Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Bidding Languages for Auction-based Distributed Scheduling

Chun Wang
Concordia Institute for Information Systems Engineering, Concordia University
Montreal, Canada
cwang@ciise.concordia.ca

Hamada H. Ghenniwa
Department of Electrical and Computer Engineering, University of Western Ontario
London, Canada
hghenniwa@eng.uwo.ca

Weiming Shen
Centre for Computer-assisted Construction Technologies, National Research Council Canada
London, Canada
weiming.shen@nrc.gc.ca

***Abstract*—The kind of bidding languages used in combinatorial auctions contributes to various aspects of computational complexities. General bidding languages use bundles of distinct items as atomic propositions associated with logical connectives. When applying these languages to auction-based scheduling, the scheduling timeline needs to be discretized into fixed time units. We show that this discretization approach is computationally expensive in terms of valuation, communication, and winner determination. We present a requirement-based bidding language designed for auction-based scheduling. In the language, bids are specified as the requirements of scheduling a set of jobs, and prices are attached to the job completion times. Without timeline discretization, this language allows the expression of scheduling valuation functions in a natural and concise way, such that valuation and communication complexities are reduced. In addition, it results in efficient winner determination problem models. We have compared the winner determination models formulated using the two types of languages in terms of solving speed and scalability. Experimental results show that the requirement-based language model exhibits superior performance.**

I. INTRODUCTION

In many combinatorial auctions (CAs), the goods to be sold are the processing times of resources, e.g. landing timeslots of airport runways [8], machine processing times of a factory [13], computation and network accessing times of internet resources [1], and the right to use railroad tracks for a period of time [7]. In this class of CAs, agents have jobs that need to be completed during specific time windows and they compete with each other for the resources to schedule their own jobs according to their respective objectives. We may refer this type of CA as auction-based scheduling.

As in other CAs, agents' valuations in auction-based scheduling often exhibit complementarities. For example, due to scheduling constraints, an agent may need to obtain a set of specific combinations of time periods on resources to process its jobs. The complementarities of agents' valuations present particular challenges for the design of bidding languages in terms of expressiveness, conciseness, and naturalness. Logical languages have been proposed to address this challenge [6]. These languages (denoted by L_B) use bundles of items with associated prices as atomic propositions and combines them using logical connectives.

L_B languages target CAs in general. However, they cannot be applied to scheduling problems directly because they are designed for auctioning discrete goods. In scheduling problems, processing time on resources exhibit continuity. In order to apply L_B, the scheduling timeline of resources needs to be discretized into fixed time units and these units are treated as distinct items in L_B [13] [3]. With timeline discretization, agents can express their time related scheduling requirements, such as release times, due dates, indirectly by attaching values to various time units combinations. It will be shown in Section III, that determining the value for a time unit's combination could be a NP-hard optimization problem in certain auction-based scheduling settings. In addition, this timeline discretization approach can generate a large number of items to be sold in the auction if the time windows in question are not small. For example, a one week time window on 10 resources can be discretized into more than 100,000 time units if the time accuracy we need is in minutes (which is a practical requirement in many scheduling domains). Generally speaking, in combinatorial auctions the number of bids is exponential in the number of the items to be sold. A large number of items can inflict heavy burdens on the auction in terms of bids evaluation, communication, and winner determination.

An alternative to the timeline discretization approach is to design languages which allow agents to directly express their time requirements and the values associated. We refer to this type of language as requirement-based language (denoted by L_R). L_R languages enable agents to explicitly express their time-related requirements without specifying the values on the combinations of resource time units. For example, in a train scheduling auction setting [7], Parkes and Ungar designed a requirement-based language which allows train agents to specify the accessing and leaving time on a rail road track in their bids. Comparing with L_B, L_R languages do not require agents to compute the values on the combinations of resource time units. However, this does not mean that the computation used to determine the values of time units' combinations has been eliminated by using L_R. Instead, it is migrated to the auctioneer's winner determination (will be explained in detail in Section 3). How this migration approach will affect the computational complexity of auctioneer's winner determination

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

is an important question that will be answered by the comparative study.

