A Performance Study on a Multiagent E-Scheduling and Coordination Framework for Maintenance Networks

Feng Zhang, Eugene Santos, Jr., *Senior Member, IEEE*, and Peter B. Luh, *Fellow, IEEE*

***Abstract*—Many maintenance networks as well as supply networks and virtual enterprises consist of multiple organizations. A common problem arising in these different domains is multiorganization scheduling and coordination. The traditional centralized methods are not appropriate because of the existence of private information and decision-making authorities at different organizations. Although many distributed mechanisms have been presented for supply networks and virtual enterprises, they may not be effective for maintenance networks because of the difficulties of scheduling the tightly related maintenance operations and handling the massive uncertainties involved. These difficulties as well as the heterogeneity of the distributed environment make it challenging to develop an efficient framework for maintenance networks that can obtain a high-quality solution under different conditions, schedule in a timely manner, solve large-scale problems, and so on. In this paper, a price-based multiagent scheduling and coordination framework for maintenance networks is explored, and a systematic experimental study is carried out to evaluate the effects of different factors on its performance. The results show that the framework is able to overcome these difficulties and could be a step toward the next generation of e-scheduling for maintenance networks.**

***Index Terms*—Lagrangian relaxation, maintenance networks, multiagent, multiorganization scheduling and coordination.**

I. Introduction

ASSET MAINTENANCE is needed almost everywhere, from everyday life to production and service industries. The reason is that the assets, such as automobiles, jet engines of airlines, and generators of electric utilities, become worn out over time. The role of maintenance networks is to conduct a remanufacturing process in which the worn-out assets are restored to a like-new condition through a series of disassembly, cleaning, repair, and assembly operations with the infusion of new parts as necessary. Similar to supply networks and virtual enterprises, many maintenance networks consist of multiple organizations, including overhaul centers (disassembling the worn-out assets into parts for repair and reassembling the parts after repair), repair shops (repairing the parts), warehouses of rotable inventory (storing the parts usable by multiple assets), spare part distributors, and manufacturers. In this paper, the focus is on maintenance of jet engines, but the work can also be applied to other asset maintenance networks. A simplified asset maintenance process is shown in Fig. 1. The worn-out assets with various priorities arrive at the overhaul center and are disassembled into serial-number-specific parts (required to be assembled into the original assets they belong to) and rotable parts (satisfying certain qualification for general use). After inspection, good parts are ready for reassembly, while the repairable parts are delivered to the repair shop and the remaining parts are scrapped. For simplicity, scrapping a part is supposed to trigger the ordering of a new part with an uncertain lead time. Hence, the warehouse of rotable inventory, which stores good as well as repaired rotable parts, has more or less a constant number of rotable parts over time. Since repair of rotable parts may take a long time, it is important to have some extra rotable parts initially (i.e., a small nonzero initial inventory level) for reducing asset turnaround times without incurring a high inventory cost. In the view that ordering a new part can be modeled similarly as a repair operation, it is assumed here that all the parts can be repaired. Assembly can start when the required parts are ready. The multiorganization maintenance scheduling problem is to find a scheduling policy to arrange the asset overhaul and part repair operations with the objective of minimizing mean asset turnaround time and mean inventory cost (INVC), subject to intraorganization constraints such as resource capacity constraints and interorganization constraints such as disassembly/repair precedence constraints [1].

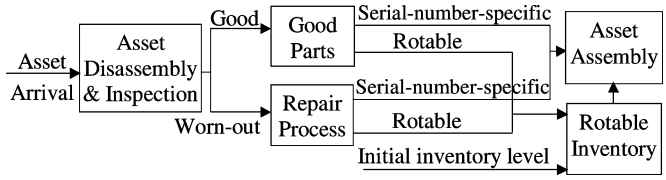

Fig. 1. Maintenance process illustration.

The multiorganization maintenance scheduling and coordination problem can hardly be solved optimally since scheduling in general is NP-hard. The goal, therefore, is to identify an approach that can obtain a nearly optimal solution efficiently. The following difficulties, however, make the maintenance scheduling problem more difficult to solve. First, the maintenance processes are generally characterized by massive uncertainties, including the uncertain asset arrivals and operation processing times. These uncertainties can lead to unpredictable asset turnaround times. Furthermore, the large

Manuscript received March 28, 2005; revised August 9, 2005. This work was supported in part by the National Science Foundation under Grant DMI-0223443. This paper was recommended by Associate Editor R. Brennan.

F. Zhang is with the Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269 USA (e-mail: fzhang@cse.uconn.edu).

E. Santos is with the Thayer School of Engineering, Dartmouth College, Hanover, NH 03755 USA (e-mail: Eugene.Santos.Jr@dartmouth.edu).

P. B. Luh is with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269 USA, and also with the Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: luh@engr.uconn.edu).

Digital Object Identifier 10.1109/TSMCC.2006.876057

number of maintenance operations are tightly related because most of the repaired parts (excluding parts in the warehouses of rotable inventory) have to be assembled back into assets. Such tight relations make it difficult for the organizations with their own sensitive information and decision-making authorities to coordinate with each other for finding a high-quality solution. Finally, the computing resources of the organizations are generally connected through the Internet. The Internet, with various processor speeds and communication delays, imposes restrictions on implementing multiorganization scheduling and coordination. For example, a synchronous coordination scheme will be either slow or hardly scalable because of its potentially high-synchronization overhead.

To overcome the difficulties, a price-based scheduling method, which considers the massive uncertainties involved, was presented in [1]. Its key idea is decomposition and coordination in that it employs Lagrangian relaxation to decompose the original problem into a set of subproblems, which are solved by using stochastic dynamic programming given the current prices (i.e., the Lagrange multipliers). With the current subproblem solutions available, the prices are updated at the upper level by using a surrogate subgradient method: increase prices if the corresponding constraints are violated, and reduce prices otherwise. Under the new prices, the previous solutions may not be cost effective, and new solutions need to be computed. Through iterative updating of both the prices and subproblem solutions, the original problem can be solved. The method was shown to be nearly optimal using a centralized setting. A challenge of applying the method is how to efficiently deploy it in a distributed environment while facilitating organization autonomy. Efficient deployment means that the performance of the distributed approach should be close to that of the centralized implementation. Since many factors may affect its performance, it is essential to systematically show how it works in a distributed environment under different conditions, such as with sufficient or insufficient resources, nonzero or zero initial inventory level, certain or uncertain information, etc.

In this paper, a multiagent scheduling and coordination framework built on the price-based scheduling method is developed to solve the multiorganization maintenance scheduling problem. The use of agents makes it possible to protect private information and respect the decision-making authorities of the organizations. Our first step is to decompose the problem into several organization-level problems. Because of complexity, each organization-level problem is further decomposed into smaller asset-level/part-level subproblems. Next, an agent is associated with each individual organization, asset, and part to solve the corresponding subproblem. Each agent will communicate with the relevant agents for exchanging nonprivate information, including subproblem solution and new prices. While the framework facilitates organization autonomy, is it efficient? Specifically, can the framework find a high-quality solution under different conditions? Can it schedule in a timely manner? Is the framework scalable?

To answer the above-mentioned questions, a systematic experimental study is carried out by evaluating the impact of selected factors on the performance of the agent framework.

1) A solution framework factor is selected to examine whether the performance of the agent scheduling framework can approach that of the centralized implementation or not. Moreover, the following four factors that reflect the different conditions arising in maintenance scheduling are chosen.
    a) Initial inventory relative level and holding cost are the two factors related to rotable inventory management. Initial inventory relative level, a ratio of initial inventory level to the total number of worn-out assets, is used to reflect the conditions with nonempty or empty initial inventory and holding cost used to reflect the conditions with high or low inventory management cost.
    b) Targeted utilization of resources of a maintenance resource type, an estimated ratio of their busy time to their total available time, is used to reflect the conditions with sufficient or insufficient resources.
    c) Uncertainty level is used to capture the uncertain asset arrivals and operation processing times.
2) To determine the timeliness of scheduling in a distributed environment, two factors, processor speed and communication delay, are selected.
3) To examine whether the framework is scalable, the number of assets is varied to reflect different problem scales.

Two orthogonal arrays are applied to systematically conduct the experiments based on the selected factors and reduce the total experimental cost. In the following, a literature review is first given. A brief presentation of the problem formulation and the price-based scheduling method follows. A multiagent e-scheduling and coordination framework is described in Section IV. In Section V, the experimental design is presented in detail. The experimental results in Section VI show that our framework is able to overcome the major difficulties. Built on the price-based scheduling method, it can obtain a high-quality solution quickly in most cases, even for large-scale problems. More importantly, relations between the selected factors and the performance measures are drawn from the results. These results can provide organizations with guidance on setting up a suitable amount of resources and initial inventory level. From these results, it is concluded that the framework performs well under the different conditions in a distributed environment and could be a step toward the next generation of e-scheduling for maintenance networks.

