Requirement-Based Bidding Language for Agent-Based Scheduling

Chun Wang, *Member IEEE*

***Abstract*—This paper presents a requirement-based bidding language for agent-based scheduling. The language allows agents to attach their valuations directly to scheduling performance requirements. Compared with general bidding languages, the proposed one reduces agents' valuation and system's communication complexities. In addition, it results in efficient winner determination problem models. Experimental results show that the requirement-based language exhibits superior winner determination performance in terms of problem-solving speed and scalability.**

***Note to Practitioners*— e-Markets, such as auctions, are useful tools for scheduling resources among independent participants (agents). We propose a requirement-based bidding language which concisely and naturally expresses scheduling requirements. The key benefit of adopting this language in the design of agent-based scheduling systems is reduced computational complexities, which makes the systems more responsive to large scale, dynamic and distributed scheduling problems.**

***Index Terms*—Agent-based scheduling, economic based negotiation mechanisms, bidding language**

I. Introduction

Combinatorial auctions have been used as a negotiation mechanism for designing agent-based scheduling systems where self-interested agents compete for the use of resource time [1-3]. Among others, a key component that impacts the systems' computational complexities is the bidding language in which agents express their valuations over scheduling outcomes. Kalagnanam and Parkes reviewed four areas of computational constraints which restrict the space of feasible combinatorial auction mechanisms [4]. Among the four areas, three (i.e., communication complexity, valuation complexity and winner determination complexity) are affected by the design of the bidding language [5]. Based on the structures of their atomic propositions, general bidding languages for combinatorial auctions can be classified into two types: L_B and L_G[6]. L_B languages use bundles of items with associated prices as atomic propositions and combine them using logical connectives. L_G languages allow bids that are logical formulae where items are taken as atomic propositions and combined using logical connectives. As an example, should a bidder desire (for price p) any two adjacent hours in a four-hour window, she can express her preferences in either L_B or L_G. In L_B, she needs to formulate 3 atomic bids and connect them using logical connective XOR to form a compound bid: $\langle\{h_1,h_2\},p\rangle\, XOR\langle\{h_2,h_3\},p\rangle XOR\langle\{h_3,h_4\},p\rangle$. In L_G, such preferences are expressed in a logical bid of the form: $\langle(h_1\wedge h_2)XOR(h_2\wedge h_3)XOR\,(h_3\wedge h_4),\ p\rangle$. While L_G and L_B only allow agents to express their valuations on bundles or formulae of items, web-based bidding interfaces for procurement applications are much more expressive in terms of representing complex side constraints and discount schedules [7, 8]. However, those interfaces are intended to be used by human sourcing experts rather than software agents.

Since L_G and L_B only allow distinct items, in order to apply them to scheduling auctions, the continuous scheduling time line must be discretized, such that the processing time of resources can be converted to a set of distinct time units [1, 2, 9]. If agents need to reveal their full valuations, such as the case in Vickrey-Clarke-Groves (VCG) mechanisms, general bidding languages require valuations on all 2^m-1 time unit bundles, where m is the number of time units available for bidding. In many realistic scheduling settings, to assure time precision, m cannot be too small. Given that the number of bundles to be evaluated grows exponentially in the number of time units, general languages can inflict heavy burdens on the auctions in terms of bids valuation, winner determination, and communication.

This paper presents a *requirement-based bidding language*. Compared with general languages, the proposed language allows agents to attach their valuations directly to the performance requirement of a schedule. By avoiding expressing valuations on a large number of bundles, agents' valuation and communication complexities are reduced accordingly. The proposed language also results in efficient winner determination models which improve problem-solving speed and scalability. In the next section, we present a *scheduling auction* model which extends the factory scheduling model described in [2]. The scheduling auction is a typical job shop scheduling setting. We use this model as the base environment for comparing bidding languages in terms of the complexities they impose on the auction.

