[10] D. D. Chamberlain and R. F. Boyce, "SEQUEL: A structured english query language," in Proc. ACM SIGMOD Workshop Data Description, Access, Control, Ann Arbor, MI, 1974.
[11] D. Greenblatt and J. Waxman, "A study of three database query languages," in *Databases: Improving Usability and Representativeness*, B. Shneiderman, Ed. New York: Academic, 1978, pp. 76–87.
[12] C. Welty and D. W. Stemple, "Human factors comparison of a procedural and nonprocedural query language," *ACM Trans. Database Syst.*, vol. 6, no. 4, pp. 626–649, 1981.
[13] H. J. Kim, H. F. Korth, and A. Silberschatz, "PICASSO: A graphical query language," *Softw.—Pract. Exp.*, vol. 18, no. 3, pp. 169–203, 1988.
[14] M. M. Zloof, "Query-By-Example: A data base language," *IBM Syst. J.*, vol. 16, no. 4, pp. 324–343, 1977.
[15] V. M. Markowitz and A. Shoshani, "Abbreviated query interpretation in entity-relationship oriented databases," in *Proc. 8th Int. Conf. Entity-Relationship Approach*, F.H Lochovsky, Ed., 1989, pp. 40–58.
[16] H. C. Chan, K. K. Wei, and K. L. Siau, "User-Database interface: The effect of abstraction levels on query performance," *MIS Quart.*, vol. 17, no. 4, pp. 441–464, 1993.
[17] W. J. Jih, D. A. Bradbard, C. A. Snyder, and N. G. A. Thompson, "The effects of relational and entity-relationship data models on query performance of end-users," *Int. J. Man-Mach. Stud.*, vol. 31, pp. 257–267, 1989.
[18] J. S. Davis, "Experimental investigation of the utility of data structure and ER diagrams in database query," *Int. J. Man-Mach. Stud.*, vol. 32, pp. 449–459, 1990.
[19] D. Batra, J. A. Hoffer, and R. P. Bostrom, "Comparing representations with relational and EER models," *Commun. ACM*, vol. 33, no. 2, pp. 126–139, 1990.
[20] S. L. Jarvenpaa and J. J. Machesky, "Data analysis and learning: an experimental study of data modeling tools," *Int. J. Man-Mach. Stud.*, vol. 31, pp. 367–391, 1989.
[21] D. Batra, "A framework for studying human error behavior in conceptual database modeling," *Inform. Manage.*, vol. 25, pp. 121–131, 1993.
[22] D. Batra and M. K. Sein, "Improving conceptual database design through feedback," *Int. J. Human-Comput. Stud.*, vol. 40, pp. 653–676, 1994.
[23] P. Reisner, "Human factors studies of database query languages, a survey and assessment," *Comput. Surv.*, vol. 13, no. 1, pp. 13–31, 1981.
[24] H.C. Chan, K.K. Wei, and K. Siau, "The effect of a database feedback system on user performance," *Behav. Inf. Technol.*, vol. 14, no. 3, pp. 152–162, May–June 1995.
[25] ——, "A system for query comprehension," *Inf. Softw. Technol.*, vol. 39, pp. 141–148, 1997.
[26] H. Chan, K. Siau, and K. Wei, "The effect of data model, system and task characteristics on user query performance—An empirical study," *DATA BASE Advances Inform. Syst.*, vol. 29, no. 1, pp. 31–49, Winter 1998.
[27] K. L. Siau, K. P. Tan, and H. C. Chan, "A CASE tool for conceptual database design," *Inform. Softw. Technol.*, vol. 34, no. 12, pp. 779–786, December 1992.
[28] K. L. Siau, H. C. Chan, and K. P. Tan, "Visual knowledge query language," *IEICE Trans. Inform. Syst.*, vol. E75-D, no. 5, pp. 697–703, 1992.
[29] T. Connolly, C. Begg, and A. Strachan, *Data Systems—A Practical Approach to Design, Implementation and Management*. Reading, MA: Addison-Wesley, 1996.
[30] J. Thomas and J. Gould, "A psychological study of query by example," in *Proc. National Computer Conf.*, 1975, pp. 439–445.
[31] H. C. Chan, K. K. Wei, and K. L. Siau, "An empirical study on end-users' update performance for different abstraction levels," *Int. J. Human-Comput. Studies*, vol. 41, pp. 309–328, 1994.
[32] A. F. Borthick, P. L. Bowen, S. T. Liew, and F. H. Rohde, "The effects of normalization on end-user query errors: An experimental evaluation," *Int. J. Accounting Inform. Syst.*, vol. 2, pp. 195–221, 2001.
[33] J. M. Boyle, K. F. Bury, and R. J. Evey, "Two studies evaluating learning and use of QBE and SQL," in *Proc. Human Factors Society 27th Annu. Meeting*, 1982, pp. 663–667.
[34] K. S. Suh and A. M. Jenkins, "A comparison of linear keyword and restricted natural language data base interfaces for novice users," *Inf. Syst. Res.*, vol. 3, no. 3, pp. 252–272, 1992.
[35] C. Liao and P. C. Palvia, "The impact of data models and task complexity on end-user performance: An experimental investigation," *Int. J. Human-Comput. Studies*, vol. 52, pp. 831–845, 2000.
[36] K. Siau, "Informational and computational equivalence in comparing information modeling methods," *J. Database Manage.*, vol. 15, no. 1, pp. 73–86, 2004.
[37] D. George and P. Mallery, *SPSS for Windows Step by Step. A Simple Guide and Reference*. Boston, MA: Allyn & Bacon, 2001.
[38] K. Siau, "Information modeling and method engineering: A psychological perspective," *J. Database Manage.*, vol. 10, no. 4, pp. 44–50, 1999.

# Internet Scheduling Environment With Market-Driven Agents

Benjamin P.-C. Yen and Owen Q. Wu

***Abstract*—This paper describes a new generation scheduling paradigm, the Internet scheduling environment. It is formed by a group of Internet scheduling agents which share computational resources to solve scheduling problems in a distributed and collaborative manner. We propose a migration scheme to transform existing standalone scheduling systems to Internet scheduling agents that can communicate with each other and solve problems beyond individual capabilities. To coordinate computational resource collaboration among agents, we introduce the market-based control mechanism in which self-interested agents initiate or participate in auctions to sell or buy scheduling problems. Efficient allocation of computational resources is achieved through the auctions. This paper also describes a prototype Internet scheduling environment named LekiNET, which is migrated from LEKIN®, a flexible job shop scheduling system. The experiments on the LekiNET testbed demonstrate that the agent-based market-driven Internet scheduling environment is feasible and advantageous to future scheduling research and development.**

***Index Terms*—Agent, distributed resource collaboration, Internet scheduling environment, market-based control.**

## I. Introduction

### A. Needs for Distributed Resource Collaboration

Scheduling problems abound in manufacturing factories, transportation systems, hospitals, publishing houses, and so on. As demands on businesses become greater, companies are faced with increasingly complex tasks.