II. Related Work

Remanufacturing and the associated repairable inventory theory have been summarized in [2] and [3]. It was concluded that a specifically designed framework and the corresponding models were lacking for the production planning and control of remanufacturing systems. In addition, a critical feature of remanufacturing, i.e., the presence of rotable inventory, has not been well addressed, although the value of rotable inventory to coordinate material flows has been identified [4]. In the past, excessive preventive maintenance has been mainly used to maximize asset availability by industries such as the power industry. The downside of excessive maintenance is unnecessary and

increased maintenance cost. To reduce the cost while maximizing asset availability, condition-based maintenance and repair has emerged [5], [6], which can often detect the potential problems early based on the historic asset condition information so that the necessary maintenance can be done less expensively. While the issues of collecting and using condition information have been addressed by the current generation of e-scheduling frameworks, organization autonomy and performance optimization have not received enough attention. Although the work in supply networks [7], [8] and virtual enterprises [9] generally consider organization autonomy, they may not be effective for maintenance networks because of the tightly related maintenance operations and the massive uncertainties involved. In our view, the next generation of e-scheduling frameworks will not only seek the needed information actively, but also optimize performance actively while facilitating autonomy.

Multiagent systems (MASs) have been studied intensively to tackle many complex and distributed application domains, such as e-commerce [10], logistics management [11], weather forecasting [12], and network management [13]. Generally, a MAS consists of a set of software agents that operate continuously and autonomously with their own threads of control in a distributed environment. These agents try to achieve their goals in an intelligent fashion by planning and targeting the best action in a dynamic environment. They can either work on their own or cooperate with each other. By employing multiple agents, MASs can achieve distributed decision-making and site autonomy, support complicated cooperation among the different entities, provide platforms for deploying the distributed scheduling methods, and efficiently utilize the available distributed computing resources. Of course, there are still several concerns currently on using MASs, such as malicious agents, interoperability between two different agent platforms, and lacking of inherent methodologies for designing and developing MASs. But, with more and more work focusing on these concerns, it is expected that MASs will be widely used.

A variety of agent-based coordination models have been presented in the past [10], [14]. Some models are based on the contract net protocol [9], [15], [16], which is simple but not good for iterative optimization in the view that it is very difficult to change the commitments made previously, while several other models, such as generalized partial global planning [17], distribute planning capabilities to all the agents and employ an extendable set of coordination mechanisms for more generic applicability. In the case of unreliable and low-bandwidth communication, a coordination protocol was presented in [18] to allow the agents to choose actions locally based on the predetermined strategies, which can be decided offline or formed during periods of full communication. This protocol may not be feasible for maintenance networks because it is often hard to identify the effective strategies beforehand in maintenance scheduling. A rule-based coordination infrastructure was presented in [9] for virtual enterprise systems, and information resource heterogeneity was addressed while impact of heterogeneous processor speeds and communication delays on system performance was not. Several approaches investigated optimal scheduling and coordination in different domains [7].

MASs have also been applied in solving maintenance problems [6], [19], [20]. A multiagent framework was developed in [19] based on the RETSINA agent architecture. In this framework, the role of the agents is to search for information that can assist the mechanics and engineers to identify the problems and select appropriate repair methods in a short time. A multiagent scheduling model based on mixed-integer programming was presented in [20] to solve a deterministic power equipment maintenance scheduling problem with the objective of minimizing maintenance cost and revenue loss.

In view of the major difficulties encountered in developing the next generation of e-scheduling for maintenance networks, few of the past work are feasible. Therefore, new models need to be developed to overcome the difficulties. In Section III, the problem formulation and the price-based scheduling method are briefly described. After that, a price-based multiagent scheduling and coordination framework, which could be a step toward the next generation of e-scheduling for maintenance networks, is presented.

III. Problem Formulation and Price-Based Scheduling Method

This section briefly summarizes the problem formulation and solution methodology presented in [1]. For the simplicity of the discussions in Sections III-A and B, assume one overhaul center, one repair shop, and one warehouse of rotable inventory. It is straightforward to extend the formulation to the case with multiple overhaul centers, repair shops, and/or warehouses of rotable inventory.

A. Problem Formulation

The multiorganization maintenance scheduling problem is subject to two categories of constraints.

1) *Intraorganization constraints:* The major constraints for the overhaul center or the repair shop include the following.
   a) *Operation processing constraints:* Each maintenance operation has to be processed by some resources for some amount of time.
   b) *Precedence constraints between adjacent operations:* Each succeeding operation cannot be started until the current operation is completed plus a required timeout (e.g., transportation time).
   c) *Arrival time constraints:* Disassembly of each asset can be started no earlier than its arrival plus a required wait time. These constraints are considered for the overhaul center.
   d) *Expected resource capacity constraints:* The expected number of active maintenance operations using the same type of resources cannot exceed the available resources at any time. Since these constraints are relaxed in the solution methodology described in Section III-B, their formulations are shown here. Let $(a; j)$ denote the jth overhaul operation of asset a. For simplicity, let $(a; 1)$ represent the aggregate disassembly operation and $(a; 2)$ the

aggregate assembly operation. The time horizon and the number of type h resources available at time k are denoted by K and M_{kh}. Let δ_{ajkh} be a binary variable defined to be one if the operation (a, j) is active at time k on the resource type h and zero otherwise. Similarly, let (a, i), J_{ai}, and $(a, i; j)$ denote part i of asset a, the total number of repair operations, and the jth operation of $(a, i)(j \in [1, J_{ai}])$. A binary variable δ_{aijkh} is defined for the operation $(a, i; j)$. Using the notation, the resource capacity constraints for the overhaul center (1) and for the repair shop (2) are

$$E[\sum_{aj} \delta_{ajkh}] \leq M_{kh}, \quad \forall k, \quad \text{and} \quad h \tag{1}$$

$$E[\sum_{aij} \delta_{aijkh}] \leq M_{kh}, \quad \forall k, \quad \text{and} \quad h. \tag{2}$$

The major constraints for the warehouse of rotable inventory are inventory dynamics. The inventory dynamics constraints describe that the number of parts of a rotable type r in the warehouse at the current time slot k, denoted by $I_r(k)$, is equal to the available parts at the previous slot plus the parts repaired in the previous slot and minus the parts used for assembly starting at the current slot. Let $\beta^r_{aiJ_{ai}k}$ be one if (a, i) is a rotable part of type r and its last operation $(a, i; J_{ai})$ completes at time $k-1$ and zero otherwise. Similarly, α^r_{a2k} indicates the beginning of the assembly operation $(a; 2)$ if a rotable part of type r is used in assembly, and is set to one if the operation $(a; 2)$ starts at time k and zero otherwise. The initial inventory level of type r at the time slot 0 is given, denoted by $I_{r0}[= I_r(0)]$. The inventory dynamics constraints for type r can be formulated as follows $(k \geq 1)$:

$$\begin{aligned} I_r(k) &= I_r(k-1) + \sum_{ai} \beta^r_{aiJ_{ai}k} - \sum_{a} \alpha^r_{a2k} \\ &= I_{r0} + \sum_{ai} \left[\sum_{\mathcal{K}=1}^{k} \beta^r_{aiJ_{ai}\mathcal{K}}\right] - \sum_{a} \left[\sum_{\mathcal{K}=1}^{k} \alpha^r_{a2\mathcal{K}}\right]. \end{aligned}$$

Since the increase and decrease of inventory level are affected by rotable part repair and asset assembly that are conducted by two different organizations, the nonnegative inventory-level constraints are classified into interorganization constraints.

2) *Interorganization constraints*: The interorganization precedence constraints, including disassembly/repair precedence constraints and repair/assembly precedence constraints, exist between the overhaul center and the repair shop. Let b_{aj}, p_{aj}, c_{aj}, and s_{aj} represent the beginning times, processing times, completion times, and the timeout values of the disassembly/assembly operations. Similarly, define $b_{aij}, p_{aij}, c_{aij}$, and s_{aij} for the repair operations. The disassembly/repair constraints (3) reflect that part repair cannot start until disassembly is done plus a required timeout (i.e., s_{a1}), while the repair/assembly constraints (4) depict that assembly has to wait till serial-number-specific (SNS) part repair is finished plus a required timeout. The nonnegative inventory-level constraints (5), existing between the warehouse of rotable inventory and the overhaul center/repair shop, state that the expected number of rotable parts in the inventory cannot be below zero. If the initial inventory level of rotable part type r is high, the corresponding constraints tend to be violated less frequently during scheduling than the case with low or zero initial inventory level

$$E[b_{a1} + p_{a1} + s_{a1} - b_{ai1}] \leq 0, \quad \forall (a, i) \tag{3}$$

$$E[b_{aiJ_{ai}} + p_{aiJ_{ai}} + s_{aiJ_{ai}} - b_{a2}] \leq 0, \quad \forall \text{SNS}(a, i) \tag{4}$$

$$E[I_r(k)] \geq 0, \quad \forall r, \quad k \in [1, K]. \tag{5}$$

The objectives of the overall maintenance scheduling include minimizing the asset turnaround times, reducing the work-in-process inventory, and keeping a low inventory cost. These objectives are weighted together and modeled by the following objective function:

$$J = E\left[\sum_{a} \left(w_a T_a^2 + \beta_a E_a\right) + \sum_{r} \left(\sum_{k=1}^{K} \gamma_r I_r(k)\right)\right]. \tag{6}$$

In the function, w_a, β_a, and γ_r represent the nonnegative penalty weight for tardiness T_a, the nonnegative penalty weight for earliness E_a, and the inventory holding cost of rotable part type r, respectively. The scheduling problem is to select the best beginning times of disassembly/assembly and repair operations to minimize the above objective function (6) subject to intraorganization and interorganization constraints.