II. Scheduling Auction

The scheduling auction consists of a set of agents. Each agent g has a set of jobs J_g to be processed. Each job $j\in J_g$ requires the processing of a sequence of operations $o_{j,k}$ $(k=1\ldots n_j)$. An operation $o_{j,k}$ has a specified processing time $p_{j,k}$, and its execution requires the exclusive use of a designated resource for the duration of its processing. Each job $j\in J_g$ is constrained by a release time r_j by which the job is available for processing, and a deadline d_j by which the job

C. Wang is with Concordia Institute for Information Systems Engineering, Concordia University, 1515 Ste Catherine West, EV 007.649, Montreal, Quebec, H3G 1M8, Canada (phone: 1-514-8482424 ext. 5628; fax: 1-514-8483171; e-mail: cwang@ciise.concordia.ca).

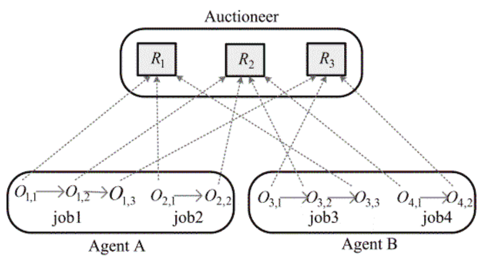

Fig. 1 Example of the Scheduling Auction Model

must be completed. For a feasible schedule with completion time c_j, $r_j < c_j \le d_j$, the agent obtains a value $v_g(c_j) > 0$. For any completion time outside $(r_j, d_j]$, $v_g(c_j) = 0$. $v_g(c_j)$ is determined by the agent's internal mechanism which is language independent. We assume that, for an agent, $v_g(c_j)$ is given for any c_j. There are precedence constraints among operations of a job, but there are no precedence constraints among jobs. An allocation of processing time to jobs forms a schedule for agent g, denoted G_g. An agent's valuation over jobs in G_g is additive, that is $v_g(G_g) = \sum_{j \in G_g} v_g(c_j)$. According to the additive valuation, as long as a schedule completes one job within $(r_j, d_j]$, its value to the agent is positive. The objective of the auction is to maximize social welfare, the sum of $v_g(G_g)$ across all agents.

Fig. 1 shows an example of the scheduling auction with three resources (R_1, R_2, R_3) controlled by the auctioneer. Agent A has job1$(O_{1,1}, O_{1,2}, O_{1,3})$ and job 2 $(O_{2,1}, O_{2,2})$ to be processed. Agent B has job 3 $(O_{3,1}, O_{3,2}, O_{3,3})$ and job 4 $(O_{4,1}, O_{4,2})$ to be processed. Arrows with solid lines represent the precedence constraints between operations; arrows with dotted line link operations to their designated processing resources. The scheduling auction can be an abstract model of various scheduling settings. In a manufacturing firm, for example, each sales agent may have a set of jobs to be processed. They may "compete" with each other for the limited processing resources to satisfy their customers' requirements. The auction can also be seen as a general model of some agent-based scheduling environments described in the literature, in which an agent represents only one job[10].

III. The Requirement-Based Bidding language

In the scheduling auction, an agent's valuation on a schedule depends on the extent to which the schedule satisfies its performance requirement. Ideally, a bidding language should provide the expressiveness which allows agents to explicitly attach their valuations to performance requirements. However, L_G and L_B do not provide such expressiveness since they only allow an item or a bundle of items in their atomic propositions. We present a requirement-based bidding language, namely L_R, to address this limitation of general bidding languages in the domain of scheduling.

As in L_G and L_B, the basic structure of L_R is an *atomic proposition*. L_R atomic proposition is more concise in terms of representing scheduling problems. It consists of a description of the job to be completed, a quality requirement and the price that the agent is willing to pay given the requirement is satisfied. The quality of a schedule can be evaluated by many types of scheduling criteria [11]. In our scheduling auction, we use the completion time of a job as the measure. Formally, an *atomic proposition* can be represented by a 3-tuple <*Job*, *Completion-Time*, *Price*>.