This paper investigates the complexity issues of using L_B and L_R in auction-based scheduling. While we focus only on bidding languages, we assume agents' strategic behavior and, therefore, auctions are a suitable mechanism for the scheduling problems. Kalagnanam and Parkes [2] reviewed four areas of computational constraints, which restrict the space of feasible combinatorial auction mechanisms, including, strategic complexity, communication complexity, valuation complexity and winner determination complexity. Since strategic complexity is not affected by languages [6], we study the other three in the context of auction-based scheduling. For valuation and communication complexities, we compare the two types of languages analytically; for the winner determination complexity, we compare them experimentally. Our main results are (1) in auction-based scheduling, L_R languages have reduced complexities in agents' valuation and system's communication; (2) although L_R languages migrate agents' valuation complexity to the auctioneer's winner determination, this type of language enables scheduling specific modeling techniques to be incorporated into the winner determination problem formulation, which results in a more efficient model than the traditional one formulated by L_B in terms of solving speed and scalability.

The rest of the paper is organized as follows: Section II formulates the scheduling auction model and specifies the L_R language used in this paper; Sections III and IV, analyze the valuation and communication complexities of L_B and L_R and in Section V, we conduct a computational study to compare the performance of the winner determination problem models formulated using L_R and L_B. We conclude the paper in Section VI.

II. The Auction for Scheduling Problems

Wellman et al. [13] modeled a factory scheduling problem as a CA problem. In the model, a factory conducts an auction for time slots on a single resource. Time slots are treated as distinct items that can be allocated for the production of customer orders. Each customer (modeled as an agent) has one single-operation job to be completed. An agent's job is defined by its duration, its release time, its deadline, and the price the agent places on the job. To complete its job, the agent must acquire a number of slots no less than the length, within its feasible time window. In this paper, we expand the model from Wellman et al. to accommodate multiple resources and multi-operational jobs. We use this general model as the base for comparing various types of complexities related to L_B and L_R. We refer to this general model as the scheduling auction.

The scheduling auction consists of a set of agents, denoted by N. Each agent $g \in N$ has a set of jobs J^g. Each job $J_j \in J^g$ requires the processing of a sequence of operations $o_{j,k}$ $\left(k = 1,...,n_j\right)$. 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. J^g is constrained by a release time r^g by which the jobs are available for processing, and a deadline d^g by which all jobs must be completed. There are precedence constraints among the operations of a job And o precedence constraints among jobs. An allocation of all jobs in J^g, on the resources over time, form a schedule for agent g, denoted by S^g. Let $C_{\max}\left(S^g\right)$ denote the completion time of the last job in S^g ($C_{\max}\left(S^g\right)$,called the makespan of S^g in machine scheduling). For each agent $g \in N$, its value for a schedule S^g is $v^g\left(S^g\right)$. An agent prefers a schedule with a shorter makespan, that is, for two schedules S^g and $\bar{S}^g$, if $C_{\max}\left(S^g\right) \le C_{\max}\left(\bar{S}^g\right)$, $v^g\left(S^g\right) \ge v^g\left(\bar{S}^g\right)$. In the context of using L_B, with a little abuse of notation, S^g can also be seen as a set of time units allocated to agent g. The overall objective of the auction is to maximize the sum of all the agents' values.

Fig. 1 shows an example of the scheduling auction problem with three resources (R_1, R_2, R_3) and four jobs. Job1 has 3 operations ($O_{1,1}, O_{1,2}, O_{1,3}$); job 2 has 2 operations ($O_{2,1}, O_{2,2}$); job 3 has 3 operations ($O_{3,1}, O_{3,2}, O_{3,3}$); job 4 has 2 operations ($O_{4,1}, O_{4,2}$). The arcs (with solid lines) represent the precedence constraints between operations; and, arcs (with dotted line) link operations to their designated processing resources.

The scheduling auction can be seen as a model of many real world scheduling problems. In manufacturing, for example, customers have jobs with different release times and deadline requirements to be processed in the factory. The factory tries to allocate the limited resources to the customers who value them the most. Similar scenarios can also be found in other domains such as transportation and grid computing. While there are many scheduling models and algorithms in classical scheduling theory, the scheduling auction modeled here assumes that agents are self-interested and they behave strategically.