B. Price-Based Scheduling Method

The price-based scheduling method [1] decomposes the maintenance scheduling problem into a set of subproblems and coordinates the subproblem solutions iteratively. Since the expected resource capacity constraints (1), (2), the interorganization precedence constraints (3), (4), and the nonnegative inventory-level constraints (5) couple different assets, parts, or organizations and are difficult to deal with, the price-based scheduling method first employs a set of nonnegative Lagrange multipliers (called "shadow prices"), specifically, π (resource prices), η (precedence prices), and μ (inventory prices) to relax these hard coupling constraints into soft penalty terms such that the relaxed problem becomes unconstrained. Given the current values of π, η, and μ, the relaxed problem is to select the best beginning times of disassembly/assembly and repair operations to minimize the summation of the original objective function and the penalty terms, as the following function

shows:

$$\tilde{L} = J + \sum_{kh} \pi_{kh} \left[\sum_{aj} E(\delta_{ajkh}) - M_{kh} \right] + \sum_{kh} \pi_{kh} \left[\sum_{aij} E(\delta_{aijkh}) - M_{kh} \right] - \sum_{kr} \mu_{kr} E(I_r(k)) + \sum_{ai} \eta_{ai1} E[b_{a1} + p_{a1} + s_{a1} - b_{ai1}] + \sum_{ai} \eta_{aiJ_{ai}} E[b_{aiJ_{ai}} + p_{aiJ_{ai}} + s_{aiJ_{ai}} - b_{a2}].$$

The dual problem is to select the optimal set of prices to maximize the minimum of $\tilde{L}$. To solve the dual problem, an iterative process is used. In each iteration, the relaxed problem is solved given the current prices and the prices are then updated. In particular, since the relaxed problem $\tilde{L}$ is separable, it is classified into asset-level/part-level subproblems, which can be solved in polynomial time by using stochastic dynamic programming. The subproblem solutions are coordinated through iterative updating of the prices. As discussed in [1] and [21], because of the potentially large number of coupling constraints of different natures, the above solution methodology may not be effective to reduce the constraint violation level, and a satisfactory solution may not be obtained in a short time. To improve algorithm convergence and solution quality, an augmented surrogate subgradient solution methodology was presented in [1], which basically incorporates extra terms for penalizing violation of the coupling constraints. Since the changes to the subproblems in the augmented methodology are straightforward, they are omitted here.

IV. Price-Based Multiagent Scheduling and Coordination Framework

In the view that the original implementation is centralized and the presented surrogate solution methodology does not facilitate organization autonomy, the problem is decomposed into multiple organization-level scheduling problems (i.e., overhaul/repair/warehouse-level) first. Because of complexity, each organization-level problem is further decomposed into smaller asset-level (corresponding to asset overhaul) and part-level (corresponding to part repair) subproblems. The subproblem corresponding to asset a (7) is to select the best beginning times of disassembly and assembly given the prices to minimize $L_a(\pi, \eta, \mu)$, which measures the costs of disassembling/assembling asset a, including the earliness and tardiness penalties, the resource usage cost, and other costs due to relaxation of the corresponding precedence constraints and inventory-level constraints

$$L_a(\pi, \eta, \mu) = E\left[\beta_a E_a + \sum_{k=b_{a1}}^{c_{a1}} \pi_{kh} + \left(\sum_i \eta_{ai1} \right) b_{a1} \right] + \sum_i [\eta_{ai1} E(p_{a1} + s_{a1})] + E\left[w_a T_a^2 + \sum_{k=b_{a2}}^{c_{a2}} \pi_{kh} - \left(\sum_i \eta_{aiJ_{ai}} \right) b_{a2} + \sum_r \left[\sum_{k=1}^{K} (\mu_{kr} - \gamma_r) \sum_{\mathcal{K}=1}^{k} \alpha_{a2\mathcal{K}}^r \right] \right]. \quad (7)$$

The subproblem corresponding to part (a, i) is to select the best beginning times of the part repair operations given the prices to minimize $L_{ai}(\pi, \eta, \mu)$, including the resource usage cost of repairing the part and other costs due to relaxation of the precedence constraints (and inventory-level constraints if the part is rotable). Equations (8) and (9) are for the SNS parts and rotable parts, respectively

$$L_{ai}(\pi, \eta, \mu) = E\left[\sum_j \left(\sum_{k=b_{aij}}^{c_{aij}} \pi_{kh} \right) - \eta_{ai1} b_{ai1} + \eta_{aiJ_{ai}} b_{aiJ_{ai}} \right] + \eta_{aiJ_{ai}} E(p_{aiJ_{ai}} + s_{aiJ_{ai}}) \quad (8)$$

$$L_{ai}(\pi, \eta, \mu) = E\left[\sum_j \left(\sum_{k=b_{aij}}^{c_{aij}} \pi_{kh} \right) - \eta_{ai1} b_{ai1} + \sum_{k=1}^{K} (\gamma_r - \mu_{kr}) \sum_{\mathcal{K}=1}^{k} \beta_{aiJ_{ai}\mathcal{K}}^r \right]. \quad (9)$$

Let L_a^* and L_{ai}^* be the minimal costs of (7) and of (8) and (9). The organization-level problems (overhaul-level, repair-level, and warehouse-level) are defined as

$$\max q_o(\pi, \eta, \mu) = \max \left[\sum_a L_a^*(\pi, \eta, \mu) - \sum_{kh} \pi_{kh} M_{kh} \right] \quad (10)$$

$$\max q_r(\pi, \eta, \mu) = \max \left[\sum_{ai} L_{ai}^*(\pi, \eta, \mu) - \sum_{kh} \pi_{kh} M_{kh} \right] \quad (11)$$

$$\max q_w(\mu) = \max \left[\sum_{kr} I_{r0}(\gamma_r - \mu_{kr}) \right]. \quad (12)$$

On the basis of the above decomposition and autonomy of the organizations, each organization is associated with a coordinator agent, and each asset or part is represented by an agent. Each agent has a behavior module, which basically solves its corresponding subproblem. Specifically, the overhaul coordinator, the repair coordinator, and the warehouse coordinator solve the overhaul-level problem (10), the repair-level problem (11), and the warehouse-level problem (12), respectively, while each asset agent or part agent solves an asset-level subproblem (7) or a part-level subproblem (8), (9). These scheduling agents do not work in isolation, but need to coordinate with each other. To provide support for possible interactions among the agents, each agent contains the other three modules: a message receiving channel, a message sending channel, and an information module. The receiving channel stores the incoming messages, while the sending channel stores the outgoing messages. The information module manages the local knowledge of the agent as well as the information collected from the outside.

Before the above agents can be applied to solve the maintenance scheduling problem in a distributed environment, several coordination-related issues need to be addressed, specifically: What information is managed by each agent and transmitted between the agents? How is asynchronous coordination achieved? and What does the coordination language among the agents, consist of? The basic idea is to let the individual agents manage the related prices. For example, an overhaul (or a repair) coordinator agent is in charge of updating the resource prices based on the solutions obtained by the asset (or part) agents, while a warehouse coordinator agent manages the inventory prices. Given the new price information, each asset agent or part agent will apply stochastic dynamic programming to solve the corresponding subproblem in polynomial time and return the best beginning times to the coordinator agents. Since a coordinator agent does not need to wait for the information from all the relevant agents (based on the surrogate solution methodology), price updating as well as subproblem solving can be overlapped. This also explains why the agents can coordinate with each other asynchronously most of the time. A challenging issue is who will manage the precedence prices, which are accessed and updated in both overhaul scheduling and repair scheduling. One way is to let the overhaul coordinator and the repair coordinator manage the prices. To avoid simultaneous updating of the prices by them, a token-based heuristic can be used as in [7]. A drawback of this strategy is that the two coordinator agents may need to exchange a lot of token messages during scheduling. Another option is to let each asset agent as well as the associated part agents use the token-based heuristic to manage the corresponding precedence prices. This alternative is used in our framework since it does not have the drawback.