Job specifies a sequence of operations and their required processing time on resources. The release time of the job and other processing constraints are also specified. Deadline of the job is not specified here. It will be expressed in the *Completion-Time*. For many scheduling models, existing general scheduling problem description languages, such as the one proposed in [12], can be used to describe the job specification. Since we focus on resource allocation in this paper, in our scheduling auction model a job is described as a set of resource and processing time pairs. For example, job 3 in Fig.1 can be presented as $((R_3, 23), (R_2, 15), (R_1, 18), r_3 = 5)$, which means it needs to be processed by R_3, R_2 and R_1 in sequence after the release time 5, and the processing times are 23, 15 and 18.

Completion-Time is the time range that constraints the completion of a schedule. For example, if the *Completion-Time* is $(20,40]$, its semantic interpretation is that the job is required to be completed after time 20 and before 40.

Price is the amount of money that the agent is willing to pay given that the schedule of the jobs satisfies Completion-Time requirement. For example, the atomic proposition $\langle job, (20,40], \$100 \rangle$ is interpreted as the agent is willing to pay \$100 if the completion time of the job is within $(20,40]$.

Expressing a simple L_R atomic proposition using L_B or L_G usually requires a compound XOR-bid with a possibly large size. For example, to represent $\langle job,\ (20,40], \$100 \rangle$ in L_B, we need to compute the set of all eligible schedules, denoted Γ_{job}, such that each $s \in \Gamma_{job}$ completes within $(20, 40]$. We then connect the set of schedules as an XOR-bid: $XOR_{s \in \Gamma_{job}} \langle s, \$100 \rangle$, where each $s \in \Gamma_{job}$ is a bundle of time units. The sizes of s and Γ_{job} are usually not small in scheduling auctions with multiple resources and multiple operation jobs, such as the case in the example of Fig.1. If L_G is used, in addition to the XOR-bid, the time units in each $s \in \Gamma_{job}$ need to be joined by $|s|$-1 conjunction connectives.

By connecting L_R atomic bids using XOR, we can express an agent's values on different levels of *Completion-Time*. For example, we can use the following compound bid $\langle job, (80, 100], \$2k \rangle XOR \langle job, (100, 120], \$1.9k \rangle$ to express the valuation: If the job is completed between $(80, 100]$, the agent is willing to pay $\$2k$; if completed within $(100, 120]$, the price is reduced to $\$1.9k$. It is worth to note that atomic bids in each of the three languages can be connected by XOR and/or OR to form compound bids. The key difference of the three bidding languages is the structure of their atomic bids.

IV. Valuation, Communication and Winner Determination Complexities

In this section we compare valuation, communication and winner determination complexities of L_R and L_B. We use the benchmark VCG mechanism as our auction setting. VCG is a type of one-shot auction. It motivates agents to submit their complete valuations truthfully and computes optimal solutions. In addition to the comparison between L_R and L_B, we also analyze the suitability of L_G as a bidding language for the scheduling auction.

A. Valuation Complexity

In the case of L_B, agents have to implicitly express their valuation by attaching values to bundles of time units. In a standard combinatorial auction setting, an agent must evaluate $2^m - 1$ bundles, where m is the number of items sold in the auction. The valuation complexity in L_B is the computation needed to evaluate the bundles of time units. We first define the value of a bundle to an agent. For a bundle B, if a schedule $G_g \subseteq B$, we say G_g is covered by B. In many cases, a bundle can cover several schedules of an agent. We define the value of a bundle as the value of the best schedule which the bundle covers.

Definition 1: *Let Γ be the set of schedules of an agent covered by B. The valuation of the agent on bundle B, denoted $v_g(B)$, is said to be the value of the best schedule $G_g^* \in \Gamma$, such that for any $G_g \in \Gamma$, $v_g(G_g^*) \geq v_g(G_g)$.*

In the scheduling auction, computing the valuation of a bundle to an agent is to solve the problem: Given a set of jobs to be allocated to a bundle of time units, what is the best value that a feasible schedule can possibly achieve? Answering this question is equivalent to solving a job shop scheduling problem with availability constraints (JSPAC), which is NP-hard [13]. This proves:

Proposition 1: *In the scheduling auction with completion time as the performance measure, computing an agent's valuation on a bundle is NP-hard.*