III. The Requirement-Based Language

In the scheduling auction models, agents derive values based on the levels that their objectives have been satisfied. In

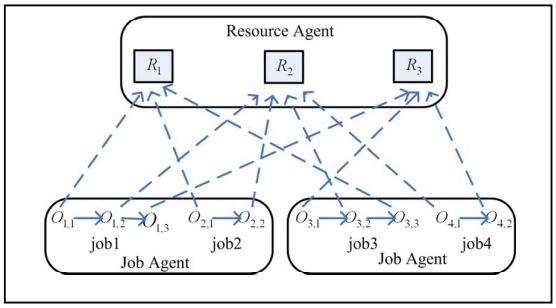

Figure 1. Example of the Scheduling Auction Model

this section, we present a requirement-based bidding language L_R, in which the atomic propositions attach prices to requirements of processing jobs rather than bundles of items (as in L_B). The purpose here is to design a language that captures the intrinsic structure of the scheduling problem, such that agents' valuations can be expressed naturally and concisely using the language.

A. General Structure of Atomic Propositions in L_R

As depicted in Fig. 2, an Atomic Proposition of L_R consists of the Requirement of completing a set of Jobs according to a Performance requirement and the Price that the agent is willing to pay given the Performance requirement is satisfied. The Performance is defined by a Measure and its Level. Formally, an Atomic Proposition can be represented by a 4-tuple $\langle Jobs, Measure, Level, \Pr ice \rangle$.

Jobs represent the set of jobs from an agent that needs to be processed. For each job, the associated operations, constraints over the operations, and eligibility constraints over resources need to be specified. The actual content language used to describe Jobs can be domain specific. We do not discuss it in this paper.

Measure is a criterion based on which the quality of a schedule for Jobs is evaluated. Some typical criteria include total-production-time (makespan), mean flow-time, maximum tardiness, and weighted tardiness.

Level is the value achieved by a schedule in terms of the objective function specified in the Measure. For example, if the Measure is makespan and the Level is 20, the semantic interpretation of the Performance is to require the jobs to be scheduled with a makespan no larger than 20.

Price is the amount of money that the agent is willing to pay given that the Jobs are scheduled at a specific level based on the measure. For example, the Atomic Proposition $\langle Jobs, Makespan, 20, \$100 \rangle$ means if the Jobs are scheduled to be completed with a makespan of 20, the agent is willing to pay \$100.

As the Performance (Measure and Level) can be defined by the job agents, the Atomic Proposition structure is general enough to capture job agents' requirements regarding the processing of their jobs. However, for a specific application domain, it is normal that only a small portion of the measures is of importance to agents. For example, in eMarket environments, the common performance measure that a client will require is the delivery date of his/her order, which is the makespan in terms of scheduling. We will specify a type of L_R, which uses makespan as the measure in the following subsection.

B. The Completion Time-Based L_R

The atomic proposition of the time-based L_R consists of a requirement of scheduling a set of jobs, the completion time before which the jobs need to be completed, and the price that the agent is willing to pay given the completion time is

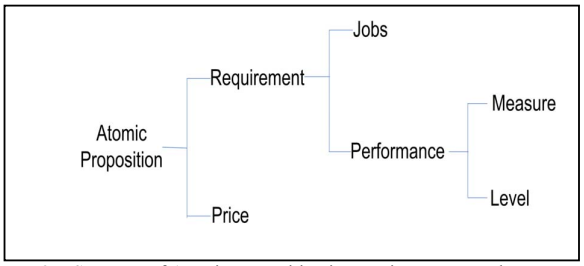

Figure 2. Structure of Atomic Proposition in Requirement-Based Language

satisfied. We refer this atomic proposition as CBid (Completion time-based bid).