Our agent negotiation and communication language is defined to facilitate transmission of solution and price information as well as coordination among the agents. It currently consists of a set of message types, including resource multiplier message, inventory multiplier message, precedence multiplier message, subproblem solution message, and agent registration/deregistration message. The resource multiplier message, containing the latest resource price information, is sent from the overhaul/repair coordinator to an asset/part agent while the inventory multiplier message, containing the latest rotable part price information, is sent from a warehouse coordinator to an asset/part agent. The precedence multiplier message is exchanged between an asset agent and a part agent. The agent registration message is sent by an asset/part agent to register itself with a coordinator agent, while the deregistration message is sent to a coordinator agent when an asset/part agent notices that the asset/part has been maintained.

Overall, our agent scheduling framework is distributed by decomposing the original problem into organization-level scheduling problems, which are further decomposed into smaller asset-level and part-level subproblems, and implementing distributed and asynchronous scheduling and coordination as described above (and illustrated in Fig. 2). Each coordinator agent will interact with the relevant asset and part agents to perform intraorganization scheduling. The asset agents also interact with the part agents for reducing the violation of precedence constraints

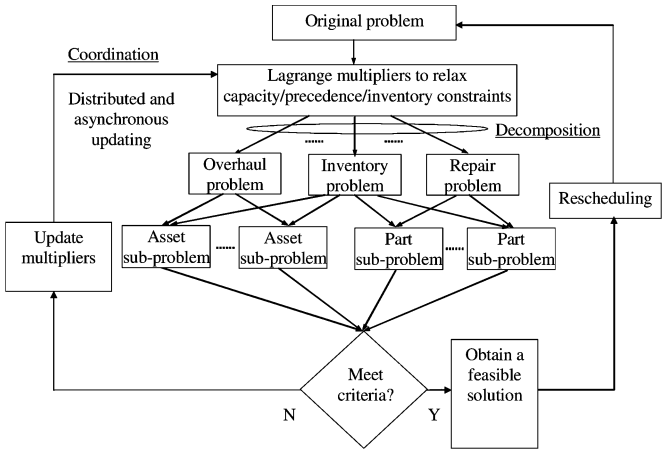

Fig. 2. Schematic of the price-based method. The role of the overhaul/repair/warehouse coordinator is to solve the overhaul-level/repair-level/warehouse-level scheduling problem, while the role of each asset/part agent is to solve the corresponding asset-level/part-level problem.

gradually. All these agents together perform the optimization, and they can be run on different computing resources. In the experimental study, the timeliness and scalability of the agent scheduling framework will be demonstrated by varying heterogeneity of the distributed environment and number of worn-out assets processed over time.

In addition to the above-mentioned scheduling agents, other types of agents can be important from a futuristic point of view. In the future, assets and parts will have computing, communication, and sensing capabilities. It is envisioned that the associated agents can directly run in the onboard processing units of assets and parts to collect asset and part condition information in real time, help improve scheduling, and monitor the execution of maintenance operations. In particular, the onboard agents will enable assets and parts to actively participate in the scheduling process by suggesting what maintenance operations are needed and exploring where and when to perform the operations. All these agents can be in different status as time passes by. The condition-information-collecting agents will be generally active if possible. The maintenance monitoring agents may be inactive before assets and parts are sent for maintenance, while the scheduling agents may be inactive if no potential problem is identified and become active otherwise. Moreover, the collecting and monitoring agents generally stay in the onboard processing units of assets and parts, while the scheduling agents may migrate from the assets and parts to the overhaul centers and repair shops to explore where and when to perform the operations. After scheduling is done, they then migrate back to their corresponding assets and parts. The agents with the ability of migrating from one computing resource (e.g., the onboard processing unit of an asset) to another resource (e.g., a computer of an overhaul center) are called mobile agents. These functionalities of agents are expected to be a key characteristic of next-generation e-scheduling. To support such functionalities, other message types such as inquiry need to be defined to extend our agent communication language.

In the view that a large number of agents in a maintenance network interact with each other for scheduling overhaul and repair operations, the multiagent architecture should be robust and scalable. Carolina [22], a Java-based distributed agent execution environment, was chosen among several because of two reasons. First, Carolina was developed with the goal of robustness and scalability, and its performance has been systematically studied. In particular, a distributed directory service was employed to avoid the risk of single-point failure and deal with a large number of agents. In addition, an ANTS-style approach for message forwarding and routing optimization was used to ensure successful communications among mobile agents. Second, we have been using Carolina for some time and feel comfortable to implement the agent-scheduling framework in Carolina. Like Jade (http://jade.tilab.com) as well as other Java-based agent systems, Carolina employs the threading support of Java to achieve the autonomy of agents, provides a set of services to be used by agents for accessing distributed resources and interacting with other agents, and relies on Java's serialization/deserialization support for transmitting messages and migrating objects (agents) through the network. A difference is that the message format in Jade conforms to the agent communication language (ACL) of FIPA, while the format in Carolina does not. To deploy our scheduling framework in a distributed environment with multiple agent infrastructures, it is necessary to implement the framework in an infrastructure such as Jade or extend Carolina with some interoperable protocols (e.g., ACL of FIPA). However, this issue is not addressed here.

V. Experimental Design

The goal of the experimental study is to answer the questions posed earlier regarding the performance of the agent scheduling framework in a distributed environment. Two hypotheses are examined. Our first hypothesis is that the framework, built on the price-based method, can obtain a high-quality solution in a timely manner. The second hypothesis is that the framework would be able to solve large-scale problems by scheduling in a distributed and asynchronous fashion. In this section, the experimental design is described in detail with the focus on selecting factors that are key to answering our questions and to validating the hypotheses. Other related issues are also addressed, including performance measures, specification of experiments, and testbed setup.

A. Performance Measures

For the ease of validating the hypotheses, the following performance measures are considered. To reflect the different requirements of high-priority assets and those of low-priority assets (as described in Section V-B), the quality of a solution is defined by three measures: mean turnaround time of high-priority assets (HTAT), mean turnaround time of low-priority assets (LTAT), and mean INVC. In addition to these three measures, another two measures of interest are the total scheduling time (SCHT) for a given test case and the total number of precedence messages that are transmitted between the asset agents and the part agents in the case of using our agent scheduling framework. The number of precedence messages reflects the amount of interorganization coordination traffic between overhaul scheduling and repair scheduling. Another reason for counting the number of precedence messages only is that the number of other types of messages is a constant, given the same iterations (as the stopping criterion of scheduling) and the same-scale test cases.

B. Factor Selection

To help answer the questions and validate the hypotheses, a total of eight factors were selected.

1) *Solution framework type (SolFrm): SolFrm* is defined for the purpose of checking whether the performance of the agent scheduling framework can approach that of a centralized implementation of the price-based scheduling method.
2) *Communication delay (CommuDelay) and Processor Speed (ProcSpd):* These two factors reflect heterogeneity in a distributed environment and may prevent the agent scheduling framework from finding a high-quality solution in a short time interval (e.g., 5 min). To see how much the framework is influenced under different communication delay and processor speed patterns, three levels are defined for each factor to reflect three cases, specifically, a homogeneous case, a less heterogeneous case, and a very heterogeneous case.
    a) *Levels for communication delay:* Since our experiments are carried out in a local area network, which generally has a small communication delay due to small latency and large bandwidth, artificial delay functions are applied to simulate the three cases of communication delays. One is a small uniform delay pattern (for homogeneous communication delay), under which all the messages are delayed between 0 and 30 ms before being sent. The reason of using a small uniform delay pattern is that, even in the homogeneous case, the actual delay for a message transmission varies in a small range. The second is a large uniform delay pattern from 0.5 to 3 s (for very heterogeneous communication delay). The last one is a combination of the previous two, with 90% of the messages being delayed slightly and 10% delayed significantly.
    b) *Levels for processor speed:* Two heterogeneous systems are considered. In one very heterogeneous system, all the repair shops use different types of workstations from those for the overhaul centers. In the less-heterogeneous system, half of the repair shops use different types of workstations while the other half uses the same types of workstations as the overhaul centers. As a comparison, a homogeneous system is also considered. In the study, three types of workstations are used. Pentium IV 2-GHz 1-GRAM workstations form the homogeneous system, and they are also utilized to schedule overhaul

operations in the heterogeneous systems (and one of them is used to test centralized scheduling). Pentium IV 2.6-GHz 512-MRAM and Pentium III 1-GHz 128-MRAM workstations are used together to schedule repair operations in the heterogeneous systems.

