the auctioneer is performed through the iterative adjustment of these prices according to (*Price-adjustment*). At the termination of such price adjusting iterations, the auctioneer selects a bid having t_c as a start time, otherwise no bid winner and the machine agent will start another auction at next decision point t_c+1 .

This is a job patching procedure, a completely reactive scheduling approach. The solution of one auction provides a ranking of operations processed on a machine so that every machine agent can construct its schedule locally and in real time, and at the same time have predictive decision making of time horizon L based on mathematical programming. All machine agent's auctions form simultaneous and repeated combinatorial auctions.

However, the advantage of this framework is its ability to handle the dynamic nature of job shop quite effectively. A new job arrival for processing within the given time horizon implies that the job agent can just start participating in next auctions announced by machine agents; processing time variations mean that the machine agent advances or postpones hosting auctions; a machine breakdown means that this machine agent stops hosting auctions until the time it is repaired; if rerouting is allowed to job agents, they can begin bidding on their second choice machine when the first choice machine fails. Under the framework, agents can enter or exit scheduling system flexibly without bothering others so that various unpredictable disruptions can be resolved as flexibly as dispatching rules to achieve scheduling robustness.

IV. SOLUTION SCHEMES FOR JOB ARRIVALS

A future scheduling research direction that needed to understand schedule robustness through the explicit consideration of uncertain available information was suggested in 1980s [20]. Some research has been done [2, 5, 21] to explicitly use uncertain available information when dealing with disruptions. By robustness it is meant that the performance of the schedule still remains satisfactory in the presence of disruptions.

In the problem formulation, there are variables r_i representing ready times of jobs for processing. If there are forecasts of jobs ready for processing within a given time horizon, pseudo job agents can be created to represent the forecasts of arriving jobs. These pseudo job agents can participate in following auctions to make predictive decision making better. With the same lengths of rolling time horizon, performance of this dynamic model will largely be determined by the goodness of forecasts. This is a procedure that combines a degree of forecast at each decision point with a predictive decision making procedure. An extreme case of such a procedure, with no forecast, is a procedure with unpredictable dynamic job arrivals. Another extreme, with perfect forecast, is a procedure that will have best predictive decisions. How about the case using partial perfect forecast with partial without forecast (i.e. partial unpredictable job arrivals) or using good forecast with stochastic information?

Therefore, there are four solution schemes for dynamic job arrivals, which make full use of uncertain available information as possibly as they can to further enhance scheduling robustness. The four solution schemes include unpredictable job arrivals (Figure 3), perfect forecast of job arrivals (Figure 4), partial unpredictable job arrivals with partial perfect forecast of job arrivals (Figure 5), and good forecast of job arrivals (Figure 6).

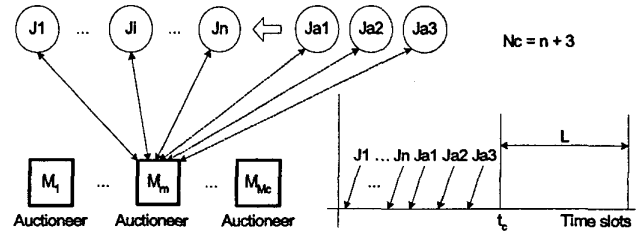


Figure 3. Unpredictable job arrivals

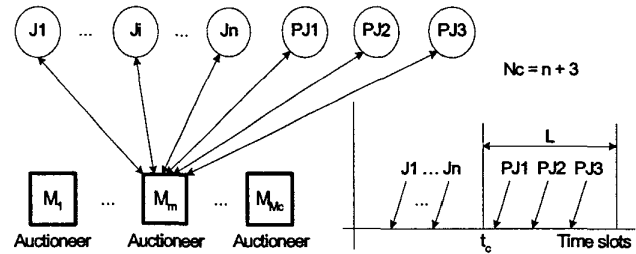


Figure 4. Perfect forecast of job arrivals

In Figures 3, 4, 5 and 6, already existing jobs J_i and just arrived jobs J_{ai} form job agents. Expecting arrived jobs PJ_i within time horizon L form pseudo job agents. All of them participate in auctions beginning at decision point t_c .

In the case of good forecast of job arrivals with stochastic information (Figure 6), variables of job ready times r_i are viewed as stochastic variables. Constraints (10) become probabilistic constraints (11), which means that constraints are not expected to be always satisfied, but only satisfied with

given probabilities (see the following mathematical formulas). Using chance constrained programming [22], the probabilistic constraints (11) can be converted into their respective deterministic equivalents (12) according to the predetermined confidence level α_i . In the deterministic constraints (12), expected job ready times r_i are replaced with their anticipated job ready times R_i that are in terms of means and variances of stochastic variables r_i . Thus, there is a parameter value defining the degree of available information of variances. More variance information will make a better forecast.