For example, in British Aerospace's largest factory in Broughton, U.K., there are around 2000 staff producing 180 sets of Airbus wings and 40 sets of Hawker 800 fuselages and wings per year. They are constantly seeking improvement in production schedules that could bring substantial savings in reducing work-in-process inventories and late deliveries.

For another example, S&A Food's Derby center of operations employs some 1000 staff to produce 1.1 million meals—or 650 tons of prepared food—per week. The schedules must be dynamically updated in response to the instant change of demands and priorities. The schedulers must also be able to answer what-if questions to prepare ahead for

Manuscript received January 1, 2000; revised February 13, 2003 and October 12, 2003. This work was supported in part by Hong Kong Government RGC under Grant 6076/00E. This paper was recommended by Associate Editor Y. Narahari.

B. P.-C. Yen is with the Faculty of Business and Economics, University of Hong Kong, Hong Kong.

O. Q. Wu is with the Sauder School of Business, University of British Columbia, Vancouver, BC V6T 1Z2 Canada.

Digital Object Identifier 10.1109/TSMCA.2003.822273

extreme situations. This adaptability to dynamic situations is also the key requirement for transportation and hospital scheduling systems.

Some special scheduling requirements also arise in practice. At Boots Contract Manufacturing (BCM), for instance, if one product has a blue fill and another order has a white fill, it is much more efficient to do the white fill product first, otherwise the vessel would have to be washed between the two processes. The solution to this problem is not complex but not every scheduling system has the solution method built in. Should such needs arise, can we solve the problem without upgrading or installing a new system?

No scheduling system works equally well in different industrial situations; selecting the right system for the business is no easy task. Moreover, installation of scheduling systems is costly. According to a recent vehicle routing software survey [12] covering 24 software used in various industries, the installation support cost is about $100 to $285 per hour, and typical time needed for installation ranges from one to ten days. This amounts to an $8 000 installation cost on average. In addition, the software alone costs $10 000 or more. Other costs including maintenance and hardware expenses.

In this paper, we ask the following question: can the existing scheduling systems communicate with each other, share computational resources and solve scheduling problems collaboratively? We try to demonstrate that efficient computational resource collaboration is feasible and could be a solution to many of the above issues. For those small manufacturers or startup companies, computational resource collaboration also eases their budgets in purchasing and installing the scheduling systems. For people working on planning and forecasting tasks, the collaboration of scheduling systems allows them to check the feasibility of their plans and forecast the expected performance under new situations.

Scheduling problems have received considerable attention in the area of operations research [18]. The solution techniques range from centralized approaches (analytic methods, heuristic search, meta-heuristic methods, etc.) to various decentralized approaches (agent-based architecture using contract net protocol, auction protocol, etc.). The potential value of integrating these approaches has been discussed recently [22]. Distributed resource collaboration can achieve a flexible and scalable integration without actually modifying the core of existing systems.

### B. Feasibility of Distributed Resource Collaboration

The first obvious requirement of distributed resource collaboration is the existence of such resources. System researchers have created many scheduling systems over the past decades, e.g., OPAL [1], TOSCA [2], intelligent scheduling systems [3], SONIA [6], DART [7], current research and development includes interactive scheduling systems [8], ISIS [9], ReDS [11], mixed-initiative scheduling systems [13], [14], CUISE [18], OPIS [23], reactive scheduling systems [24], etc.

The second requirement of distributed resource collaboration is to unify the scheduling problem description and build a common standard language so that all the existing scheduling systems are able to communicate. This paper describes the design and development of a communication environment—the *Internet scheduling environment* (ISE).

The reader may ask why we focus on the scheduling area, and whether the distributed resource collaboration is feasible in a more general sense that we can construct other types of problem-solving environment. General problem-solving environment could be feasible provided that systems are highly intelligent so that right resource can be identified to solve right problems. However, to design the general communication language and implement advanced agent analysis ability might cost more than the benefit brought by the environment. Scheduling problems are well-defined and well-describable. Scheduling area has abundant computational resources and human expertise. Moreover, with the advent of the Internet, some scheduling systems are already made remotely accessible on the Internet. Resource collaboration is thus desirable for the scheduling area.

### C. Definition of Internet Scheduling Environment

General definition of intelligent agents can be found in Wooldridge and Jennings [26]. In this paper, we define the ISE as a scheduling system that can reach out for scheduling resources on the Internet to solve problems, and can communicate with other Internet scheduling agents to share resources and solve problems collaboratively. An ISE is defined as a scalable network of Internet scheduling agents that flexibly collaborate to solve scheduling problems that may be beyond individual capabilities.

A noteworthy issue is that the agent in this paper does not represent machine, tool or other physical entity. The latter notion of agent is widely used in the literature on agent-based scheduling systems (see [4], [16], and [21] for examples; see also [22] for discussion). We treat each system as an agent and build the relations between the systems. The notion of collaboration in this context is thus different from that in the agent-based scheduling system literature. The latter refers to collaboration as the teamwork of different parties solving a single scheduling problem within an enterprise. This type of collaboration usually contains strongly related parties that team up to solve a particular problem. Whereas we emphasize the *dynamic nature* of the collaboration among scheduling agents:

1) the collaboration is dynamically established for problem-solving and terminated after it is finished;
2) parties play dynamic roles in collaboration to solve various scheduling problems rather than one single problem;
3) the whole collaborative community evolves dynamically with parties entering and leaving continually.

The outline of this paper is as follows. Section II proposes a migration scheme that transforms existing scheduling systems to Internet scheduling agents and designs a market mechanism that controls computational resource collaboration. Section III discusses the LekiNET testbed, an implementation of the ISE. Section IV describes extensive experimentation to support the feasibility of the ISE. Conclusion follows in the last section.

## II. Design of Internet Scheduling Agents and Formation of the ISE

### A. Migration From Standalone Scheduling Systems to Internet Scheduling Agents

A typical standalone scheduling system has three main modules: user interface, scheduling engine, and database [27], [19]. The key to the migration is to endow the systems with agent features. Genesereth and Ketchpel [10] discussed three ways to agentify an existing system: adding an intermediate communication agent, wrapping the system, and rewriting the system. Considering the fact that most scheduling systems do not have built-in communication infrastructure, our migration scheme is to wrap the scheduling engine and attach it to a communication agent, as shown in Fig. 1. The wrapper is thin—it just provides a communication interface for the engine, while all the agent functionalities are implemented outside the wrapper. This structure facilitates migration of different standalone systems, since only the wrappers need to be specifically designed for each system, while a single design of agent internal structure can be adaptable to all.

Users interact directly with the agent. The user interface could be migrated from the original system or developed separately. The agent have decision-making features: if the scheduling tasks are within the local engine's capability, the agent uses the local engine to solve the problem, otherwise he reaches out for other computational resources (see the details in the next section).