3) *Number of assets:* This factor reflects the problem scale, and the agent scheduling framework scalability can be shown by examining the results of scheduling different-scale problems. This factor specifies the total number of worn-out assets considered in a test case. While the exact number of assets arriving over time may not be known beforehand in practice, it can often be estimated. So, in the experiments, it is assumed to be known. In addition, in practice, some worn-out assets are considered to be more important than others. As such, in the experiments, the assets are classified into two categories, specifically, 30% high-priority assets and 70% low-priority assets. Those high-priority assets should be maintained as soon as possible without being delayed by maintenance of the low-priority ones.
4) *Initial inventory relative level (InitInv), holding cost (γ_r), targeted resource utilization (ResUtil), and uncertainty level (UncertL):* These factors reflect the different conditions arising in maintenance scheduling.
   a) *InitInv*: This is defined to be a ratio of the initial inventory level to the number of assets. The initial inventory level I_{r0} is proportional to the number of assets, being equal to *InitInv*$*$*#_of_assets*. A high initial inventory relative level may improve mean HTAT and LTAT as well as lead to a high inventory cost. So it needs to be chosen suitably. For simplicity, only one rotable part type is assumed here (note that, in the case of multiple rotable part types, a different initial inventory relative level may be specified for each type).
   b) γ_r: At each time slot t, the current inventory level multiplied by γ_r is the inventory cost at t. The total inventory cost is the summation of the inventory cost at each time slot, and mean INVC is the total cost divided by the total time. While a high initial inventory relative level may shorten mean HTAT and LTAT, it may increase mean INVC because the current inventory level at any time slot t is more likely to be high.
   c) *ResUtil*: This factor is defined to be a ratio of the total busy time to the total available time of the resources of a maintenance resource type. A resource is busy at time slot t if it is conducting a maintenance operation at that time. This factor is used to simulate the cases with sufficient or insufficient maintenance resources. Specifically, given a targeted level of resource utilization, the estimated processing requirements of maintenance operations, and the available time per resource, the numbers of available resources in the overhaul centers and repair shops can be decided.

TABLE I
FACTORS AND THEIR LEVELS

Factor	Level 1	Level 2	Level 3
SolFrm	Centralized	Agent-based	N/A
CommuDelay	0∼0.03s	0∼0.03/0.5∼3s	0.5∼3s
ProcSpd	Homo.	Hete. 1	Hete. 2
#_of_assets	100	200	400
InitInv	0	5%	10%
γ_r (Holding cost)	0	0.5	1
ResUtil.	60%	70%	80%
UncertL	0	1	2

TABLE II
ORTHOGONAL ARRAY L_9, CONSISTING OF FOUR COLUMNS AND NINE ROWS

Expt-No	Col. 1	Col. 2	Col. 3	Col. 4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

   d) *UncertL*: This factor captures the uncertainties involved in scheduling, including uncertain asset arrivals and operation processing times. Here, it denotes the standard deviation of each uncertain variable. Each variable is assumed to take three values with equal probability if *UncertL* is nonzero.

For each of the factors, since the framework may behave differently under its different levels, its impact is studied. Furthermore, these factors together may have different effects on system performance from the impact of each individual one. In this study, the interactions between *CommuDelay* and *ProcSpd* are examined to determine the timeliness of agent-based scheduling. In addition, the interactions between *InitInv* and *ResUtil* are studied because both factors are important for reducing mean asset turnaround time.

Except for the solution framework type, which has only two levels, every factor has three levels because any interesting trends can be seen only by using more than two levels (as shown in Table I).

C. Specification of Experiments and Testbed Setup

Even with the above eight factors selected, it is still too costly to measure the impact of all level combinations of these factors (2×3^7 combinations). So, a subset of level combinations has to be considered, which can be decided by employing orthogonal arrays [23]. An orthogonal array is a two-dimensional matrix with orthogonality satisfied. The rows are called cells, representing experiments to be performed, while the columns correspond to the different factors whose effects will be studied. An orthogonal array of strength $g (g \geq 2)$ means that, for any g columns, all level combinations of factors represented by the g columns occur an equal number of times. This balancing property is called orthogonality. As an example, the standard orthogonal array L_9 [23, p. 287] has nine cells and four columns

(representing four three-level factors at most), as shown in Table II.

To measure the main effect of each of the eight factors on system performance, the standard orthogonal array L_{18} [23, p. 291] is applied. Orthogonal array L_{18} can handle seven three-level factors and one two-level factor. Furthermore, to identify the possible interactions between *CommuDelay* and *ProcSpd* as well as between *InitInv* and *ResUtil* in using the agent scheduling framework, another orthogonal array L_9 is applied. In this case, the middle levels of the rest five factors are always used. Overall, one set of test cases is generated based on L_{18} and two other sets of test cases based on L_9.

In addition to the above eight factors, other factors, such as the number of operations per part repair and the operation processing times, need to be specified as well. Our rule for setting these factors is based on simplicity and consistency so that their effects can be minimized. Specifically, the high-priority assets are assigned a tardiness weight of 10.0, and the low-priority ones are assigned a weight of 1.0. Each asset contains one SNS part and one rotable part. Asset overhaul has one disassembly operation and one assembly operation, and part repair consists of two operations. Each type of the operations (e.g., disassembly, first operation of rotable repair, etc.) is conducted by a different maintenance resource type. The expected processing time is set to 4 for rotable part repair operations and 3 for all the other operations. Using these settings, it is clear that the maintenance of an asset (i.e., disassembly plus assembly plus two repair operations on the SNS part) can be done in 12 time slots on average if there are sufficient maintenance resources and rotable parts. However, it is often the case that insufficient maintenance resources are available. So the due date should not be set too tight. Here, it is set to the expected asset arrival time plus 17 (22) for a high-priority (low-priority) asset. The maintenance network consists of one overhaul center and two repair shops, one in charge of the SNS parts and the other responsible for the rotable parts. All these values are kept the same in the test cases to minimize the effects of the associated factors.

To conduct the experiment, a simulation control module is implemented at the top of the agent scheduling framework. It loads the test cases into the system one at a time and calls agent-based scheduling or centralized scheduling on the scenario captured by each test case. Before agent-based scheduling is called, all the needed scheduling agents are created first on the agent servers running in the homogeneous or heterogeneous system, as described in Section V-B. Since two single-processor computers are used to perform scheduling of the overhaul center as well as each repair shop, the corresponding coordinator agent runs on one computer, while the related asset agents or part agents are evenly dispatched to the two assigned computers. To handle dynamic asset arrivals, the control module calls rescheduling at a specified interval. Specifically, the simulation horizon is set long enough for maintaining all the assets, while for (re)scheduling, a shorter horizon is used to restrict the attention to the assets that will arrive in the near future as well as those that have arrived but not finished maintenance. Upon the completion of (re)scheduling, the scheduling policy obtained is recorded and used to guide the arrangement of maintenance operations. To make measures meaningful, the behavior of the system in the initial warm-up phase and final phase is intentionally ignored. In particular, only the assets completed during a chosen interval are counted, and the interval starts right after the first 20% of assets finish maintenance and terminates when 70% of assets are maintained.

To facilitate the study, all the test cases are generated offline, each of which contains complete information of what maintenance resources are there in each organization, when the worn-out assets arrive, what operations need to be carried out, and so on. Such complete information, however, is not used all at once during (re)scheduling. As mentioned above, only the information that is considered currently available is used in each scheduling period. Hence, our scheduling framework is dynamic.

TABLE III
RESULTING p-VALUES OF ONE-WAY ANOVA ANALYSIS

Factor	HTAT	LTAT	INVC	MSGN	SCHT
InitInv	0.14	0.006	0.034	>0.5	>0.8
γ_r	>0.7	>0.9	0.11	>0.9	>0.9
ResUtil	0.11	0.036	>0.4	>0.9	>0.9
UncerLvl	0.061	>0.6	>0.4	>0.6	>0.9
SolFrm	>0.9	>0.5	>0.9	<0.001	<0.001
CommuDelay	>0.9	>0.9	>0.7	>0.9	>0.1
ProcSpd	>0.3	>0.5	>0.7	>0.3	>0.4
#_of_assets	>0.3	>0.8	>0.9	>0.1	>0.4

If the p-value of a factor on a metric is less than or equal to 0.05, the factor has a significant impect on the metric. If the p-value is greater than 0.05 but less than or equal to 0.10, the factor has a marginal impect on the metric. Otherwise the factor has insignificant impact.