$$\text{Prob} \left[\sum_{t=t_c}^{t_c+L} tX_{it} - p_{it} \geq r_i \right] \geq \alpha_i, \quad 0 \leq \alpha_i \leq 1 \quad (11)$$

$$\sum_{t=t_c}^{t_c+L} tX_{it} \geq p_{it} + R_i \quad (12)$$

$$R_i = F^{-1}(\alpha_i)$$

$$\text{Prob}[r_i \leq z] = F(z)$$

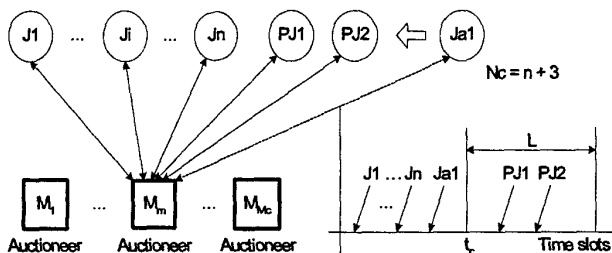


Figure 5. Partial unpredictable arrivals and partial perfect forecast

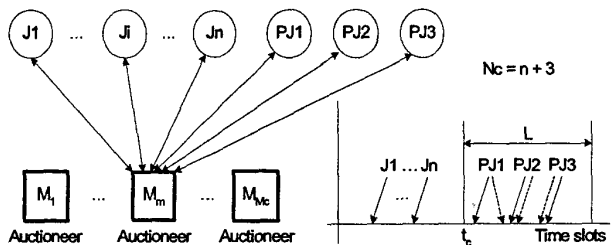


Figure 6. Good forecast of job arrivals

V. SUMMARY

In this paper, we provide a theoretical justification of the arguments about a complete multiple agents' framework for reactive job shop scheduling and solution schemes for dynamic job arrivals, which make full use of available information as possibly as they can to further enhance scheduling robustness. Currently we are performing computational experiments for an experimental justification of these arguments.

VI. ACKNOWLEDGEMENT

This study was supported by Center for Manufacturing Research, Tennessee Technological University.

VII. REFERENCES

- [1] V. Suresh and Dipak Chaudhuri, "Dynamic scheduling-a survey of research," *International Journal of Production Economics*, vol. 32, pp. 53-63, 1993.
- [2] V. Jorge Leon, S. David Wu, and Robert H. Storer, "Robustness measures and robust scheduling for job shops," *IIE Transactions*, vol. 26, no. 5, pp. 32-43, September 1994.
- [3] K. Bhaskaran and M. Pinedo, "Dispatching," in G. Salvendy (ed.), *Handbook of Industrial Engineering*, Wiley, 1991, chapter 83.
- [4] M. Yamamoto and S. Y. Nof, "Scheduling/rescheduling in the manufacturing operating system environment," *International Journal of Production Research*, vol. 23, no. 4, pp. 705-722, 1985.
- [5] Sanjay V. Mehta and Reha M. Uzsoy, "Predictable scheduling of a job shop subject to breakdowns," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 365-378, June 1998.
- [6] Elizabeth Szelke and Roger M. Kerr, "Knowledge-based reactive scheduling," *Production Planning & Control*, vol. 5, no. 2, pp. 124-145, 1994.
- [7] Weiming Shen, "Distributed manufacturing scheduling using intelligent agents," *IEEE Intelligent Systems and Their Applications*, vol. 17, no. 1, pp. 88-94, January/February, 2002.
- [8] R. G. Smith, "The contract net protocol: high-level communication and control in distributed problem solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104-1113, Dec. 1980.
- [9] Michael J. Shaw, "A distributed scheduling method for computed integrated manufacturing: the use of local area networks in cellular systems," *International Journal of Production Research*, vol. 25, no. 9, pp. 1285-1303, 1987.
- [10] Grace Yuh-Jiun Lin and James J. Solberg, "Integrated shop floor control using autonomous agents," *IIE Transactions*, vol. 24, no. 3, pp. 57-71, July 1992.
- [11] T-Tung Dang and B. Frankovic, "Agent-based scheduling in production systems," *International Journal of Production Research*, vol. 40, no. 15, pp. 3669-3679, 2002.
- [12] J. Váncza and A. Márkus, "An agent model for incentive-based production scheduling," *Computers in Industry*, vol. 43, no. 2, 2000.
- [13] Sanjay E. Ramaswamy, *Distributed Control of Automated Manufacturing Systems*, Ph.D. Dissertation, Pennsylvania State University, 1995.
- [14] Erhan Kutanoglu and S. David Wu, "On combinatorial auction and Lagrangean relaxation for distributed resource scheduling," *IIE Transactions*, vol. 31, no. 9, pp. 813-826, 1999.
- [15] Ling Gou, Peter B. Luh, and Yuji Kyoya, "Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation," *Computers in Industry*, vol. 37, pp. 213-231, 1998.
- [16] Pooja Dewan and Sanjay Joshi, "Auction-based distributed scheduling in a dynamic job shop environment," *International Journal of Production Research*, vol. 40, no. 5, pp. 1173-1191, 2002.
- [17] Pooja Dewan and Sanjay Joshi, "Dynamic single-machine scheduling under distributed decision-making," *International Journal of Production Research*, vol. 38, no. 16, pp. 3759-3777, 2000.
- [18] Thomas W. Malone and Stephen A. Smith, "Modeling the performance of organizational structures," *Operations Research*, vol. 36, no. 3, pp. 421-435, May-June 1988.
- [19] Thomas E. Morton, "Forward algorithms for forward-thinking managers," in Randall L. Schultz (ed.), *Applications of Management Science*, vol. 1, JAI Press Inc., 1981, pp. 1-55.
- [20] Stephen C. Graves, "A review of production scheduling," *Operations Research*, vol. 29, pp. 646-675, 1981.
- [21] V. J. Leon, S. D. Wu, and R. H. Storer, "A game-theoretic control approach for job shops in the presence of disruptions," *International Journal of Production Research*, vol. 32, no. 6, pp. 1451-1476, 1994.
- [22] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management Science*, vol. 6, pp. 73-79, 1960.