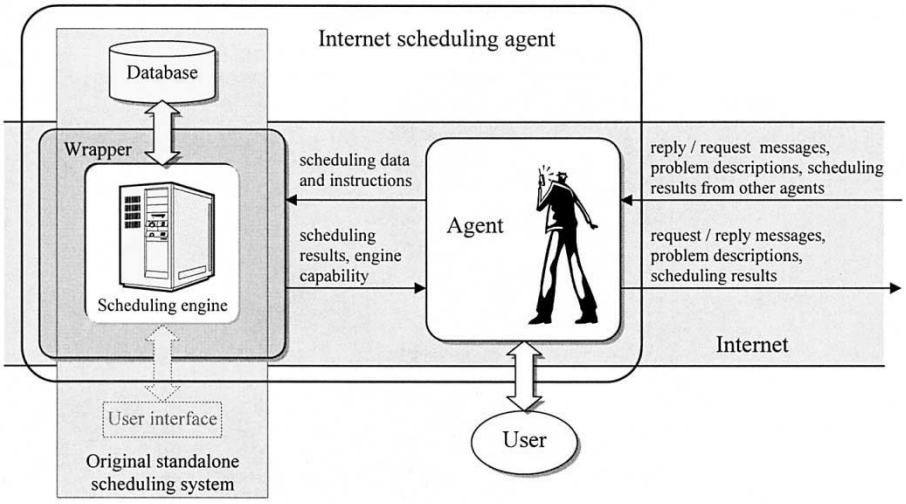


Fig. 1. Migration to an Internet scheduling agent.

The agent communication ability is based on exchanging mutually understandable information, among which scheduling problem description is the most important. Considering both scheduling theory and practice, we not only describe a scheduling problem from its mathematical characteristics but also include real-world situations as problem-solving guidelines. The scheduling problem description can be written as follows.

```
scheduling problem
{
 problem category
 additional constraints
 scheduling data
 initial solution
 maximum allowed computing time (urgency)
 priority level (priority)
 reward and penalty (cost)
 other user requirements and preferences
}
```

The scheduling problem category (e.g., the $\alpha|\beta|\gamma$ notation in manufacturing scheduling [18]) and additional constraints describe the basic mathematical characteristics of scheduling problems. User requirements (e.g., urgency, priority, and cost) are explicitly included in the description. Initial solution may also be specified if the user hopes to make improvement on an existing solution.

### *B. Market-Based Control of Computational Resources Collaboration in the ISE*

Computational resources include executable scheduling algorithms and hardware for running them. Computational resource collaboration is to decide who solve what and when. The distributiveness, dynamics, heterogeneity, and information asymmetry render the centralized control method extremely difficult. The market-based control approach is motivated by certain features of markets, including decentralization, interacting agents, and resource allocation. Clearwater [5] brought together the research on market-based control from diverse fields and addressed that market-based control is a paradigm for controlling complex systems that would otherwise be very difficult to control, maintain or expand.

Price, reward, and penalty are the key interaction instruments in real markets, so do the markets in the ISE. Whereas the market mechanisms in the ISE serve as controls for distributed resource collaboration, and no real monetary transaction will be made in the ISE. (Some service fee may apply if the agent becomes a commercial product.)

The customer submits his scheduling problem to an agent to solve. Usually, the customer has requirements on multiple performance measures, e.g., makespan, number of late jobs, computing time,[1] etc. We write the customer requirements as a set of the following inequalities:

$$M_i \leq r_i, \quad i = 1\text{–}n \tag{1}$$

that is, the $i$th performance measure $M_i$ is required to lie under an upper bound $r_i$. The customer has various preferences of how the scheduling task must be performed. He may allow or disallow contracting the problems to other agents. He may ask multiple agents to "compete" in solving the same problem, and the solution that best meets the requirements will be honored.

Let $p$ be the agreed price of solving the scheduling problem. In our current implementation, $p$ is exogenously determined based on the problem category and size.[2] If all the requirements are met, the customer pays the full price $p$, otherwise he imposes penalty (specified in problem descriptions) on the agent. The penalty $u_i$ is a function of the shortfall of the performance $i$. An example of simple linear penalty function is

$$u_i = \alpha_i \max(m_i - r_i, 0) \tag{2}$$

where $m_i$ is the actual performance, $r_i$ is the required upper bound, and $\alpha_i$ proxies the stringency of the requirement. The total penalty is bounded by $p$. The final payment $P$ is thus

$$P = \max\left(p - \sum_i u_i, 0\right). \tag{3}$$

As mentioned earlier, all these prices and penalties are market instruments used to control the resource collaboration. No real transaction actually occurs, but the agents are programmed to be "revenue" maximizers. They decide when to solve the problem on his own, when to initiate an auction to sell the problem to others, and when to reject the task. The following simple threshold-value method can be used in

[1]Computing time measures the timespan of the problem-solving procedure. This performance measure is generally not included in the scheduling literature.

[2]In the auction mechanisms discussed soon after, the bids and selling prices are endogenously generated from the auction processes. Future research could make price *p* endogenously depend on the customer demands and agents' availability.

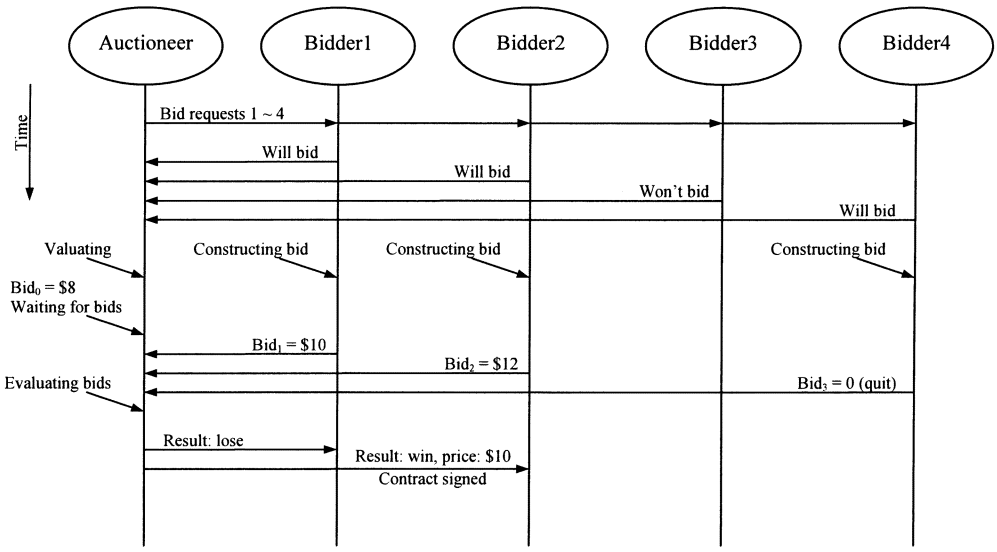


Fig. 2. A typical auction session in the ISE.

decision-making processes and is currently implemented in our prototype. When the agent receives a customer request, he estimates the expected revenue (equals to the payment $P$) if solving the problem on his own. If contracting is disallowed by the customer and the revenue-price ratio is below a threshold value (e.g., 0.2), then the task is considered unworthy of time and effort, and thus rejected. If the ratio is above the threshold, the task is accepted and solved locally. When contracting with other agents is allowed, if the revenue-price ratio is above a threshold value (e.g., 0.95),[3] the task is immediately accepted and solved locally. Below the threshold, the agent will look for other potential solvers by initiating an auction. If a better solver is found, he may sell the task at a price higher than that would be if he solves the problem on his own. Through the auction, the best solver is identified and the efficient resource collaboration is achieved.