TABLE IV
RESULTING p-VALUES OF TWO-WAY ANOVA ANALYSIS

Interactions	HTAT	LTAT	INVC	MSGN	SCHT
CommuDelay & ProcSpd	>0.5	>0.3	>0.3	<0.001	0.12
InitInv & ResUtil	>0.6	<0.001	<0.001	0.004	0.10

VI. EXPERIMENTAL RESULTS

Analysis of variance (ANOVA) [24] is a statistical technique that measures the relative effects of the different factors on system performance. In this study, one-way and two-way ANOVAs are applied to analyze the main effects of the individual factors (see Table III) and the possible interactions between communication delays and processor speeds as well as between inventory levels and resource utilizations (see Table IV), respectively. From the results and analyses, it is concluded that the agent scheduling framework can approach similar performance as a centralized implementation of the price-based scheduling method in terms of the solution-quality-related measures. Furthermore, the framework can schedule in a timely manner in that its SCHT is within 5 min (from about 60 to 270 s) for most test cases. Only for the 400-asset test cases is the time over 5 min, about 330 s, under large uniform communication delays. As a comparison, the SCHT of the centralized scheduling is less than 1 min. This is because Java is slow in message transmission. When new techniques become available, message transmission time may be reduced greatly. Overall, the agent

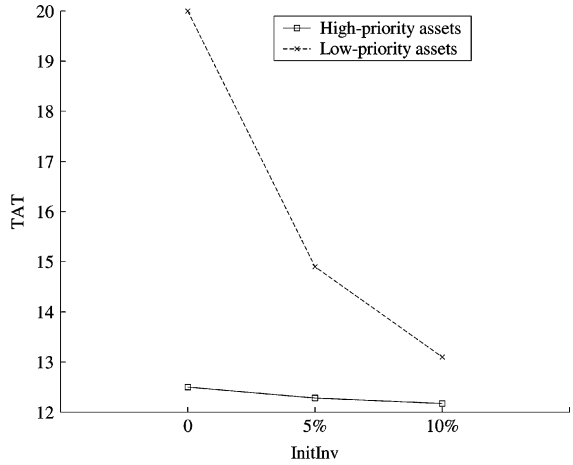

Fig. 3. Relationship between initial inventory relative level *(InitInv)* and mean asset turnaround time.

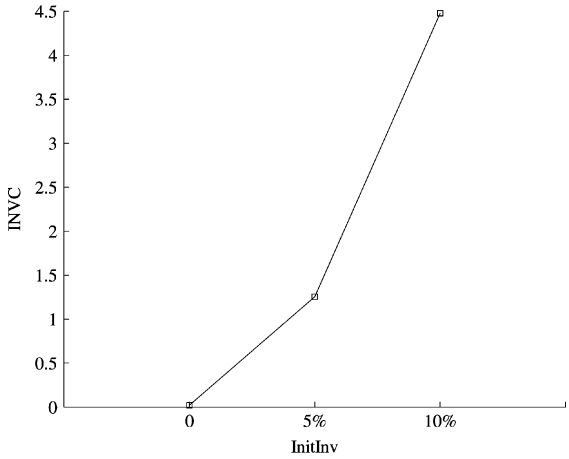

Fig. 4 Nonlinear relationship between initial inventory relative level *(InitInv)* and mean INVC.

scheduling framework is able to solve the large-scale test cases. These results basically validate our hypotheses and answer the questions posed earlier.

In the following, the main effects of the individual factors are presented first. The interaction analysis follows. In all the analyses, the significance level is set to 0.05, and the marginal significance level is set to 0.10.

A. Main Effects of the Individual Factors

The results from the test cases and the following analysis show that the high-priority assets generally have short turnaround times with small variances, while the low-priority assets tend to be delayed and have long turnaround times under insufficient maintenance resources. Furthermore, having a nonempty initial inventory can shorten low-priority asset turnaround times as well as reduce the effect of uncertainties. However, having a high inventory may not be necessary.

1) Initial Inventory Relative Level: As shown in Table III, the initial inventory relative level (*InitInv*) significantly affects mean LTAT and mean INVC, but its impact on mean HTAT is not significant. The reason is that parts, especially rotable parts, cannot be repaired in a timely fashion such that assembly of low-priority assets is delayed and a nonempty rotable inventory will be helpful. In particular, as shown in Fig. 3, by using 5% *InitInv*, LTAT becomes 14.9, which is about 25% less than the value (i.e., 20.0) with empty initial inventory. It decreases to 13.1 if *InitInv* is set to 10%. Since the benefit of using higher *InitInv* like 10% is diminishing and LTAT being 14.9 is already much better than the desired LTAT, it seems that 5% *InitInv* is a good choice. Its insignificant impact on HTAT is mainly because HTAT is already low, close to the minimal value of finishing the whole maintenance process of a worn-out asset. Our explanation of its significant impact on INVC is that, under a high-inventory relative level, it is more likely that some rotable parts are always available in the inventory, as reflected in Fig. 4. From this point of view, it is not desirable to use a high *InitInv* relative level.

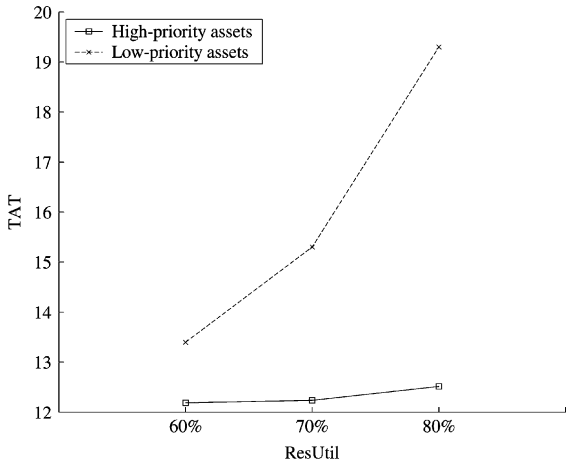

Fig. 5. Relationship between resource utilization *(ResUtil)* and mean asset turnaround time.

2) Targeted Resource Utilization: As shown in Table III, the impact of targeted resource utilization *(ResUtil)* on mean LTAT is significant, while its impact on mean HTAT is not. On the one hand, maintenance of low-priority assets may be delayed under high resource utilization because the needed parts cannot be repaired in time, as shown in Fig. 5. One interesting trend is that LTAT slightly increases from 13.4 to 15.3 when the ratio changes from 60% to 70%. But, it becomes 19.3 with an 80% ratio. With such a nonlinear relationship, it is expected that a ratio higher than 80% will lead to a much higher LTAT. On the other hand, high-priority assets are maintained in a timely fashion even under high resource utilization. As a result, the impact of this factor on HTAT is insignificant.

3) Uncertainty Level: The uncertainty level marginally affects mean HTAT, as shown in Table III. From Table V, it is clear that a high uncertainty level does lead to the increase of both HTAT and LTAT. The variance of HTAT is small due to the timely maintenance of high-priority assets such that the increasing trend of HTAT becomes marginally significant. On the

TABLE V
MEAN HTAT AND LTAT AND THEIR STANDARD DEVIATION VALUES UNDER DIFFERENT UNCERTAINTY LEVELS

UncertL	HTAT		LTAT	
	Mean	Std. Dev.	Mean	Std. Dev.
0	12.15	0.13	14.65	2.34
1	12.27	0.25	16.47	5.24
2	12.53	0.35	16.88	5.06

contrary, the variance of LTAT is much larger. Our explanation is that under high resource utilization, LTAT tends to increase quickly with the increase of uncertainty level if *InitInv* is zero, but a nonzero *InitInv* may reduce the effect of the uncertainty level. When these different scenarios are grouped together, the variance of LTAT is big, which makes the increasing trend insignificant.

The impact of solution framework type, problem scale, processor speed, and communication delay on mean HTAT and of LTAT and mean INVC is insignificant. These results show that the agent scheduling framework as well as the price-based scheduling method performs efficiently under different conditions in the homogeneous as well as heterogeneous environments. In addition, by scheduling in a distributed and asynchronous fashion, the framework is scalable. What we did not expect is that the holding cost did not have significant impact. This is possibly due to the quadratic tardiness penalty term in the objective function (6) which dominates the linear inventory cost term.

B. Interactions of Pairs of Factors

By using two-way ANOVA, it is shown that the inventory relative levels and resource utilizations have significant interaction effects on mean LTAT and mean INVC, while the interaction effects between communication delays and processor speeds on these solution-quality-related measures are insignificant.

1) Communication Delays and Processor Speeds: As shown in Table IV, there is a significant interaction effect between communication delays and processor speeds on the number of transmitted precedence messages (MSGNs). The reason is that under small communication delays, the asset agents and the part agents in the homogeneous system can progress approximately at the same pace in solving their subproblems, while in the two heterogeneous systems, the part agents slightly lag behind the asset agents. As a result, the asset agents and the part agents in the homogeneous system can transmit more precedence messages on average, as shown in Fig. 6. However, under large communication delays, the time for solving a subproblem in one iteration becomes negligible compared to the time waiting for messages between two iterations. So the systems make no impact since they all transmit similar amounts of messages. In the view that the MSGN somehow reflects the amount of interorganization coordination traffic between overhaul scheduling and repair scheduling, given a significant impact of *CommuDelay* and *ProcSpd* on MSGN, it seems that at least one of the three solution-quality-related measures should be influenced. The analysis, however, shows that the impact is insignificant.