While L_B requires an agent to solve an NP-hard optimization problem to determine its value on a bundle, L_R does not need to evaluate bundles. Since $v_g(c_j)$ is given, valuation complexity in L_R is trivial. It just involves the assignment of $v_g(c_j)$ to c_j for $r_j < c_j \leq d_j$. The maximum number of positive $v_g(c_j)$ is $d_j - r_j$. From agents' perspective, L_R has the advantage of avoiding the hard job shop scheduling problem during the valuation phase. However, this does not mean the complexity of scheduling jobs to resources has been eliminated in a L_R model. In fact, this computation burden is shifted to the auctioneer's winner determination problem (WDP). In L_R WDP, the auctioneer has to determine the winning bids and, at the same time, schedule jobs to resources. However, even with the extra task of job scheduling, as shown later in this section, L_RWDP is still more efficient than the standard L_BWDP in terms of solving speed and scalability.

B. Communication Complexity

Communication complexity is concerned with the size of the messages that must be sent by agents in order to compute the outcome of an auction. In combinatorial auctions, a full valuation requires agents to determine the value of $2^m - 1$ bundles, which is exponential in the number of items. However, not all of them need to be sent to the auctioneer. Since we assume a VCG auction setting, an agent's optimal strategy is to bid for the bundles for which it has a positive value. In the scheduling auction, if a bundle does not cover a feasible schedule for at least one job of an agent, it is called an infeasible bundle to the agent. Those *infeasible bundles* have zero value to the agent and need not be sent. A job j can have a set of many *feasible schedules*, denoted $FS(j)$. For a $G_j \in FS(j)$, let $FB(G_j)$ denote the set of bundles that cover G_j, $FB(G_j) = \{B | B \supseteq G_j\}$, where the universe of discourse is the set of $2^m - 1$ bundles. Also, let $FB(j)$ denote the union of $FB(G_j)$, for all $G_j \in FS(j)$, which is $\bigcup_{G_j \in FS(j)} FB(G_j)$. Note that if an agent only has one job j to be processed, the set of feasible bundles it needs to send is $FB(j)$.

Let's now consider the sizes of $FB(G_j)$ and $FB(j)$. By definition, $FB(G_j)$ contains all super sets of G_j. Its size can be computed from the formula $|FB(G_j)| = 2^{m-|G_j|}$. Since $|G_j|$ is the number of the processing time units required by job j and is a constant for a job, the size of $FB(G_j)$ is $O(2^m)$. Since $FB(G_j) \subseteq FB(j)$, it follows that the size of $FB(j)$ is at least $O(2^m)$, which proves the following proposition.

Proposition 2: *If an agent only has one job to be processed, the number of feasible bundles it needs to send to the auctioneer is at least $O(2^m)$.*

Given agents' additive valuation over jobs, it is also true that if an agent has multiple jobs, its number of feasible bundles is still at least $O(2^m)$. We prove this statement by showing that adding more jobs will not decrease the feasible bundle set (or equivalently, increase the infeasible bundle set) of an agent.

Proposition 3: *Given a set of resource time units, adding a new job will not increase an agent's infeasible bundle set.*

Proof. Let $FB(J_g)$ contains feasible bundles of agent g given the set of jobs J_g. Accordingly $\overline{FB(J_g)}$ contains infeasible bundles. Suppose that after adding a new job j', $\overline{FB(J_g \cup j')}$ will increase. In this case, at least one bundle has to be moved from $FB(J_g)$ to $\overline{FB(J_g \cup j')}$. However, according to the additive valuation of agents, each bundle in $FB(J_g)$ covers a feasible schedule of at least one job in J_g. Therefore no bundle can be moved from $FB(J_g)$ to $\overline{FB(J_g \cup j')}$. The proposition is proved by contradiction. ■