An important measure of the advantage of the ISE over the standalone systems is customer satisfaction or, equivalently, dissatisfaction. Customer dissatisfaction is incurred when the requirements are not met or the request is rejected.

$$D = \begin{cases} D_a(m, r), & \text{if accepted} \\ D_r, & \text{if rejected.} \end{cases} \tag{4}$$

where $D_a$ may take the same form as the penalty functions, and $D_r$ can take constant or depend on the reasons for rejection.[4]

Four basic types of auctions are widely analyzed in the literature [17], [20]. Among them, the *Vickrey auction* (second-price[5] sealed-bid auction) owns inherent nice properties that make it the best choice for the ISE:

1) in the Vickrey auction, each bidder's strategy is to simply bid his private value without any game-theoretic analysis on other bidders;
2) sealed-bid auctions can be implemented in secure point-to-point communication, while open auctions are more difficult to implement and less secure than Vickrey auctions;
3) the Vickrey auction is Pareto-optimal in private-value case [25]. It guarantees that scheduling tasks go to the agents with the highest values, thus achieving efficient resource collaboration.

Fig. 2 shows the auction mechanism of the ISE.

1) First, the auctioneer requests some selected agents for bids. The bid requests contain short problem descriptions with no massive data.
2) Each bidder immediately uses the threshold method to make participation decision. He will not participate in the auction if he is incapable of solving the problem or he is currently busy (e.g., he gets several tasks waiting for solving).
3) Then each participant agent constructs a bid based on the price, customer requirements and engine capability. He can either submit an effective bid $(>0)$ indicating his intention to buy the task, or an ineffective bid $(=0)$ to quit the auction. Meanwhile, the auctioneer constructs his private value of the task (in the figure, $\mathrm{Bid}_0 = \$8$).[6]
4) Finally, the auctioneer evaluates the bids received and announces the auction results. The scheduling task is then contracted to the winner. The contract can be canceled later on if situation changes (e.g., the contractor wins other more valuable auctions and cancels the less valuable).

The market-based ISE are easily scalable because addition or deletion of agents will not change the complexity of the decision problem facing any other agent. Since the environment changes dynamically, it is more efficient to maintain the agent identity information on a separate server than to broadcast messages to the ISE. The name list servers are used for this purpose. On entering/leaving the ISE, an agent registers/deregisters on one or more name list servers. By querying a name

[3]We choose threshold 0.95, considering the tradeoff between the time spent on holding an auction and the probability of finding a better solver. The threshold should also depend on the type of the problem, but for the ease of implementation we use a constant.

[4]There are three possible reasons for rejection:

1) the agent is incapable of solving the problem and contracting is disallowed;
2) the agent is incapable of solving the problem and cannot find any other contractor;
3) find some contractors but later the they canceled contracts.

[5]In the case of multiple agents competing in solving the same problem, the multiunit Vickrey auction is used, in which *m* identical tasks are sold to the *m* highest bidders at a price equal to the $m$ + 1st bid.

[6]The auctioneer can announce a reserve price, below which he will not sell out the task. Riley and Samuelson [20] showed that an optimal reserve price that maximizes the auctioneer's expected gain is strictly greater than his private value, resulting in inefficient task allocation. Nevertheless, computerized agents are under our control and we can force the auctioneer to set the reserve price equal his private value.

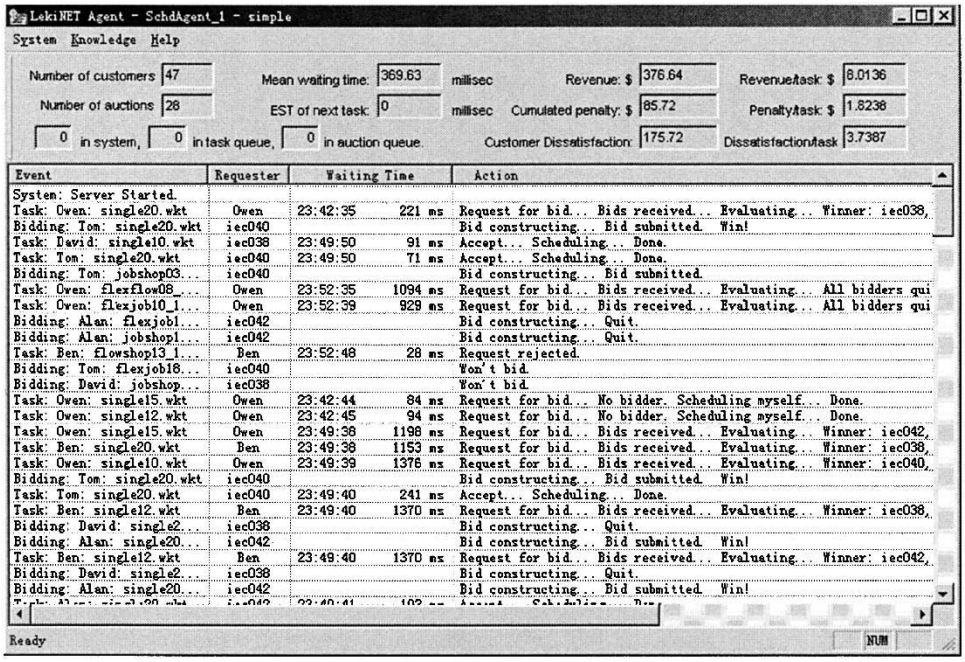


Fig. 3. Main window of the LekiNET agent.

list server, an agent gets to know a group of potential collaborators, to whom bid requests will be sent.

The market-based control mechanisms in this section and implemented in the next are preliminary. More advanced techniques, such as fuzzy logic and neural network, will enhance the mechanism in the future.