CBid is a 4-tuple $\langle J^g, C^g_{\max}, lft, p \rangle$ where J^g is a set that contains the descriptions of the jobs and constraints; $C^g_{\max}$ defines that the measure being used is makespan; *lft* is the latest finishing time and p is the price that the agent is willing to pay for $C^g_{\max} \le lft$. CBids can also be connected by logical connectives. For example, if an agent is willing to pay \$100 for the guarantee of completing its jobs before 4:00PM or \$60 for completing its job before 6:00PM, it can express this valuation by submitting an XOR-CBid: $\langle J^g, C^g_{\max}, 4:00PM, \$100 \rangle XOR \langle J^g, C^g_{\max}, 6:00PM, \$60 \rangle$. Note that CBid requires an agent to reveal its true processing requirements (job descriptions). However, it does not require the agent to reveal its true valuation information (*lft* and p). This is quite reasonable in many real world situations. For example, a customer may benefit from lying about the true value and due date of manufacturing a part, however, there is no need to lie about the processing requirements of the part because the part will eventually be processed based on the requirements.

IV. Valuation and Communication Complexities

A. Valuation Complexity

Valuation is usually costly when using L_B for auction-based scheduling because agents need to solve a hard optimization problem in order to determine the value of a bundle. In this section, we analyze the computational complexity of agent valuation.

We first define the value of a bundle in the L_B setting. Let S^g be a schedule that contains jobs of agent g. For a bundle B of time units, if $S^g \subseteq B$, we say S^g is covered by B. In many cases, a bundle can cover several feasible schedules for an agent. We define the value of a bundle to an agent as the value of the best schedule (with the shortest makespan) the bundle covers.

Definition 1: Let Γ be the set of schedules of agent g covered by B. The valuation of agent g on bundle B is set to be the value of the best schedule $S^g_* \in \Gamma$, such that for any $S^g \in \Gamma$, $v^g(S^g_*) \ge v^g(S^g)$.

If we assume that, for any $S^g \in \Gamma$, $v^g\left(S^g\right)$ has been given to agent g, according to Definition 1, the valuation problem for a bundle of time units in the L_B model can be described as: given a set of jobs of an agent to be allocated to a bundle of time units of various resources, what is the shortest possible makespan that a feasible schedule can have? Answering this question is equivalent to solving a job shop scheduling problem with availability constraints (JSPAC), which is NP-hard [5]. This proves:

Proposition 1: In the scheduling auction model using L_B, an agent's valuation problem for a bundle of time units is NP-hard.

While agents' valuation problems in L_B models are NP-hard, they become trivial in L_R models. In L_R models, agents do not deal with bundles of time units. In other words, they do not spend their computational time on finding appropriate time units combinations on resources to schedule their job requirements. Instead, they just send their requirements (jobs and required completion times) and associated values as L_R bids to the auctioneer. Since we have assumed that, for any $S^g \in \Gamma$, $v^g\left(S^g\right)$ is a given, the task of finding the value for a schedule is trivial for agents using L_R bids. Therefore, from the agents' point of view, the L_R model has the advantage of avoiding the NP-hard problem of solving the JSPACs. However, this does not mean agents' valuation complexity has been eliminated in L_R models. In fact, this computational burden is shifted to the auctioneer's winner determination because, in L_R WDP, the auctioneer has to determine the winning bids and, at the same time, schedule jobs on resources. This idea is illustrated in Fig. 3.

B. Communication Complexity

The communication complexity of an auction considers the size of messages that must be sent between the agents and the auctioneer. A simple measure of the size of messages could be the number of bids needed to implement the outcome of an auction. In general CAs, the number of bids for an agent is $2^m - 1$, where m is the number of items to be sold. However, in the scheduling auction model, the number of feasible bids can be restricted by the scheduling constraints. Formally, consider an agent g has a job J_j with n_j operations to be processed in a time window with release time r^g and deadline d^g. Let $W = d^g - r^g$ be the size of the time window, which is the number of time units between the release time and the deadline. For each operation o_k (since we only consider one job for the time being, we drop the job subscript j to simplify the notations), a processing time p_k is given. To schedule the set of operations in W, three constraints have to be satisfied:

$$S_{k-1} + p_{k-1} \le S_k \qquad for\ 1 < k \le n_j \tag{1}$$

$$S_1 \ge r^g \tag{2}$$

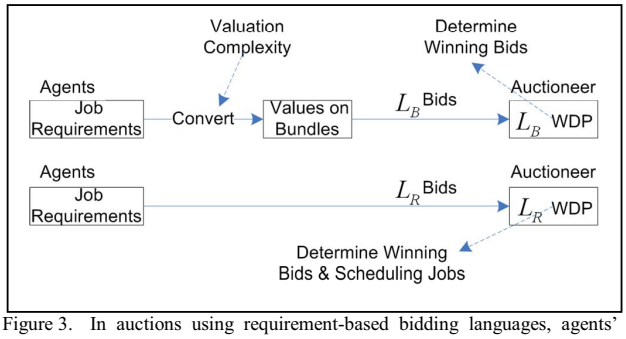

Figure 3. In auctions using requirement-based bidding languages, agents' valuation complexity is migrated to the auctioneer's winner determination. In addition to determining winning bids, the auctioneer needs to schedule jobs at the same time.

$$S_{n_j} \le d^g - p_{n_j} \tag{3}$$

Where S_k is the starting time of o_k. The starting time of an operation could vary in different feasible schedules. By counting the number of all combinations of feasible starting times of operations, we can calculate the number of feasible schedules in a time window W by the following formula:

$$\sum_{S_1=0}^{W-p_1-p_2\cdots p_{n_j}} \sum_{S_2=p_1+S_1}^{W-p_2-p_3\cdots p_{n_j}} \cdots \sum_{S_{n_j}=p_{n_{j-1}}+S_{n_{j-1}}}^{W-p_{n_j}} 1 \tag{4}$$

By relaxing constraint (1) and set $p_1 = p_2 = \ldots p_{n_j} = 1$, an upper bound of (4) can be obtained as $\left(W - n_j + 1\right)^{n_j}$. Since an agent cab, at most, attach one value to a feasible schedule, the following proposition holds:

Proposition 2 For an agent with one job J_j, the number of bids in W is bounded by $\left(W - n_j + 1\right)^{n_j}$.

Although Proposition 2 shows that the number of L_B bids that an agent needs to submit does not grow exponentially in W, it still increases drastically when W increases. In real world applications, to maintain time accuracy, the time window size W cannot be too small, which often results in a large number of bids. If an agent has multiple jobs, the number of multi-job bids will grow even more quickly because of the combinations of single-job bids.

In the L_R model, a CBid represents the value that an agent has over the completion time of its jobs. Without loss of generality, we assume the completion times are of integer values. If an agent has a set of jobs J^g with a release time r^g and a deadline d^g, the number of bids that the agent needs to submit is bounded by $W = d^g - r^g$. For any problem instance with operations' processing times bigger than 2, in terms of the communication complexity, the L_B upper bound $\left(W - n_j + 1\right)^{n_j}$ is greater than the L_R upper bound W.

V. WINNER DETERMINATION COMPLEXITY

In this section we conduct a computational study to experimentally evaluate the complexities of winner determination problems formulated using L_B and L_R. We use a commercial optimization package CPLEX 10.1 as the winner determination algorithm. For the L_B winner determination problem formulation we use the one presented in [12]. For the L_R formulation we use that presented in [14].

A. Experimental Setup

Common combinatorial auction benchmarks distributions, such as those presented in [10], are designed for general CAs. They are not for scheduling problems. In [4] Leyton-Brown et al. presented a set of scheduling benchmark distributions generated, based on the factory scheduling economy from Wellman et al. [13]. These scheduling distributions are single-resource, single-operation problems, which are special cases of our scheduling auction model. Sandholm et al. have reported in [11] that CPLEX 8.0 is slightly faster than CABOB on the set of single-resource, single-operations scheduling distributions. We design our scheduling test problems based on a suite of job shop CSP benchmark problems developed in [9]. While the job shop CSP benchmark problems are constraint satisfaction problems, we have added a price parameter P to construct the scheduling auction problem set. The price of job j is randomly drawn from a uniform distribution on $U\left(Pdu_j, du + Pdu_j\right)$, where du is the average duration of all jobs, and du_j is the duration of job j. By considering different sizes of problems (determined by the number of jobs in a problem and number of operations in a job), a problem set was randomly generated. In these problems the number of operations ranges from 2 to 6; the number of jobs ranges from 2 to 7; and, 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. For a problem instance, we first convert it to L_B WDP and L_R WDP. Then we solve the two WDPs using CPLEX 10.1, respectively. Each point in each plot is the mean run time for 10 problem instances with the same numbers of jobs and same numbers of operations in each instance.

B. Experimental Results

Since we intended to compare the performance of L_B WDP and L_R WDP in terms of solving speed and scalability, we present the experimental results from two perspectives: (1) given a fixed number of operations in the problems, how run times change when the number of jobs increases (Fig. 4); (2) given a fixed number of jobs, how run times change when the number of operations increases (Fig. 5).

As shown in Fig. 4, for the first two groups of problems (operation number=2 and operation number=3), the running times of L_B WDP and L_R WDP are, initially, close. When the number of jobs increases, the differences increase quickly. For the rest of the two groups of problems (operation number=4 and operation number=5), L_R WDP is more than 10 times faster than L_B WDP even at the size of 2 jobs. It is observed that, L_B WDP does not scale well. L_B WDP can be 100 to 1000 times slower when the number of jobs reaches 7.

Fig. 5 presents the results from a different angle. Again, we see that L_B WDP does not scale well when the number of operations increases. On the contrary, the running times of L_R WDP are virtually unaffected when the number of operations increases from 2 to 5 in all four groups of problems. The scalability characteristics of L_R WDP are further illustrated in Fig. 6 and Fig. 7. It is shown in Fig. 6 that the scalability L_R WDP remains good when the number of jobs is smaller than 5. When the number of jobs goes beyond 5, the scalability of L_R WDP decreases with a higher rate. Fig. 7 shows that L_R WDP's scale very well along the number of operations at all job number levels.

VI. CONCLUSION

As bidding languages have an impact on various aspects of the computational complexities of an auction, the understanding of the complexity implications of various languages, in the context of the auction-based scheduling, is of practical interest in the design of auctions for scheduling problems. We have compared the general bidding languages and the requirement-based bidding languages in terms of their implications to the valuation, communication, and winner determination complexities in auction-based scheduling. We show that the requirement-based language provides concise, natural representations of agents' valuations and reduces agents' valuation complexity and system's communication complexity. An interesting finding is, although the auctioneer has to solve winner determination and scheduling problems concurrently, when allowing the requirement-based bidding language, a L_R WDP formulated by incorporating scheduling specific modeling techniques can be far more efficient than the standard L_B WDP in terms of solving speed and scalability. This computational efficiency and reduced valuation and communication complexity makes the requirement-based bidding language a suitable choice for auction-based scheduling.

Another key benefit that the requirement-based bidding languages can offer is that it preserves natural scheduling constraints in the WDP formulation, which enables the design of effective winner determination algorithms by leveraging the wealth of scheduling research in the past several decades. The general optimization package we have used in this paper has produced good results. It is reasonable to predict that the design of scheduling specific winner determined algorithms is a promising direction worth exploring.

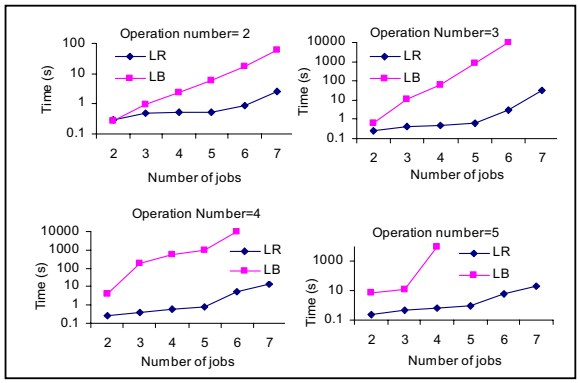

Figure 4. Run times of L_B WDP and L_R WDP over jobs

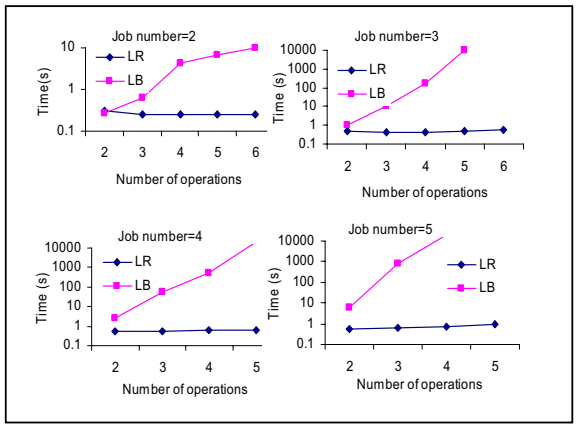

Figure 5. Run times of L_B WDP and L_R WDP over operations

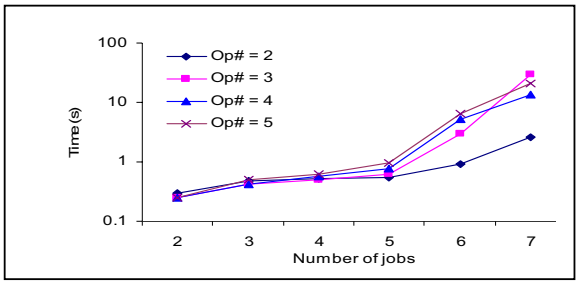

Figure 6. L_R WDP scalability over jobs

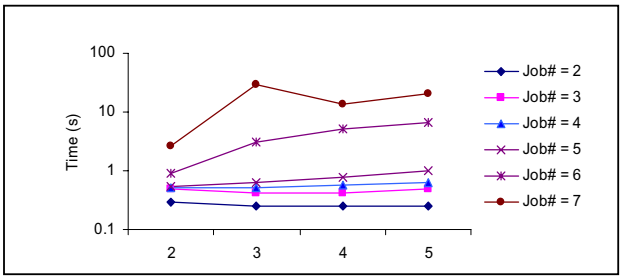

Figure 7. L_R scalability over operations

REFERENCES

[1] Buyya, R., "Economic Paradigm for Distributed Resource Management and Scheduling for Service Oriented Grid Computing," Ph. D thesis, Monash University, April 12, 2002

[2] Kalagnanam, J. and D. C. Parkes, 2004, Auctions, bidding and exchange design, David Simchi-Levi, S. David Wu, and Z. Max Shen (Eds.) Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era, Kluwer Academic Publishers.

[3] Kutanoglu, E., Wu, S. D. On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. IIE Trans., 31, 9 (Sept. 1999), 813-826.

[4] Leyton-Brown, K., Pearson, M., Shoham, Y. Towards a universal test suite for combinatorial auction algorithms. In Proc. ACM Conf. Electronic Commerce (ACM-EC), Minneapolis, MN. ACM, New York, 2000, 66–76.

[5] Mauguière, P., Billaut, J., and Bouquard, J. 2005. New Single Machine and Job-Shop Scheduling Problems with Availability Constraints. J. of Scheduling 8, 3, pp.211-231.

[6] Nisan, N., 2006. Bidding languages for combinatorial auctions. Combinatorial Auctions, Cramton, Shoham, and Steinberg, eds., MIT Press.

[7] Parkes, D. C. and Ungar, L. An Auction-Based Method for Decentralized Train Scheduling. In Proceedings of 5th International Conference on Autonomous Agents (AGENTS-01), Montreal, Quebec, Canada, 2001, 43-50.

[8] Rassenti, S. J., Smith V. L., and Bulfin, R. L. A Combinatorial Auction Mechanism for Airport Time Slot Allocation. Bell Journal of Economics, vol. 13, no. 2, pp. 402-417, 1982.

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

[10] Sandholm, T. Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence, 135, 2002, 1-54.

[11] Sandholm, T., Suri, S., Gilpin, A. and Levine, D. CABOB: A Fast Optimal Algorithm for Winner Determination in Combinatorial Auctions. Management Science, 51, 3, 2005, 374-390.

[12] de Vries, S., Vohra, R.V. Combinatorial Auctions: A Survey. INFORMS journal on Computing, 15, 3, 2003, 284-309.

[13] Wellman, M. P., Walsh, E., Wurman, P. R., and MacKie-Mason, J. K. Auction Protocols for Decentralized Scheduling. *Games and Economic Behavior*, 35(1-2), 2001, 271-303.

[14] Wang, C., Ghenniwa, H., Shen, W., "Constraint-Based Winner Determination for Auction-Based Scheduling," IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 39, No. 3, pp. 609-618, 2009.