In fact, adding a new job will likely decrease an agent's infeasible bundle set (or increase its feasible bundle set) because a previously infeasible bundle may now be able to cover the newly added job. Compared with L_B, the communication complexity of L_R is much lower. In L_R, an agent only needs to send $|J_g|$ XOR bids each for a job in J_g. Now we take a close look at the scenario where the agent has only one job to complete. To schedule the sequence of operations $(o_{j,1}, \dots, o_{j,n_j})$ in w ($w = d_j - r_j$), three constraints have to be satisfied:

$$S_{j,k} + p_{j,k} \leq S_{j,k+1}, \quad 1 \leq k \leq n_j - 1 \tag{1}$$

$$S_{j,1} \geq r_j \tag{2}$$

$$S_{j,n_j} \leq d_j - p_{j,n_j} \tag{3}$$

where $S_{j,k}$is the starting time of $o_{j,k}$; $p_{j,k}$ is the processing time of $o_{j,k}$. The starting time of an operation could vary in different feasible schedules. By counting the number of combinations of feasible starting time of all operations, we can calculate the number of feasible schedules using the following formula:

$$\sum_{S_{j,1}=0}^{w-p_{j,1}-\cdots-p_{j,n_j}} \sum_{S_{j,2}=p_{j,1}+S_{j,1}}^{w-p_{j,2}-\cdots-p_{j,n_j}} \cdots \sum_{S_{j,n_j}=p_{j,n_j-1}+S_{j,n_j-1}}^{w-p_{j,n_j}} 1 \quad (4)$$

For the set of randomly generated jobs (used in the experiments which we will describe in the next subsection), the average number of feasible schedules for one job is around six hundred thousand and the number increases to a million if we extend the deadlines of jobs by five percent. According to Proposition 3, adding more jobs will likely increase the agent's communication complexity.

C. *Winner Determination Complexity*

Winner determination complexity refers to the amount of computation required to compute the global schedule. We compare the complexities of L_B WDP and L_R WDP through experiments. L_B WDP is a standard combinatorial auction problem (CAP) formulated in[14]. Here, we formulate L_R WDP as a mixed integer program.

1) *L_R WDP Formulation*

Suppose that each agent submits a set of L_R *XOR* bids for its jobs. Bids from all agents form a set J. The WDP involves the selection of a subset of J such that the sum of the values of the selected bids is maximized and that all scheduling constraints are satisfied. To model the competition of jobs for resources, we define $q_{j,k,j',k'}=1$ if two operations $o_{j,k}$ and $o_{j',k'}$ need to be processed on the same resource and $q_{j,k,j',k'}=0$ otherwise. Using the following variables:

$S_{j,k}$ the starting time of the operation k of the bid of job j

$$Z_j = \begin{cases} 1 \text{ if bid of job } j \text{ wins} \\ 0 \text{ otherwise} \end{cases}$$

$$Y_{j,k,j',k'} = \begin{cases} 1 \text{ if } o_{j,k} \text{is performed before } o_{j',k'} \\ 0 \text{ otherwise} \end{cases}$$

The L_RWDP can be formulated as follows.

$$max \sum_{j \in J} Z_j v_g \left(S_{j,n_j} + p_{j,n_j}\right)$$

subject to

$$S_{j,1} \geq r_j Z_j \quad (5)$$

$$S_{j,k-1} + p_{j,k-1} - S_{j,k} \leq H(1 - Z_j) \quad (6)$$

$$S_{j,k} + p_{j,k} - S_{j',k'} + Hq_{j,k,j',k'} + HZ_j + HZ_{j'} + HY_{j,k,j',k'} \leq 4H \quad (7)$$

$$Y_{j,k,j',k'} + Y_{j',k',j,k,} + 2H \geq 1 + HZ_j + HZ_{j'} \quad (8)$$

$$Y_{j,k,j',k'} + Y_{j',k',j,k,} + +HZ_j + HZ_{j'} \leq 1 + 2H \quad (9)$$

$$Y_{j,k,j',k'} \in \{0,1\}, Z_j \in \{0,1\} \text{ and } S_{j,k} \geq 0 \quad (10)$$

where $j, j' \in J$, $j \neq j'$, $1 < k \leq n_j, 1 < k' \leq n_{j'}$; H is a large finite positive number. The set of constraints (5) ensures that the operations of a bid do not start before its release time. The set of constraints (6) ensures that an operation does not start before the previous operation of the same bid is completed. The set of constraints (7) is a set of logical constraints saying: *If* two bids j and j' are selected in the schedule, *and* operations $S_{j,k}$ and $S_{j',k'}$ are to be processed on the same resource ($q_{j,k,j',k'} = 1$), *and* $S_{j,k}$ precedes $S_{j',k'}$ ($Y_{j,k,j',k'} = 1$), *then* $S_{j,k} + p_{j,k} \leq S_{j',k'}$. These constraints ensure that, at most, one operation can be processed by a particular resource at a time, where H is a large positive constant, which is used for the linearization of the logical constraint "*if*". The minimum value of H depends on the problem instance. In general, a $H = max(d_j) + max(p_{j,k})$, where $j \in J$ and $k = 1, \ldots, n_j$, is large enough to enforce the logical "*if*" constraint. Constraints (8) and (9) ensure the values assigned to the two related variables $Y_{j,k,j',k'}$ and $Y_{j',k',j,k,}$ are consistent, that is, *if* $S_{j,k}$ and $S_{j',k'}$ are to be processed on the same resource, *then* $Y_{j,k,j',k'} + Y_{j',k',j,k,} = 1$. Constraints (10) are non-negative and integer constraints.

To prove the NP-hardness of the model, construct a special case by randomly picking an atomic bid from each XOR bid and replacing the XOR bid with this atomic bid. Also assume that all jobs must be scheduled on one resource. In this case, the decision version of the special case is equivalent to the job

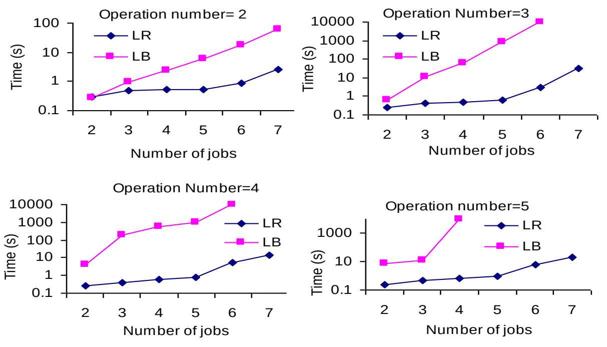

Fig. 2 Runtime of L_B WDP and L_R WDP over jobs

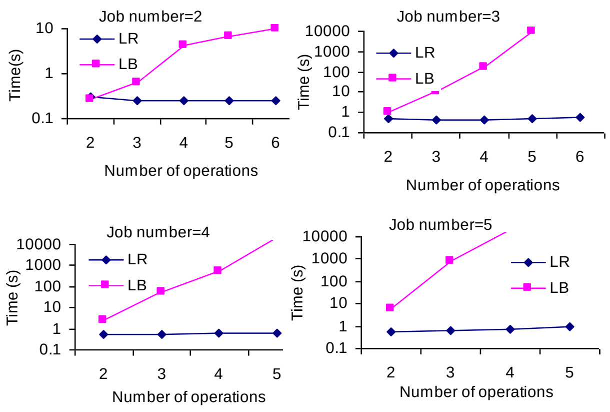

Fig. 3 Runtime of L_B WDP and L_R WDP over operations

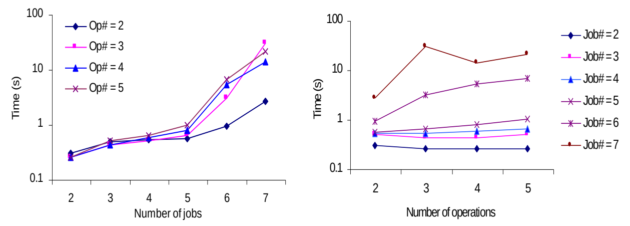

Fig. 4 L_R WDP scalability over jobs and operations

shop interval selection problem, which is NP-complete [15].

2) A Computational Study

We evaluate the complexities of winner determination problems formulated in L_B and L_R through experiments. CPLEX 10.1 was used to solve the winner determination models. We design our scheduling test problems based on a suite of job shop CSP benchmark problems developed in [16]. While the job shop CSP benchmark problems are constraint satisfaction problems, we add a price parameter P to construct the scheduling auction problem set. The price of job j is randomly drown from a uniform distribution on $U\big(du_j, du + du_j\big)$, where du is the average duration of all jobs, and du_j is the duration of job j. A problem set was randomly generated to include different sizes of problems (determined by the number of jobs in a problem and the number of operations in a job). In these problems the number of operations ranges from 2 to 6; the number of jobs ranges from 2 to 7. In each problem instance, the number of resources is equal to the number of operations.

The experiments were conducted on a 2.8 GHz Pentium PC. We first constructed L_BWDP and L_RWDP based on the description of a problem instance. Then we solved the two WDPs using CPLEX 10.1 respectively. In Fig. 2 to 4, each point in each plot is the average runtime for 10 problem instances with the same numbers of jobs and same numbers of operations in each instance.

We present the experimental results from two perspectives: (1) given a fixed number of operations in the problems, how runtime changes when the number of jobs increases and (2) given a fixed number of jobs, how runtime change when the number of operations increases. As shown in Fig. 2, for the first two groups of problems (operation number=2 and operation number=3), the runtime of L_BWDP and L_RWDP are initially close. As the number of jobs increases, the difference increases quickly. For the rest two groups of problems (operation number=4 and operation number=5), L_RWDP is more than 10 times faster than L_BWDP even at the size of 2 jobs. L_BWDP can be 100 to 1000 times slower when the number of jobs reaches 7. Fig. 3 presents the results from a different angle: L_BWDP does not scale well when the number of operations increases. On the contrary, the runtime of L_RWDP is virtually unaffected when the number of operations increases from 2 to 5 in all four groups of problems. The scalability characteristics of L_RWDP are further illustrated in Fig. 4. The scalability of L_RWDP remains good when the number of jobs is less than 5. As the number of jobs increases beyond 5, the scalability of L_RWDP decreases with a higher rate. Along the number of operations L_RWDP scales very well at all job number levels.

D. *Complexities of* L_G

L_B and L_G take an item or a bundle of items as atomic proposition, which does not allow agents to explicitly express their valuation on scheduling performance requirements. Because an agent has to evaluate the $2^m - 1$ bundles in both cases, L_B and L_G are identical in terms of valuation complexity. Compared with L_B, representing a schedule in L_G is rather unnatural. For example, to represent a feasible schedule G_j in L_G, all time units in G_j need to be joined together by $|G_j| - 1$ conjunction connectives. If bundle-level languages such as L_B are used, then we can treat G_j as a bundle without placing any connectives between time units. It is reported in [6] that preferences involving disjunction and sharable resources can be expressed much more compactly in L_G than in L_B. However, this is not the case in the scheduling auction, where agents' valuations exhibit strong complementarities. To represent complementary preferences using L_G, time units in a feasible bundle need to be connected by connective $\wedge$ (conjunction) and the feasible bundles are connected by XOR, which is essentially the same structure used by L_B. Since a large number of conjunction connectives have to be added to join time units in feasible schedules, L_G incurs a higher level of communication complexity in the scheduling auction than L_B does. Based on the previous comparison between L_B and L_R, it can be concluded that L_R has a lower level of communication complexity than L_G does. In terms of winner determination, a direct integer program formulation of the L_G WDP exhibits superior performance than that of L_B for several problem distributions[17]. However, the performance on XOR bids which are required in scheduling auction is not reported in [15]. The idea of the direct L_G formulation is to exploit the specific structure of logically specified bids represented in L_G to solve problems more effectively. However, the bids in scheduling auction are "flat" bundles. Representing them using L_G does not provide scheduling specific structural information that can be exploited by a winner determination algorithm. On the contrary, the requirement-based language preserves the scheduling specific structure, which opens the door to the design of specialized, high-performance scheduling winner determination algorithms. As we have shown in the experiments, even a general optimization package can benefit significantly from the preservation of scheduling specific problem structures.

V. Conclusions

When agent's valuations on schedules are represented in general bidding languages, scheduling specific structural information is lost. Agents have to attach their valuations to "flat" bundles, which leads to drastically increased valuation, communication, and winner determination complexities. We compared the general bidding languages and our requirement-based bidding language in terms of their complexity implications to agent-based scheduling. We show that the requirement-based language provides concise, natural representations of agents' valuations and reduces system's communication complexity. An interesting finding is that although the auctioneer has to solve winner determination and scheduling problems concurrently when adopting L_R, L_R WDP formulated by incorporating scheduling specific modeling techniques can be more efficient than the standard L_BWDP in terms of solving speed and scalability.

Acknowledgements

Thanks to Hamada Ghenniwa and Weiming Shen for their helpful discussions and the anonymous referees for their suggestions.

References

[1] E. Kutanoglu and S. D. Wu, "On combinatorial auction and Lagrangean relaxation for distributed resource scheduling," *IIE Transactions,* vol. 31, pp. 813-826, 1999.

[2] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason, "Auction protocols for decentralized scheduling," *Games and Economic Behavior,* vol. 35, pp. 271-303, 2001.

[3] J. Collins and M. Gini, "Scheduling tasks using combinatorial auctions: The magnet approach," *Business Computing,* vol. 3, p. 263, 2009.

[4] J. Kalagnanam and D. C. Parkes, "Auctions, Bidding and Exchange Design," in *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, D. Simchi-Levi, S. D. Wu, and M. Z. Shen, Eds.: Kluwer Academic Publishers, 2004.

[5] N. Nisan, "Bidding languages for combinatorial auctions " in *Combinatorial Auctions*, Cramton, Shoham, and Steinberg, Eds.: MIT Press, 2006.

[6] C. Boutilier and H. H. Hoos, "Bidding languages for combinatorial auctions," in *Proc. 17th International Joint Conference on Artificial Intelligence,* Seattle, Washington, 2001, pp. 1211–1217.

[7] T. Sandholm, "Expressive Commerce and Its Application to Sourcing: How We Conducted $35 Billion of Generalized Combinatorial Auctions," *AI Magazine,* vol. 28, pp. 45-58, 2007.

[8] T. S. Chandrashekar, Y. Narahari, C. H. Rosa, D. M. Kulkarni, J. D. Tew, and P. Dayama, "Auction-Based Mechanisms for Electronic Procurement," *Automation Science and Engineering, IEEE Transactions on,* vol. 4, pp. 297-321, 2007.

[9] L. Hoong Chuin, Z. J. Zhao, G. Shuzhi Sam, and L. Tong Heng, "Allocating Resources in Multiagent Flowshops With Adaptive Auctions," *Automation Science and Engineering, IEEE Transactions on,* vol. 8, pp. 732-743, 2011.

[10] W. Shen, "Distributed manufacturing scheduling using intelligent agents," *IEEE intelligent systems,* pp. 88-94, 2002.

[11] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*: Prentice Hall, 2002.

[12] S. F. Smith and M. A. Becker, "An ontology for constructing scheduling systems," in *Working Notes of 1997 AAAI Symposium on Ontological Engineering*, Stanford, CA, 1997.

[13] P. Mauguière, J. Billaut, and J. Bouquard, "New Single Machine and Job-Shop Scheduling Problems with Availability Constraints," *Journal of Scheduling* vol. 8, pp. 211-231, 2005.

[14] S. de Vries and R. V. Vohra, "Combinatorial auctions: a survey," *INFORMS Journal on Computing,* vol. 15, pp. 284-309, 2003.

[15] J. M. Keil, "On the complexity of scheduling tasks with discrete starting times " *Operations Research Letters,* vol. 12, pp. 293-295, 1992.

[16] N. Sadeh and M. S. Fox, "Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem," *Artificial Intelligence,* vol. 86, pp. 1-41, 1996.

[17] C. Boutilier, "Solving concisely expressed combinatorial auction problems," in *the Proceedings of Eighteenth National Conference on Artificial Intelligence (AAAI-02)* Edmonton, Alberta, Canada, 2002, pp. 359-366.