## III. LekiNET: An Implementation of the ISE

LekiNET is a prototype ISE implemented in the Information System Lab of the Hong Kong University of Science and Technology. The LekiNET is migrated from LEKIN, a standalone manufacturing scheduling system developed in our previous research.[7]

The communication infrastructure of the LekiNET is implemented using the common object request broker architecture (CORBA), a programming architecture that enables interoperability among distributed applications. The main implementation tools are Inprise VisiBroker 4.0 and Microsoft Visual C++ 6.0. The name list server discussed at the end of Section II is provided by the Naming Service in CORBA. We have installed the LekiNET agents on 40 computers in the Lab., forming an ISE testbed. Fig. 3 shows the main window of the LekiNET agent. The upper part of the window shows the agent status. The lower part is the event log, recording the occurrence time and duration of each event, and actions taken in each. These log data are analyzed in Section IV. We also built the LekiNET client program that interacts directly with customers and accesses the ISE. The client program accepts users' input (including problem description, requirements, penalty terms) through friendly interfaces migrated from the original LEKIN. The client then connects to LekiNET agents, and displays returned schedules in desired format, e.g., Gantt charts.

## IV. Experiments on the LekiNET Testbed

We conducted some preliminary experiments to show that the distributed scheduling resource collaboration is feasible on the LekiNET testbed. The experiments also exhibit agents' opportunistic and goal-directed behavior in the ISE.

[7]See http://www.stern.nyu.edu/om/software/lekin/index.htm for more information.

TABLE I
Configuration of the Agents in the Experiment

| Agent name | Scheduling engine [a] | CPU (MHz) | RAM (MB) |
|---|---|---|---|
| S | Simple scheduler | P-II 400 | 64 |
| P | Parallel machine specialist [b] | P-II 400 | 64 |
| F | Flow shop specialist | P-II 400 | 64 |
| J | Job shop specialist | P-II 400 | 64 |
| F&J | Flow shop and job shop specialist | P-II 400 | 64 |
| FF | Flow shop and flexible flow shop specialist | P-II 400 | 64 |
| FJ | Job shop and flexible job shop specialist | P-II 400 | 64 |
| G | Generalist [c] | P-II 400 | 64 |
| S2 | Simple scheduler | P-200 | 32 |
| F2 | Flow shop specialist | P-III 650 | 128 |
| J2 | Job shop specialist | P-200 | 32 |
| G2 | Generalist | P-III 650 | 128 |

[a] The scheduling algorithms are adopted from LEKIN.
[b] A specialist is a scheduling engine with special algorithms designed for particular types of scheduling problems.
[c] A generalist is a specialist in any type of problems.

### A. Configuring the LekiNET Testbed

Our main experiment uses 12 heterogeneous agents, as shown in Table I. The first eight agents are configured with the same hardware, and the rest four are different. This allows for comparison of hardware effect, an important aspect of computational resources.

In a real ISE, scheduling problems can be of very large scale; a single problem may keep an engine occupied for tens of hours. Significant

network delays also creep in if communications involve agents located at far off places. These include delays due to inviting bids, transferring data, collecting results, etc. To simulate the real ISE, we scale down the time: typical problem-solving needs just a few seconds to five minutes; customers send requests every other a few minutes, and network delays are minimum in the Lab. Specifically, the input to the testbed is as follows. Each agent serves three *fixed* customers who *continually* send him scheduling problems *randomly* drawn from a *problem pool*.

1) Customers do not shift to different agent during the experiment. There are 36 customers in total, and each agent gets three. Customers are all identical so that no bias is introduced.
2) Each customer submits one problem to the agent at a time. After the results are returned, the customer waits for a random period of time (uniformly distributed between 0 and 1 minute) and then sends another problem. He behaves so throughout the experiment.
3) The problem pool contains 113 problems: five single machine scheduling problems, ten parallel machines scheduling problems, 37 job shop scheduling problems, 36 flow shop scheduling problems, 11 flexible flow shop scheduling problems, and 14 flexible job shop scheduling problems. Each problem only specifies machine and job settings, while the objectives and customer requirements are randomly generated for each problem-solving request. Thus, the actual number of scheduling problems generated from the pool is far more than 113.

The prices of solving problems are exogenously fixed based on problem category and size (see Section II-B). For example, in job shop scheduling, problems of two machines and less than five jobs are worth \$30; problems of three to five machines and six to ten jobs are worth \$100; problems of more than ten machines and more than ten jobs are worth \$250.

The knowledge base of each scheduling agent contains knowledge about the expected computing time and solution quality for each scheduling algorithm used for each problem type. Solution quality is a relative measure of scheduling algorithm capability which is used to estimate the expected reward or penalty during agents decision-making processes. The best algorithm is of quality 100. For example, the shifting-bottleneck algorithm for minimizing the makespan of a ten-machine ten-job job-shop problem requires 200 s on average, and the solution quality is expected to be 92. The knowledge bases are fixed; learning algorithms are not implemented at this stage.

### B. Experimental Results

With the configuration in the above section, we let all the agents initially idle, and then let 36 customers start sending requests simultaneously. Problem-solving, auctioning and bidding all start at the same time. The experiment is run for three hours. This time length is sufficient to reach a stationary state (see Fig. 9 for example).

*1) Standalone Systems vs. Internet Scheduling Agents:* The first question we ask is whether the collaboration brings value to the agents and customers. To study this, we ran a supplementary experiment with communication disabled. The agent performances under no collaboration are tested under the same customer behavior. Fig. 4 shows the results.

Standalone systems rely on their own scheduling engines and can only make money on manageable tasks, while scheduling agents can make money on unmanageable ones by selling them to others. Even for the manageable tasks, an agent may also consider auctioning them off to gain more profits. Thus, a universal increase in the task average revenues is expected, as shown in Fig. 4(a). Fig. 4(b) shows an almost universal decrease in the task average dissatisfaction—the quality of service has been improved by collaboration.

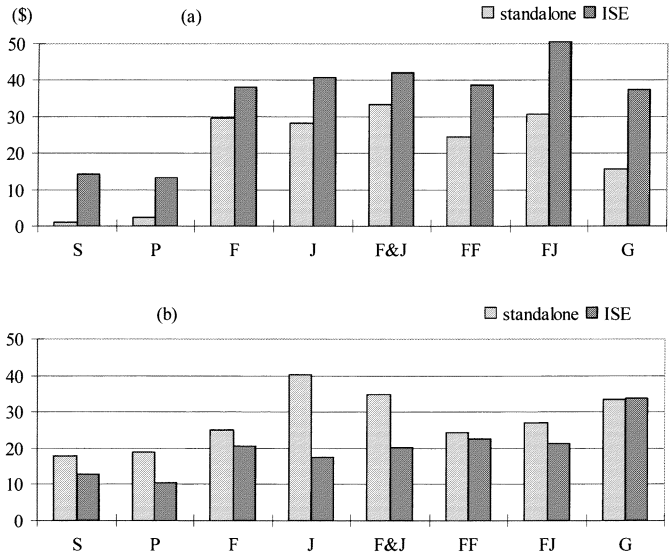


Fig. 4. Standalone systems vs. Internet scheduling agents.

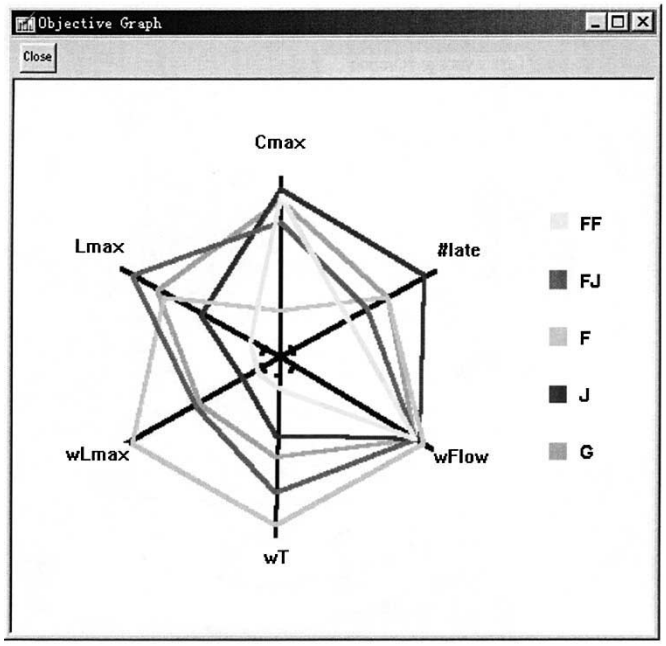


Fig. 5. Multiple agents solving the same problem.

The following experimental result shows the additional benefit brought by multiple agents solving the same problem. The customer can graphically compare multiple solution schedules returned from multiple agents, as shown in Fig. 5. The quality of each schedule is represented by a hexagon—$C_{\max}$ (makespan), $L_{\max}$ (maximum lateness), #late (number of late jobs), etc. The nearer a vertex is to the center, the better the corresponding performance measure. In Fig. 5, the customer originally hopes to minimize the makespan. The agent F has a very good algorithm to minimize the makespan, but overall solution quality may not be satisfactory. Other agents, though not as good as F in minimizing the makespan, may achieve overall satisfactory solutions. On the other hand, the customer usually fails to describe his requirements precisely, thus efforts in trying to meet the customer requirements often fail to satisfy the customer. This happens frequently in real industrial scenarios, where customer' objectives change with what the system can offer. By using multiple

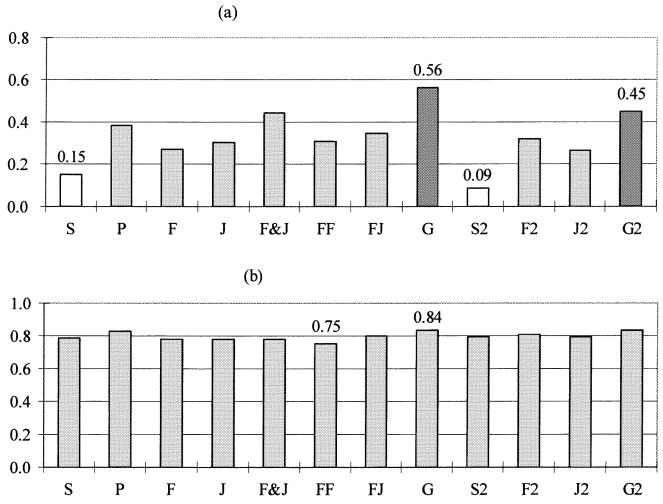


Fig. 6. LekiNET agents serving customers.

solvers, the customer gains much more flexibility in comparing and selecting the best solution. He may even give up his original objective of minimizing the makespan and adopt the solution from FF, which has good overall performance. The quality of service is thus improved.

*2) LekiNET Agents Serving Customers:* A LekiNET agent receives two types of problem-solving requests: *direct requests* sent directly by customers, and *indirect requests* sent by customers to other agents and then contracted to this agent.

Fig. 6(a) shows the proportion of total requests that are indirect requests. This proportion indicates how active an agent is in "helping" (he gets paid to help) others to solve problems. As expected, the two generalizts are the most active helpers, while the two simple agents are the least active ones due to their incapability.

Some direct requests are immediately served by local scheduling engines, while others need to be handled by auctions. The proportion of direct requests that are handled by auctions reflects the agent's activeness in seeking for help. Fig. 6(b) shows that the activeness in seeking for help is almost the same for all agents. There are possibly two reasons: high threshold 0.95 indicates that the agent will initiate an action as long as he expects less than 95% of the full price. Secondly, although the generalizts and some specialists are able to satisfy customers locally, their activeness in helping others makes themselves too busy to serve direct requests. In the future, the agent can balance the efforts in helping others and seeking for help more intelligently.

*3) LekiNET Agents in Auctions:* In this 12-agent-sized experiment, an auctioneer requests all the other 11 agents for bids. Fig. 7(a) compares the bidders behavior using the absolute numbers of auctions during the 3 hours' experiment. Fig. 7(b)–(e) compare the bidders behavior using relative values.

The ratio of participations to requests shows the agents' activeness in bidding. In (b), the agents S, S2, and P are relatively more active in participating in auctions because they do not solve complicated scheduling problems and have more time to participate in auctions. Nevertheless, as shown by the ratio of effective bids to participations in (c), S, S2, and P quit the auctions more often that others because of their inability in solving complex scheduling problems. While G, G2 and some others are comparatively more "serious" bidders: they often submit effective bids because they are capable of treating various scheduling tasks.

Fig. 7(d) shows the winning probability conditioned on participating in an auction. As expected, this winning probability is almost in proportion to the agent capability: the generalizts have more than 50% of chance to win if they participate, while the simple agents participate many but win few. Fig. 7(e) shows the winning probability conditioned

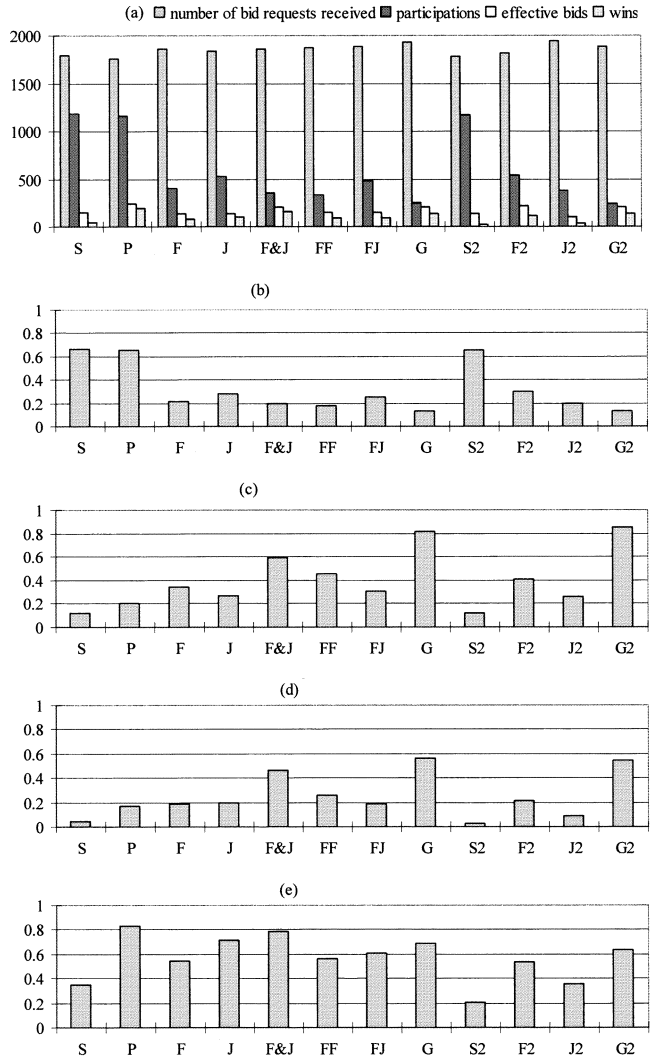


Fig. 7. LekiNET agents in auctions.

on submitting an effective bid. This winning probability exhibits the *competition intensity* of each agent: the more intense the competition, the lower this winning probability. Simple agents have the worst competition environment because they are not competitive in any scheduling problems. While for the agent P, only G and G2 are his major competitors in solving parallel machine problems, but they are often busier than P, thus the competition environment favors P in this experiment, making his winning probability over 80%.

*4) Agents' Wealth:* Fig. 8 compares the wealth of each agent. Regardless of the effect of hardware configurations, we see that the agents F, FF, J, FJ, F&J, and G earn from \$9000–\$16 000, S and P earn around \$5000, whereas, the generalizts did not earn as much as some specialists. The generalizts could do better (might be the best) if they only help others with expensive tasks while auction off cheap ones. But since more intelligent behavior, such as forecasting, is not implemented yet, the generalizts may still buy cheaper tasks and miss more profitable ones, which renders them less wealthy than some specialists.

It can also be seen from the graph that the hardware configurations noticeably affect the agents' wealth. 128 M RAM and Pentium III CPU greatly increase the wealth of F2 and G2, even making F2 the wealthiest, while low configurations clearly damage the agents' wealth.

The task average revenues reflect the agent ability of making money on each task. Fig. 4(a) already shows a universal increase in the task average revenues compared to the standalone systems. Fig. 9 shows that

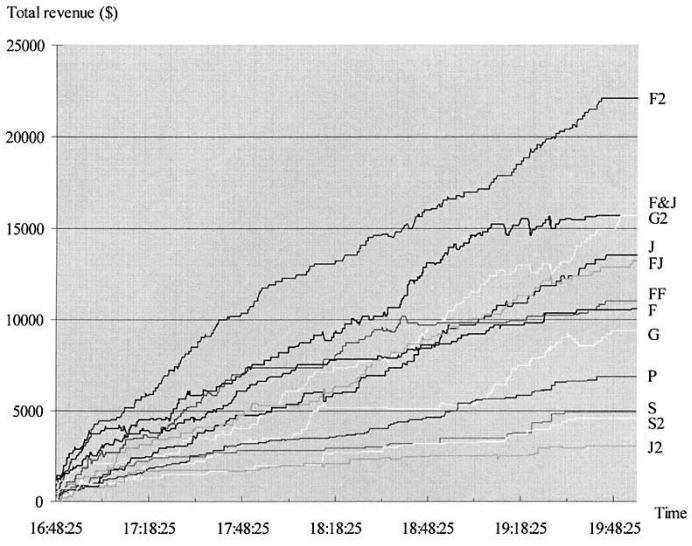


Fig. 8. Evolution of total revenues.

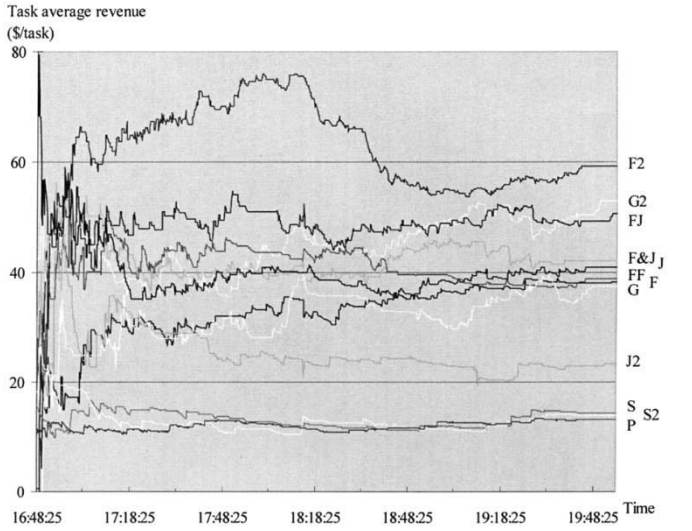


Fig. 9. Evolution of task average revenues.

the task average revenues also go stationary with the lapse of time. It is interesting to note that the agent P makes the least money on each task, but he serves the greatest number of requests (P served 516 requests during the three hours, nearly 14% of the total requests received by the ISE). This is because he wins many cheap parallel machine scheduling tasks, but he needs to pay to the auctioneer for each task bought, thus each task actually brings him little profit. From another point of view, this is P's strategy—he is not capable of making money on complex tasks, but he tries to take the advantage of his favorable competition environment [discussed in Fig. 7(e)] to gain revenue by working on many cheap tasks, thus his total revenue climbs up to nearly $7000. Notice that this strategy is not programmed, but exhibited by the basic decision-making processes that are actually programmed.

The LekiNET agent possesses the following characteristics. *Autonomous*—the agent analyzes all the user requirements and autonomously finds solutions that best meet the requirements. *Responsive*—the agent responds in a timely fashion to the events in the ISE. *Proactive*—the agent exhibits opportunistic and goal-directed behavior in serving customers and auctions. *Social*—the agent communicates with others when appropriate, in order to solve his own problems or help others with their activities. (See [15] and [26] for the notion of these characteristics.)

## V. Conclusion

Scheduling issues have been under research for more than a half century. The advent of the Internet puts a premium on the development of worldwide accessible, communicative and collaborative scheduling systems. The ISE designed and implemented in this paper is an unprecedented attempt to bring all the scheduling computational resources together at a worldwide collaborative level.

The system migration scheme proposed in this paper is an innovative force pushing diverse research efforts into a melting pot. Original standalone scheduling systems thus become active and communicative scheduling agents on the Internet. A community of such agents forms the ISE, under which resource sharing and collaboration among agents becomes a favorable scheduling paradigm. The implementation of the LekiNET shows that this migration is feasible.

The auction market determines whether a particular scheduling task is desirable to an agent, and at what price the task should be contracted. Thus, scheduling tasks always go to those agents who are capable of performing the tasks well, which means computational resources are efficiently allocated. During this procedure, the control emerges from the individual goals of the scheduling agents rather than a central goal imposed from above.