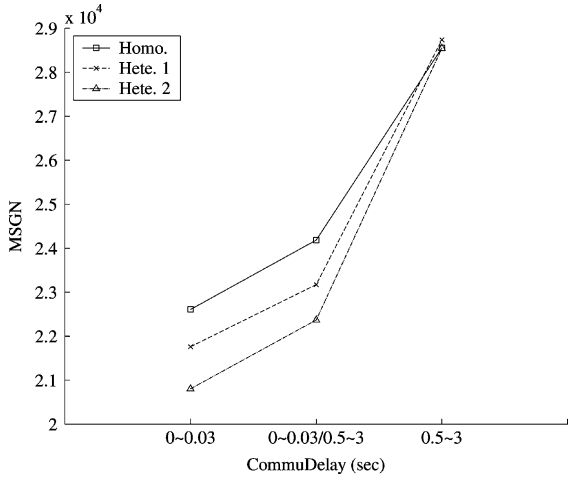

Fig. 6. MSGN under different processor speeds *(ProcSpd)* and communication delays *(CommuDelay)*.

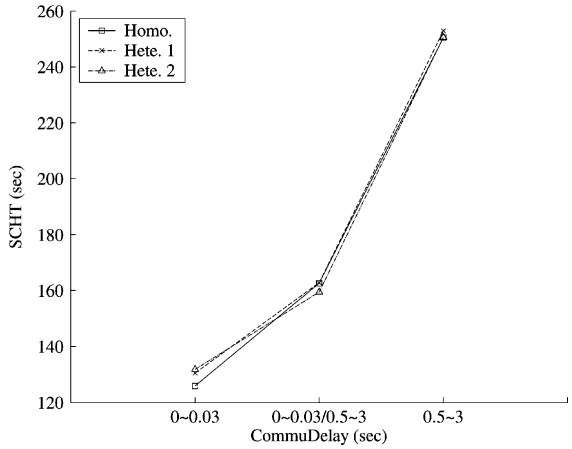

Fig. 7. SCHT under different processor speeds *(ProcSpd)* and communication delays *(CommuDelay)*.

Our explanation is that the iterative updating and exchanging of resource prices and inventory prices also indirectly achieve interorganization coordination.

Table IV shows that the interaction effect on the SCHT is insignificant. This can be explained by observing the SCHT difference between the centralized method and our agent scheduling framework. The centralized method can finish in about 17 s, while the agent scheduling framework has to spend about 130 s even under low communication delays. As mentioned earlier, this is because Java is slow in message transmission compared to other languages like C. Hence, our agent scheduling framework spends more time on communication than computation, and different processor speeds will not lead to large differences on SCHT, as shown in Fig. 7. Another reason is that our agent scheduling framework schedules and coordinates in an asynchronous fashion. Although the individual part agents in the heterogeneous systems may take more time in solving their problems, the overall SCHT does not change much.

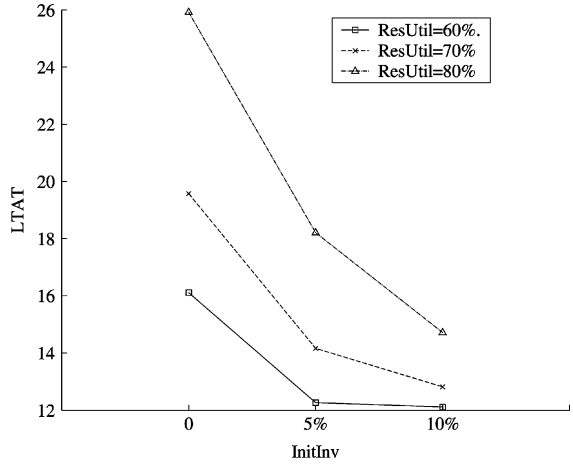

Fig. 8. Mean LTAT under different resource utilizations *(ResUtil)* and initial inventory relative levels *(InitInv)*.

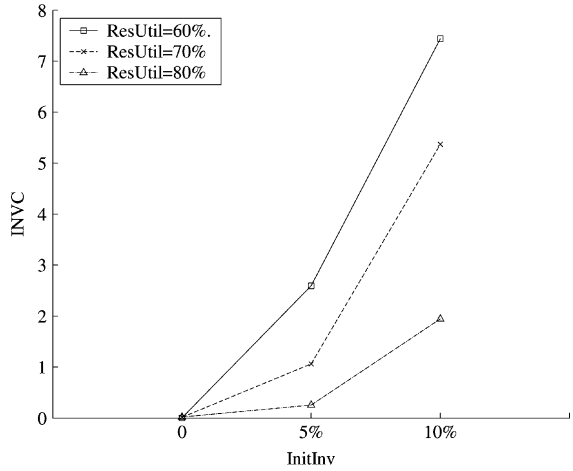

Fig. 9. Mean INVC under different resource utilizations *(ResUtil)* and initial inventory relative levels *(InitInv)*.

2) Initial Inventory Relative Levels and Targeted Resource Utilizations: Their interaction effects on mean LTAT and mean INVC are significant. This is because LTAT increases quickly with the increase of resource utilization if the initial inventory relative level is zero and the degree of increase is reduced under nonzero inventory relative level, as shown in Fig. 8. Furthermore, under low resource utilization, a 5% inventory relative level may be sufficient for achieving desired turnaround time for most low-priority assets, while under high resource utilization, such a level may not be enough. Similarly, Fig. 9 shows that, under low resource utilization, mean INVC becomes high with the increase of inventory relative level. Our explanation is that rotable inventory always has extra rotable parts over the time. While under high resource utilization, the rotable parts are consumed quickly even with a high inventory relative level.

As explained above, mean HTAT is already low. From Fig. 10, it can be seen that the HTAT increases (or decreases) slightly with the increase of resource utilization (or the increase of inventory level). As a result, the trend by varying initial inventory relative levels or resource utilizations is similar for different resource utilizations or different initial inventory relative levels. The reason that HTAT does not decrease when the initial inventory relative level increases from 5% to 10% in the case of *ResUtil* being 60% is that it has already reached the minimal expected value. Its increase from 12.00 to 12.02 is insignificant and can be contributed to sampling error in generating test cases.

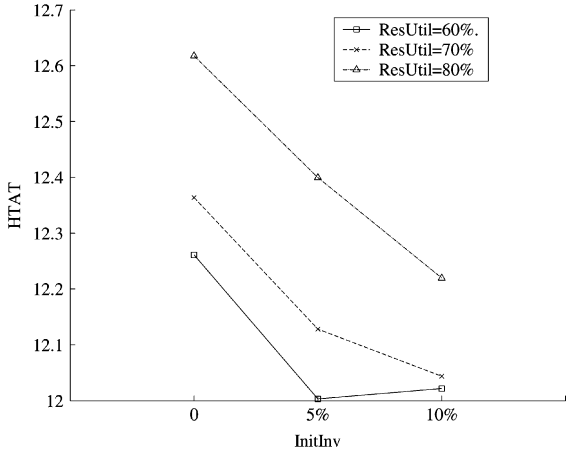

Fig. 10. Mean HTAT under different resource utilizations (*ResUtil*) and initial inventory relative levels (*InitInv*).

TABLE VI
COMPARISON BETWEEN THE AGENT-BASED AND THE CENTRALIZED MODELS

Item	Agent-based	Centralized
Requiring global info.	No	Yes
Organization autonomy	Supported	Not-supported
Scheduling time	Long	Short
# of Commu. Messages	Prop. to # of iter.	Constant
Sol. qual. under ProcSpd	Not-affected	N/A
Sol. qual. under CommuDelay	Not-affected	N/A
Sol. qual. under ResUtil	Comparable	Comparable
Sol. qual. under InitInv	Comparable	Comparable
Sol. qual. under UncertL	Comparable	Comparable
Sol. qual. under holding cost	Comparable	Comparable
Perf. on large-scale probl.	Scalable	Scalable

Table IV also shows that these two factors have a significant interaction impact on the MSGN and a marginally significant impact on SCHT. The reason is that MSGN and SCHT are approximately proportional to the total number of rescheduling times for each test case and the number of rescheduling may be less for a case with nonzero inventory relative level and/or low resource utilization than that of another case with zero inventory relative level and/or high resource utilization.

C. Summary of Results

In this section, the main effects of the individual factors as well as the interactions of pairs of factors are analyzed, as summarized in Table VI. From the results and analyses, our hypotheses as stated in the beginning of Section V are supported, and the questions are positively answered. Specifically,

the price-based agent scheduling and coordination framework can obtain a high-quality solution under different conditions in a timely manner. On the one hand, the framework achieves the same level of performance as a centralized implementation of the price-based scheduling method in terms of the solution-quality-related measures. On the other hand, heterogeneity in a distributed environment does not prevent the framework from performing effectively. In addition, the results and analyses reflect that high-priority assets are maintained in time while many low-priority assets are delayed under high resource utilization because of the delay of repairing parts, especially rotable parts. That is why a nonzero initial inventory relative level may lead to shorter turnaround times for low-priority assets. These relations can provide organizations with guidance on setting up a suitable amount of resources and initial inventory relative level. Since these factors, such as resource utilization and heterogeneity, are not restricted to the maintenance scheduling problem only, the results may provide insights for other multiorganization scheduling problems.