The LekiNET testbed and the experiments on the behavior of the agents demonstrate that the agent-based market-driven ISE is feasible and advantageous to future scheduling research and development. The experiments also highlight a number of perspectives that arouse research interests in the future.

Lastly, the ISE is in its primary stage. The core of the ISE needs to be solidified by a series of steps toward real industrial applications. We sincerely hope that the research in this paper would inspire other researchers to build a real ISE for industrial applications.

## Acknowledgment

The authors gratefully acknowledge two anonymous referees for their thoughtful comments and important suggestions.

## References

[1] C. Badie, G. Bel, E. Basana, and G. Verfaillie, "Operations research and artificial intelligence cooperation to solve scheduling problems: The OPAL and OSCAR systems," in *Proc. 1st Int. Conf. Expert Planning Systems*, Brighton, U.K., 1990, pp. 1–5.

[2] H. Beck, "The management of job-shop scheduling constraints in TOSCA," in *Proc. NSF Workshop Intelligent Dynamic Scheduling Manufacturing Systems*, 1993, pp. 2–14.

[3] R. E. Brown and W. T. Scherer, *Intelligent Scheduling Systems*. Norwell, MA: Kluwer, 1995.

[4] W. T. Chan, D. K. H. Chua, and L. Xiong, "Collaborative scheduling over the internet," *Comput. Aided Civil Infrastruct. Eng.*, vol. 14, no. 1, pp. 15–24, 1999.

[5] S. H. Clearwater, Ed., *Market-Based Control: A Paradigm for Distributed Resource Allocation*, Singapore: World Scientific, 1996.

[6] A. Collinot, C. LePape, and G. Pinoteau, "SONIA: A knowledge-based scheduling system," *Int. J. Artif. Intell. Eng.*, vol. 3, no. 2, pp. 86–94, 1988.

[7] S. E. Cross and E. Walker, "DART: Applying knowledge based planning and scheduling to crisis action planning," in *Intelligent Scheduling*, M. Zweben and M. Fox, Eds. San Mateo, CA: Morgan Kaufmann, 1994.

[8] J. S. Davis and J. J. Kanet, "Production scheduling: An interactive graphical approach," *J. Syst. Softw.*, vol. 38, no. 2, pp. 155–163, 1997.

[9] M. S. Fox and S. F. Smith, "ISIS—A knowledge-based system for factory scheduling," *Expert Syst.*, vol. 1, no. 1, pp. 25–49, 1984.

[10] M. R. Genesereth and S. P. Ketchpel, "Software agents," *Commun. ACM*, vol. 37, no. 7, pp. 48–53, 1994.

[11] K. C. Hadavi, "ReDS: A real time production scheduling system from conception to practice," in *Intelligent Scheduling*, M. Zweben and M. Fox, Eds. San Mateo, CA: Morgan Kaufmann, 1994.
[12] R. W. Hall, "Vehicle routing software survey," *OR/MS Today*, Feb. 2002.
[13] D. W. Hildum, N. M. Sadeh, T. J. Laliberty, S. F. Smith, J. McA'Nulty, and D. Kjenstad, "Mixed-initiative management of integrated process-planning and production-scheduling solutions," in *Proc. Artif. Intell. Manufacturing Research Planning Workshop*, 1996, pp. 71–80.
[14] W.-L. Hsu, M. J. Prietula, G. L. Thompson, and P. S. Ow, "A mixed-initiative scheduling workbench: Integrating AI, OR, and HCI," *Dec. Support Syst.*, vol. 9, no. 3, pp. 245–257, 1993.
[15] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Auton. Agents Multi-Agent Syst.*, vol. 1, pp. 7–38, 1998.
[16] K. H. Kim, J. W. Bae, J. Y. Song, and H. Y. Lee, "A distributed scheduling and shop floor control method," *Comput. Ind. Eng.*, vol. 31, no. 3–4, pp. 583–586, 1996.
[17] P. Klemperer, *The Economic Theory of Auctions*. Cheltenham, U.K.: International Library Critical Writings Economics, Edward Elgar, 2000, vol. I & II. 113.
[18] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[19] M. Pinedo and B. P. Yen, "On the design and development of object-oriented scheduling systems," *Ann. Oper. Res.*, vol. 70, pp. 359–378, 1997.
[20] J. G. Riley and W. F. Samuelson, "Optimal auctions," *Amer. Econ. Rev.*, vol. 71, no. 3, pp. 381–392, 1981.
[21] J. Sauer, "Integrating transportation in a multi-site scheduling environment," in *Proc. 33rd Annu. Hawaii Int. Conf. System Sciences*, vol. 2, 2000, p. 9.
[22] W. Shen, "Distributed manufacturing scheduling using intelligent agents," *IEEE Intell. Syst.*, vol. 17, pp. 88–94, 2002.
[23] S. F. Smith, "OPIS: A methodology and architecture for reactive scheduling," in *Intelligent Scheduling*. San Mateo, CA: Morgan Kaufmann, 1994.
[24] S. F. Smith and O. Lassila, "Configurable systems for reactive production management," in *Knowledge-Based Reactive Scheduling*, E. Szelke and R. M. Kerr, Eds. New York: Elsevier, 1994, pp. 93–106.
[25] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *J. Finance*, vol. XVI, pp. 8–37, 1961.
[26] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, 1995.
[27] B. P.-C. Yen, "On the Architecture of Object-Oriented Scheduling System," Ph.D. dissertation, Dept. Industrial Eng. Oper. Res, Columbia Univ., New York, 1995.

## Correction to "Elementary Siphons of Petri Nets and Their Application to Deadlock Prevention in Flexible Manufacturing Systems"

In the above paper [1], there is an error on pgs. 50–51. For clarification, the references are reprinted as follows

[8] Z. Banaszak and B. H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. Robot Automat.*, vol. 6, pp. 724–734, Dec. 1990.
[27] F. Tricas and J. Martinez, "An extension of the liveness of concurrent sequential processes competing for shared resource," in *Proc. IEEE Int. Conf. System, Man, Cybernetics*, Vancouver, BC, Canada, 1995, pp. 3035–3040.

### REFERENCES

[1] Z. Li, M. Zhou, and D. Wang, "Elementary Siphons of Petri Nets and Their Application to Deadlock Prevention in Flexible Manufacturing Systems," *IEEE Trans. Syst., Man, Cybern. A.*, vol. 34, pp. 38–51, Jan. 2004.

Manuscript received September 1, 2002; revised April 28, 2003. This work was supported in part by the National Defense Research Foundation of China, Youth Scientific Research Workstation in Xidian University, and the New Jersey Commission on Science and Technology. This paper was recommended by Guest Editors M. D. Jeng and M. P. Fanti.

Z. Li is with the School of Electro-Mechanical Engineering, Xidian University, Xi'an, 710071 China. He is also with Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, M5S 3G4 Canada (e-mail: zhwli@xidian.edu.cn).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA and also with Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China (e-mail: zhou@njit.edu).

Digital Object Identifier 10.1109/TSMCA.2004.824363