VII. Conclusion

In this paper, a price-based multiagent scheduling and coordination framework for maintenance networks is explored, and several key factors that may affect the performance of the framework are identified and their effects experimentally studied. Two orthogonal arrays are applied to systematically conduct the experiments and reduce the total experimental effort. The results validate our hypotheses and demonstrate that our framework overcomes the major difficulties and could be a first step toward the next generation of e-scheduling for maintenance networks.

The current model incorporates a number of simplifying assumptions. To deploy it, the activities of other service providers, such as ordering and production planning of part distributors and spare part manufacturers, need to be modeled. In addition, rather than giving the information to the agents, it is better for the agents to retrieve the information automatically and update their knowledge over time. To obtain faster convergence, intraorganization scheduling as well as interorganization coordination needs to be further improved. With the advancement of computing, communication, and sensing capabilities of assets and parts, it is important to explore how to monitor asset/part conditions in real time and improve scheduling by using mobile software agents. Finally, the issues related with security and fault tolerance should be addressed.

Acknowledgment

The authors would like to thank D. Yu for her helpful discussions and the anonymous reviewers who helped to significantly improve this paper. Earlier preliminary work was reported in [25] and [26].

References

[1] P. B. Luh, D. Yu, S. Soorapanth, A. I. Khibnik, and R. Rajamani, "A Lagrangian relaxation based approach to schedule asset overhaul and repair services," *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 2, pp. 145–157, Apr. 2005.

[2] V. Guide, Jr., M. Kraus, and R. Srivastava, "Scheduling policies for remanufacturing," *Int. J. Prod. Econ.*, vol. 48, pp. 187–204, 1997.

[3] V. Guide, Jr., V. Jayaraman, and R. Srivastava, "Production planning and control for remanufacturing: A state-of-the-art survey," *Robot. Comput. Integr. Manuf.*, vol. 15, pp. 221–230, 1999.

[4] V. Guide, Jr. and R. Srivastava, "Inventory buffers in recoverable manufacturing," *J. Oper. Manage.*, vol. 16, pp. 551–568, 1998.

[5] R. Pool, "If it ain't broke, fix it," *Technol. Rev.*, vol. 104, no. 7, p. 64, Sep. 2001.

[6] M. Koc and J. Lee, "System framework for next-generation e-maintenance systems," *Trans. Chin. Mech. Eng.*, vol. 12, no. 5, 2001.

[7] P. B. Luh, M. Ni, H. Chen, and L. S. Thakur, "Price-based approach for activity coordination in a supply network," *IEEE Trans. Robot. Autom.*, vol. 19, no. 2, pp. 335–346, Apr. 2003.

[8] N. M. Sadeh, D. W. Hildum, D. Kjenstad, and A. Tseng, "MASCOT: An agent-based architecture for dynamic supply chain creation and coordination in the internet economy," *Prod. Planning Control*, vol. 12, no. 3, 2001.

[9] N. Zarour, M. Boufaida, L. Seinturier, and P. Estraillier, "Supporting virtual enterprise systems using agent coordination," *Knowl. Inf. Syst.*, vol. 8, no. 3, pp. 330–349, Sep. 2005.

[10] H. Pham, "Software agents for internet-based systems and their design," in *Intelligent Agents and Their Applications*, L. Jain, Z. Chen, and N. Ichalkaranje, Eds. Heidelberg, Germany: Physica-Verlag, 2002, pp. 109–154.

[11] E. Santos, Jr., F. Zhang, and P. B. Luh, "Intra-organizational logistics management through multi-agent systems," *Electron. Commerce Res.*, vol. 3, no. 3–4, pp. 337–364, 2003.

[12] R. Lee and J. Liu, "iJADE WeatherMAN: A weather forecasting system using intelligent multiagent-based fuzzy neuro network," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 34, no. 3, pp. 369–377, May 2004.

[13] I. Satoh, "Building reusable mobile agents for network management," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 33, no. 3, pp. 350–357, May 2003.

[14] E. Oliveira, K. Fischer, and O. Stepankova, "Multi-agent systems: Which research for which applications," *Robot. Autonom. Syst.*, vol. 27, pp. 91–106, 1999.

[15] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. 29, no. 12, pp. 1104–1113, Dec. 1980.

[16] N. M. Sadeh, D. W. Hildum, and D. Kjenstad, "Agent-based e-supply chain decision support," *J. Org. Comput. Electron. Commerce*, vol. 13, no. 3, pp. 225–241, 2003.

[17] K. Decker and V. Lesser, "Designing a family of coordination algorithms," in *Proc. 1st Int. Conf. Multi-Agent Systems (ICMAS'95)*, San Francisco, CA, 1995, pp. 73–80.

[18] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artif. Intell.*, vol. 110, no. 2, pp. 241–273, 1999.

[19] O. Shehory, G. Sukthankar, and K. Sycara, "Agent aided aircraft maintenance," presented at the Conf. Autonomous Agents, Seattle, WA, 1999.

[20] X. Xu and M. Kezunovic, "Mobile agent software applied in maintenance scheduling," presented at the 2001 North American Power Symp., College Station, TX, 2001.

[21] D. Yu, P. B. Luh, and S. Soorapanth, "A new Lagrangian relaxation based method to improve schedule quality," presented at the 2003 IEEE/RSJ Int. Conf. Intell. Robots Syst., Las Vegas, NV, 2003.

[22] M. G. Saba and E. Santos, Jr., "The multi-agent distributed goal satisfaction system," in *Proc. Int. ICSC Symp. Multi-Agents and Mobile Agents in Virtual Org. and E-Commerce*, 2000, pp. 389–394.

[23] M. S. Phadke, *Quality Engineering Using Robust Design*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[24] D. G. Kleinbaum, L. L. Kupper, K. E. Muller, and A. Nizam, *Applied Regression Analysis and Other Multivariable Methods*. Duxbury, 1997.

[25] F. Zhang, E. Santos, Jr., and P. B. Luh, "Mobile multi-agent-based scheduling and coordination of maintenance networks," in *Proc. Int. Conf. Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, 2003, pp. 279–285.

[26] F. Zhang, P. B. Luh, and E. Santos, Jr., "Performance study of multi-agent scheduling and coordination framework for maintenance networks," in *Proc. 2004 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2004)*, Sendai, Japan, 2004, pp. 2390–2395.

Feng Zhang received the B.S. degree in computer science and technology and the M.S. degree in computer applications from Tsinghua University, Beijing, China, in 1993 and 1995, respectively. He is currently pursuing the Ph.D. degree at the University of Connecticut, Storrs.

From 1995 to 1999, he was a Software Engineer with the Tsinghua Unispendour Group, Beijing, working on research and development of computer-aided-design software. His research interests include multiagent scheduling and coordination, load balancing, cooperative algorithm portfolios, and modeling of restricted processor sharing.

Eugene Santos, Jr. (M'93–SM'04) received the B.S. degree in mathematics and computer science and the M.S. degree in mathematics (specializing in numerical analysis) from Youngstown State University, Youngstown, OH, in 1985 and 1986, respectively, and the Sc.M. and Ph.D. degrees in computer science from Brown University, Providence, RI, in 1988 and 1992, respectively.

He is currently a Professor of engineering at the Thayer School of Engineering, Dartmouth College, Hanover, NH, and Director of the Distributed Information and Intelligence Analysis Group (DI^2AG). Previously, he was a faculty member at the Air Force Institute of Technology, Wright-Patterson AFB, and the University of Connecticut, Storrs. He has over 130 refereed technical publications and specializes in modern statistical and probabilistic methods with applications to intelligent systems, uncertain reasoning, and decision science. Most recently, he has pioneered new research on user and adversarial behavioral modeling. He is an Associate Editor for the *International Journal of Image and Graphics*.

Dr. Santos is an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. He has chaired or served on numerous major IEEE conferences and professional society meetings.

Peter B. Luh (S'76–M'80–SM'91–F'95) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1973, the M.S. degree in aeronautics and astronautics engineering from the Massachusetts Institute of Technology, Cambridge, in 1977, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, in 1980.

Since 1980, he has been with the University of Connecticut, Storrs, and currently is the SNET Professor of communications and information technologies in the Department of Electrical and Computer Engineering. He is also a Visiting Professor at the Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing, China. He is interested in planning, scheduling, and coordination of design, manufacturing, supply chain, and other activities; configuration and operation of building elevator and HVAC systems for normal and emergency conditions; schedule, auction, portfolio optimization, and load/price forecasting for power systems; and decision making under uncertain, fuzzy, or distributed environments. He is an Associate Editor of the *IIE Transactions on Design and Manufacturing* and *Discrete Event Dynamic Systems*.

Dr. Luh is the founding Editor-in-Chief of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. From 1999 to 2003, he was the Editor-in-Chief of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION.