2011 Society for Design and Process Science
Printed in the United States of America

A*d*SCHE: DESIGN OF AN AUCTION-BASED FRAMEWORK FOR DECENTRALIZED SCHEDULIING

Chun Wang
Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada

Decentralized scheduling is one of the newly emerged avenues in scheduling research. It is concerned with allocating resources to alternative possible uses over time, where competing uses are represented by autonomous agents. Compared with classical scheduling models, decentralized scheduling is characterized with the distribution of scheduling knowledge and control, which introduces new levels of complexities, namely the coordination complexity due to the interaction problems among agents and the mechanism design complexity due to the self-interested nature of agents. These complexities intertwine and need to be addressed concurrently. This paper presents an auction-based framework which tackles coordination and mechanism design complexities through integrating an iterative bidding protocol, a requirement-based bidding language, and a constraint-based winner determination approach. Without imposing a time window discretization on resources the requirement-based bidding language allows bidders to bid for the processing of a set of jobs with constraints. Prices can be attached to quality attributes of schedules. The winner determination algorithm uses a depth first branch and bound search. A constraint directed scheduling procedure is used at each node to verify the feasibility of the allocation. The bidding procedure is implemented by an ascending auction protocol. Experimental results show that the proposed auction framework exhibits improved computational properties compared with the general combinatorial auctions. A case study of applying the framework to decentralized media content scheduling in narrowcasting is also presented.

Keywords: *decentralized scheduling, combinatorial auctions, iterative bidding, bidding languages, winner determination*

1. Introduction

Decentralized scheduling is one of the newly emerged avenues in scheduling research. Compared with classical scheduling problem models, decentralized scheduling problems differentiate themselves in the distribution of both scheduling knowledge and control. In a decentralized scheduling problem, independent entities, e.g., individuals, enterprises, and computational devices, have jobs that need to be completed during specific time windows. They compete with each other for the resources to schedule their own jobs according to their respective objectives. Scheduling related information such as jobs, resources, constraints and objectives are distributed among individual entities. An entity is motivated by its own objectives and not controlled by other entities or a system wide authority. Many scheduling problems are inherently decentralized, such as those in supply chain management, production management, network-based information-processing environments, transportation and distribution settings, and many other types of service industries. In recent years, automated decentralized scheduling systems attracted more attention as a result of the significant growth of eMarkets which have presented tremendous potential in changing the way companies buy and sell goods, integrate their supply chains, and collaborate with their business partners. In a considerable portion of eMarket applications, the goods to be sold are processing times of resources, e.g. landing timeslots of airport runways (Rassenti et al., 1982), machine processing times of a factory (Wellman et al., 2001), computation and network accessing times of internet resources (Buyya, 2002), and the right to use railroad tracks (Brewer, 1996). For this type of eMarket applications, decentralized scheduling plays an essential role in the design and implementation of software trading systems which facilitate the commercial activities among market participants.

The computational complexity involved in solving scheduling problems has been the central theme of the classical scheduling theory. Due to the NP-hard nature of scheduling models, the computation demanded for solving many problems at practical sizes is prohibitive. In addition, decentralized scheduling presents two other challenges attributable to the distribution of scheduling knowledge and the distribution of control. In decentralized scheduling, scheduling related information is located in entities which are distributed across the system. These entities are independent and self-interested[1]. In other words, no entity has control over other entities. Each entity has the authority to decide how to deploy its resources in service of its objectives. To recognize the *independent* and *autonomous* nature of the entities, we treat them as *agents*[2] in our decentralized scheduling models.

By modelling the processing times available on resources as goods to be sold, decentralized scheduling problems can be mapped to a type of *combinatorial allocation problems* to which combinatorial auctions have been proposed as solution approaches[3]. However, the mapping is not straightforward since the processing times in decentralized scheduling problems and the goods in combinatorial allocation problems are essentially different in terms of divisibility. The processing times are divisible; however, the goods are distinct items, thus, indivisible. One possible mapping approach is to impose a discretization on the time windows of resources to be scheduled and treat the time slots generated by the discretization as distinct items (Kutanoglu and Wu, 2006; Wellman et al., 2001). However, this discretization approach can generate a large number of items 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 thousand time slots if the time accuracy we need is in minutes (which is a practical requirement in many application domains). Generally speaking, in combinatorial auctions the number of possible bids is exponential in the number of the items to be sold. A large number of items can inflict heavy computational burdens on software agents in terms of bids evaluation, on the auctioneer in terms of winner determination, and on the system in terms of communication.

The objective of this research is to design solution approaches to scheduling problems in decentralized environments. In particular, we propose an integrated framework which consists of a bidding language, an iterative bidding framework, and a constraint-based winner auction-based model. The framework is designed to tackle the high computational complexities in terms of bids valuation, communication, and winner determination. Our approach is to explore the interface between computational mechanism design and classical scheduling theory in the context of auction-based decentralized scheduling and enrich both fields by (1) extending classical centralized scheduling models to decentralized models; (2) embedding scheduling specific problem solving structures and heuristics in combinatorial auctions in order that computational challenges in auction-based

[1] Self-interest is an assumption of classical economic theory meaning that individuals are motivated in their actions by self interest. In *The Wealth of Nations* (Smith, 1937), Adam Smith makes the claim that, within the system of capitalism, an individual acting for his own good tends also to promote the good of his community. He attributed this principle to a social mechanism that he called "the invisible hand".

[2] In this thesis, *agents* are referred in the context of distributed system engineering. Specifically, *agents* are defined as intelligent, rational, and autonomous software artifacts, which are able to cooperate with each other, reason, and act appropriately with respect to their environment. They may represent individuals, enterprises, or computational resources in decentralized scheduling environments.

[3] A combinatorial allocation problem is a resource allocation problem, in which a non-empty set of items are to be allocated across a set of agents. Agents are assumed to value bundles of items. This problem has been studied in the combinatorial auction literature from various perspectives (de Vries and Vohra, 2003).

decentralized scheduling can be addressed effectively. The rest of the paper is structured as follows. Section 2 reviews related work in the literature. Section 3 defines a decentralized scheduling problem model: the scheduling auction. Section 4 describes the auction-based decentralized scheduling (A*d*SCHE) framework. Computational performance of A*d*SCHE is also evaluated. In Section 5, we present a case study of applying A*d*SCHE to decentralized media content scheduling in the narrowcasting environment. Section 6 concludes the paper and discusses future research directions.

2. Literature Review

Scheduling can be seen as a class of resource allocation problems in which resources are allocated to tasks over the time dimension. Many economic-based resource allocation models have been studied in the literature. While giving a comprehensive review of these models is beyond the scope of this paper, we summarize four models which are of importance to decentralized scheduling. In economics, the concept of a set of interrelated goods in balance is called general equilibrium. General equilibrium theory provides a distributed method for efficiently allocating goods and resources among agents based on market prices. In applying this general equilibrium based mechanism to decentralized scheduling, the goods in the markets need to be specified by imposing a discretization on the continuous timeline to be scheduled on the resources. These goods are discrete ones, which violate the infinite divisibility of goods condition of general equilibrium theory. Markets with discrete goods and complementary preferences of agents can lack equilibria (Walsh and Wellman, 1999). The performance of general equilibrium based market mechanisms on decentralized scheduling is not guaranteed.

Sequential and simultaneous auctions price bundles as the sum price of the individual items. However, they do not allow bidders to bid on bundles of items. Sequential auctions suppose that the set of items are auctioned in sequence. Bidders bid for items in a specific, known order, and can choose how much (and whether) to bid for an item depending on past successes, failures, prices, and so on. Sequential auctions are particularly useful in situations where setting up combinatorial or simultaneous auctions are infeasible. Simultaneous auctions sell multiple items in separate markets simultaneously. Bidders have to interact with simultaneous but distinct markets in order to obtain a combination of items sufficient to accomplish their task. Real-world markets quite typically operate separately and concurrently despite significant interactions in preferences. A typical example is the series of FCC spectrum auctions (McAfee and McMillan, 1996). In Parkes and Ungar (2000), simultaneous auctions are designed for decentralized train scheduling problems. A review of the uses of economic theory in simultaneous auction design can be found in Milgrom (2000). Sequential and simultaneous auctions tackle the complementarities over resources in the same spirit of general equilibrium theory. These auctions fail when there are no prices that support an efficient solution (the existence problem) and also when agents bid cautiously to avoid purchasing an incomplete bundle (the exposure problem). However, given that these auctions are more practical in terms of computation, they are two important models worthy of further study.

Combinatorial auctions (CAs) allow bidders to place bids on bundles of items. It addresses complementary preference issue explicitly. However, the computation required for solving hard valuation problems and winner determination problems can be prohibitive. In general, CAs are likely to be practical for smaller size problems. The computational complexities of CAs have been studied by various researchers (de Vries and Vohra, 2003). Some sophisticated algorithms have produced promising results (Sandholm, 2002; Sandholm et al., 2005). In terms of applying CAs to scheduling, if general bundle languages, such as L_G and L_B (Boutilier and Hoos, 2001) are used, the timeline of the firm's production resources needs to be discretized into small time units. This timeline discretization usually results in a large amount of items to be sold in the auction, which leads to bigger size problems. Applying CAs to a big size scheduling problem can inflict heavy computation burdens on both the customer and the firm side. Another limitation with general CAs is the so called "lying auctioneer" problem (Sandholm, 1999), which partially explains why Vickery auction is not widely used in practice, even though it has been proposed since 1960's.

Table 1 Summary of the key characteristics of the economic models

Economic models	**Key Characteristics**	**Exemplary References**
General equilibrium mechanisms	Solve resource allocation or scheduling problems by constructing computational markets based on general equilibrium theory.	Ygge and Akkemans (1996) Walsh and Wellman (1999) Wellman (1995)
Sequential and simultaneous auctions	Do not allow bids on bundles of items. Sequential auctions sell multiple items in sequence. Simultaneous auctions sell multiple items in separate markets simultaneously.	Boutilier *et al.* (1999) Engelbrecht and Weber (1983) McAfee and McMillan (1996) Reeves *et al.* (2005) Parkes and Ungar (2001)
Combinatorial auctions (CAs)	Allow bidders to submit valuations on bundles of items.	Kutanoglu and Wu (2006) de Vries and Vohra (2003) Sandholm et al. (2005) Sandholm (2002)
Iterative bundle auctions	Allow bidders to submit multiple bids during an auction and provides information feedback to support adaptive and focused eliciation.	Parkes and Ungar (2000) Bikhchandani and Ostroy (2006) Kutanoglu and Wu (1999) Wellman et al. (2001)

Iterative bundle auctions are iterative implementations of CAs. This class of auction has practical significance because it addresses the computational and informational complexities of CAs by allowing bidders to reveal their preference information only as necessary as the auction proceeds, and bidders are not required to submit (and compute) complete and exact information about their private valuations. With careful design of the structure and components, iterative bundle auctions have the potential of significantly reducing computational costs in CAs. In addition, iterative auctions specially designed for scheduling problems have also been proposed in the literature. In Kutanoglu and Wu (1999), iterative auctions are applied to the job shop scheduling problem. The focus in Kutanoglu and Wu (1999) is to investigate the links between combinatorial auctions and Lagrangean relaxation, and to design auctions based on the Lagrangean based decomposition. In Wellman *et al.* (2001), the properties of several iterative auction protocols are investigated in the context of decentralized scheduling. In Mackie-Mason *et al.* (2004) and Reeves *et al.* (2005) price prediction and bidding strategies for simultaneous auctions are studied in the setting of market-based scheduling. The proposed framework in this paper is an iterative bundle auction specially designed for scheduling problems. In many cases, iterative auctions present better computational and privacy properties than those of CAs. In addition, iterative auctions have the potential of accommodating dynamic events, which is required in many real-world scheduling applications. Compared with existing iterative bundle auctions, the novelty of our design is that it uses a requirement-based bidding language to represent scheduling domain specific due date, pricing, and job requirements. Unlike general iterative auctions which use bundle languages, the requirement-based language avoids imposing timeline discretization, which causes a large amount of items to be sold in the auction; the adoption of this language also enables the design of more efficient winner determination algorithms which take advantage of the domain specific information to improve the search efficiency. Our previous study (Wang *et al.*, 2009a) has shown that, in auction-based decentralized scheduling, requirement-based languages result in more efficient winner determination models than bundle languages do. The key characteristics of the four economic models are summarized in Table 1.

In agent-based manufacturing control literature, the contract net (Smith, 1980) and its later variants have been applied to scheduling as a class of distributed decision making protocols. Unlike auctions, which usually require a mediator, contract nets are purely distributed models, in which any agent can act as a manager and subcontract tasks to other agents. Most of the agent-based control systems were designed for the coordination of production processes within the boundary of an enterprise. References and reviews of this line of research can be found in Shen *et al.* (2006).

3. The Scheduling Auction

In this section, we present a *scheduling auction* model which extends the factory scheduling model described in Wellman *et al.* (2001). The scheduling auction is a typical decentralized job shop scheduling setting. We use this model as the base environment for evaluating the proposed A*d*SCHE framework. 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 $o_{j,k}$ $(k = 1 \dots n_j)$. 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 must be completed. For a feasible schedule with completion time c_j, $r_j < c_j \leq d_j$, 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.

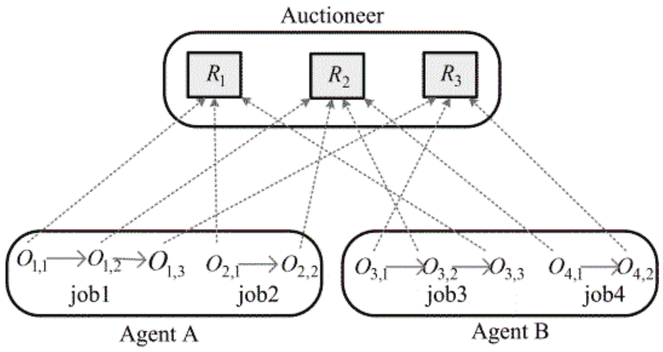

Fig. 1 Example of the Scheduling Auction Model

Figure 1 shows an example of the scheduling auction with three resources $(\mathrm{R}_1, \mathrm{R}_2, \mathrm{R}_3)$ 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 (Shen, 2002).

4. The A*d*SCHE

In this section, we propose A*d*SCHE as a solution framework for auction-based decentralized scheduling. A*d*SCHE is a type of ascending combinatorial auctions. Comparing to general iterative auctions in literature, it uses the requirement-based bidding languages and the constraint-based winner determination, which are specially designed for decentralized scheduling problems. A*d*SCHE contains three major components, requirement-based bidding languages, constraint-based winner determination algorithms, and an iterative bidding procedure. The requirement-based bidding languages allow an agent's bid to be expressed by a requirement of processing a set of jobs with constraints, which avoid the use of time window discretization approach. The winner determination algorithm is designed for solving the winner determination problems formulated using the bids expressed in the requirement-based languages. The iterative bidding procedure reduces agents' information revelation and adds the potential of accommodating dynamic changes during the auction process. In the following, we describe the detailed design of the three components.

4.1. 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 (Boutilier and Hoos, 2001) 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 (or bid) is more expressive in terms of representing scheduling problems. It consists of a description of the job to be completed, a performance requirement and the price that the agent is willing to pay given the requirement is satisfied. The performance requirement is defined by a *Measure* and its *Level*. Formally, an *atomic proposition* can be represented by a 4-tuple $\langle Job, Measure, Level, Price\rangle$.

Job specifies a sequence of operations and their required processing time on resources. In addition, the release time, deadline of the job and other processing constraints are also specified. For many scheduling models, existing general scheduling problem description languages, such as the one proposed in Smith and Becker (1997) 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 Figure 1 can be presented as $\left((R_3,23),(R_2,15),(R_1,18),r_3=20,d_3=80\right)$, which means it needs to be processed by R_3, R_2 and R_1 in sequence within the time window $(20,80]$, and the processing times are 23, 15 and 18.

Measure is a criterion based on which the quality of a schedule is evaluated. A schedule can be evaluated by many types of scheduling criteria (Pinedo, 2002). In our scheduling auction, we use the completion time of a job as the measure.

Level is the range achieved by a schedule in terms of the criterion specified in the *Measure*. For example, if the *Measure* is completion time and *Level* is $(20,40]$, the semantic interpretation of the performance requirement is that the job is 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 certain level of performance measure. For example, the atomic proposition $\langle job, completion, (20,40], \$100\rangle$ is interpreted as the agent is willing to pay \$100 if the completion time of the job is within $(20,40]$. In direct revelation mechanisms, such as VCG, the price is agent's value.

By connecting $\mathrm{L_R}$ atomic bids using XOR, we can express an agent's values on different levels of the performance measure. For example, we can use the following compound bid $\langle job, completion, (80, 100], \$2k\rangle XOR \langle job, completion, (100, 120], \$1.9k\rangle$ 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. Finally, since the agent has additive valuations over jobs, it can express them by connecting the XOR bids, each for a job, using the logical connector OR. In L_R, OR cannot be used to connect atomic bids from one job because a job can only be scheduled once. 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. A more detailed analysis of the properties of L_R can be found in Wang *et al.* (2009a).

4.2. Constraint-Based Winner Determination

We design a branch and bound algorithm for computing the awarded bids from agents. A branch bound algorithm for winner determination problems can branch on items (Sandholm, 2002) or bids (Sandholm, 2005). Since distinct items are not modeled in A*d*SCHE, we propose an algorithmic framework which branches on bids. Branch-on-bids starts with an empty temporary solution and gradually adds bids to it along the search path. To detect the unfeasible branches at early stages, feasibility of the temporary solution needs to be checked when a new bid is added in at each node. For the combinatorial auctions with distinct items, feasibility checking is easy since as long as any two winning bids do not share an item, the solution is feasible. However, since $L_R$is used in A*d*SCHE, validating the feasibility of a solution is equal to answering the question: given a collection of jobs belong to different agents does a schedule exist that allocates the jobs on the resources, such that all constraints are satisfied? This decision problem is actually a job shop Constraint Satisfaction Problem which is known to be NP-complete (Garey and Johnson, 1979). For this problem, we propose a branch-on-bids algorithm embedded with constraint-directed feasibility validation for winner determination in auction-based decentralized scheduling using requirement-based bidding languages.

The actual algorithm designed is a depth-first tree search. The search starts with an empty temporal schedule, called $TEMP$. Along the path $TEMP$ is expended by adding more bids form AV , which is a set that constraints available (not winning) bids. The best $TEMP$ found so far is $TEMP^*$. sum is the revenue of $TEMP$, which is the summation of prices of bids in $TEMP$, and sum^* be the revenue of $TEMP^*$. h is an upper bound on how much the bids in AV can contribute. The search is invoked by calling BRANCH-BOUND-SCHEDULING (*bids*).

Algorithm 1

```
function BRANCH-BOUND-SCHEDULING ( bids) returns solution
    $TEMP = \phi, TEMP^* = \phi$, and $sum^* = 0$
    RECURSIVE-BRANCHING ( $bids$ , 0 )
    return $TEMP^*$
function RECURSIVE-BRANCHING ( $AV$ , $sum$ )
if $sum > sum^*$ then $sum^* \leftarrow sum$ , $TEMP^* \leftarrow TEMP$
if $AV = \phi$ then return
$h \leftarrow \sum_{bid \in AV} priceOf(bid)$
If $sum + h \le sum^*$ , then return
$bid \leftarrow$ SELECT-UNASSIGNED-BID ( $AV$ )
```

```
$TEMP = TEMP \cup \{bid\}$ , $AV = AV - \{bid\}$
$sum \leftarrow sum + priceOf\,(bid)$
**if** CHECK-FEASIBILITY ( $TEMP$ ) returns pass
   **then** RECURSIVE-BRANCHING ( $AV$ , $sum$ )
$TEMP \leftarrow TEMP - \{bid\}, sum = sum - priceOf\,(bid)$
RECURSIVE-BRANCHING ( $AV$ , $sum$ )
add $bid$ to $AV$ , **return**
```

The function CHECK-FEASIBILITY used in Algorithm 1 validates the feasibility of each node (a set of winner bids) along the search patch. This checking process is equivalent to solving a job shop constraint satisfaction problem. We have implemented the CHECK-FEASIBILITY function using a constraint-directed backtrack search procedure. Usually, a constraint-directed search procedure consists of propagators, heuristic-commitment techniques and retraction techniques. The feasibility validation algorithm integrates Constraint-Based Analysis (a propagator, developed in Beck and Fox (1998), Precedence Constraint Posting (a commitment heuristic, developed in Smith and Cheng (1993)), and a chronological backtracking. Details of the algorithm can be found in Wang *et al.* (2009b).

It should be noted that the winner determined algorithm described above allows agents to negotiate only over a single attribute which is price. For the single attribute negotiation, each agent needs to submit a simple atomic bid to express the price it is willing to pay for the job to be completed within a specific completion time interval. To accommodate the multi-attribute bidding protocol proposed in the next section, we expand the single attribute model by allowing agents to negotiate over both price and completion time. An agent's valuation over these two attributes can be expressed by a set of atomic bids connected by XOR. However, at most one atomic bid of an XOR-Bid can be included in a provisional schedule. To handle this logic, we add a checking mechanism to the winner determination algorithm presented above to prevent the algorithm from including more than one atomic bid from the same XOR-Bid into a provisional schedule. The checking mechanism is implemented in the Select-Unassigned-Bid (AV) method of Algorithm 1. When the method selects an unassigned bid, it first checks the current schedule. If there is a bid from the same XOR-Bid has already been included, the unassigned bid will be excluded from the selection.

4.3. The Iterative Multi-Attribute Bidding Protocol

A*d*SCHE is a multi-attribute auction framework, which allows the negotiation over price and a non-price attribute: the completion time of a schedule. In addition, A*d*SCHE has good privacy preserving properties. For example, unlike general combinatorial auctions, A*d*SCHE does not require agents' knowledge about the resources, such as their capabilities, availabilities and configurations. Also, before the auction starts, the auctioneer does not have to know what kinds of services that agents will require.

Figure 2 shows a generic view of the bidding and pricing process of A*d*SCHE. Before the bidding starts, agents may initialize the ask prices of their processing requirements (jobs and makespan-intervals) based on "common knowledge" about the cost of the processing requirements. Appropriate setting-up of initial ask prices can speed up the overall bidding process and, at the same time, maintain the solution quality. If an agent has no information about the cost of its processing requirements, it can set the initial ask prices as zero. However, in A*d*SCHE, agents have the incentive to calculate the right reserve prices for their processing requirements. For agents, it is irrational to submit any bids below reserve prices because those bids will be rejected by the auctioneer at the bids screening stage. An alternative way is to acquire initial ask prices from the auctioneer before the bidding starts. After setting up the initial ask prices for the processing requirements, agents calculate the utility maximizing bids based on the ask prices and start the bidding process. In each round of the auction, the auctioneer first screens out illegal bids (defined in section 4.3.2). Those bids will not be considered in the following winner determination procedure. Then, the auctioneer check the termination condition (defined in section 4.3.2). If the condition holds, the auctioneer implements the current provisional schedule and terminates the auction. Otherwise, it uses the winner determination algorithm to compute a new provisional schedule based on the bids collected. After winner determination, the ask prices of processing requirements of losing bids are increased by a minimum price increment determined by the auctioneer and these increased prices are sent out to the agents who own the processing requirements. Upon receiving the price update, the agents recalculate the utility maximization bids and start a new round of bidding.

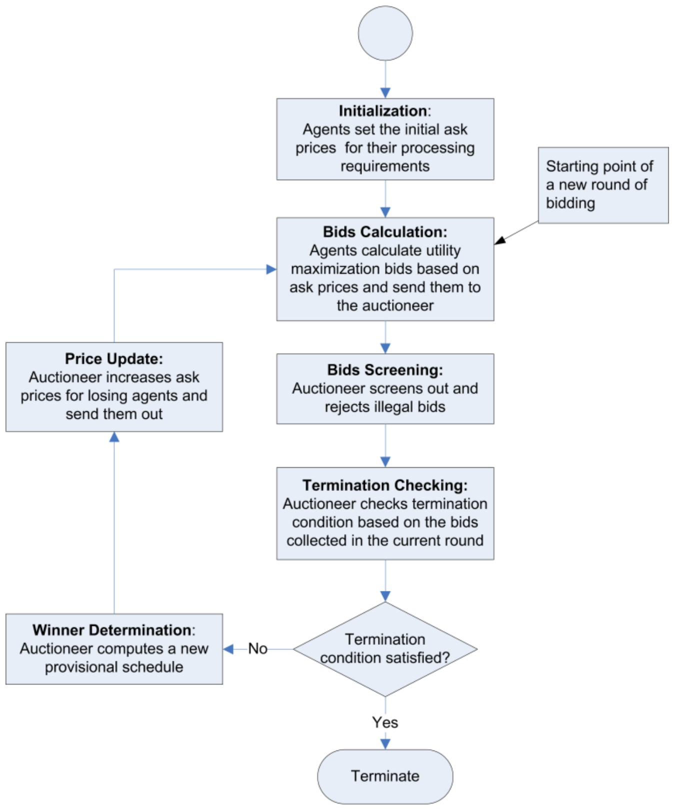
Initialization: Agents set the initial ask prices for their processing requirements
Starting point of a new round of bidding
Bids Calculation: Agents calculate utility maximization bids based on ask prices and send them to the auctioneer
Price Update: Auctioneer increases ask prices for losing agents and send them out
Bids Screening: Auctioneer screens out and rejects illegal bids
Termination Checking: Auctioneer checks termination condition based on the bids collected in the current round
Winner Determination: Auctioneer computes a new provisional schedule
No
Termination condition satisfied?
Yes
Terminate

Fig. 2 Overview of the bidding and pricing process in AdSCHE

4.3.1. Bidding

Using the L_R bidding language, each agent can express its values over different completion time intervals as an XOR-Bid. We assume the reserve prices for processing jobs within different completion time intervals is common knowledge. For the first round of the bidding, agents use the reserve prices as ask prices. At the beginning of round t, an agent g selects the set of completion time intervals that maximize its utility function given the ask prices. It then generates an XOR-Bid that represents the set of utility maximization completion time intervals. Note that all atomic bids in the XOR-Bid equally maximize the utility function of agent g given their completion time intervals and the associated prices. That is for any two atomic bids in the XOR-Bid at round t, $v_j(eft_j, lft_j) - p_{bid,t}(eft_j, lft_j) = v_k(eft_k, lft_k) - p_{bid,t}(eft_k, lft_k)$, where $v_k(eft_k, lft_k)$is the agent's valuation on the completion time interval $(eft_k, lft_k]$, and $p_{bid,t}(eft_k, lft_k)$ is the price that the agent bid on $(eft_k, lft_k]$ at round t .

Given the ask prices, the following bids are allowed for an agent:

- If the agent is notified to be included in the current provisional schedule, it replies with an empty bid as the acknowledgement.
- If the agent is not included in the current provisional schedule, two types of bids are allowed: (1) the agent can bid either at or greater than the ask prices; (2) the agent can also bid within the minimum price increment, denoted by ε, below the ask price. This is called "ε-discount" in (Parkes, 1999), which allows agents to bid for the completion time intervals priced slightly above their values. However, if an agent takes this discount, the auctioneer will consider the discounted bid as the *final bid* from the agent for the completion time interval and the agent is forbidden from bidding the completion time interval in future rounds.

Since agents are assumed to be rational in maximizing their utilities, they, in general, do not bid greater than the ask price. However, if for some reasons (e.g. to reduce computation and/or communication cost) an agent does not want to bid frequently, it can place a much higher bid (but still under its value) than the ask price to increase the possibility that its bids will be included in the provisional schedules for a certain number of consequent rounds.

4.3.2. Termination

After receiving bids from agents, the auctioneer screens out invalid bids, which are defined as follows:

- Bids are not within the ε below the ask prices.
- Bids for makespan-intervals that have been bidden by final bids.
- Bids are below the reserve prices.
- Empty bids from agents that are not included in the provisional schedule from last round.

The auctioneer then checks the termination condition against the valid bids. In A*d*SCHE, the auction terminates if all valid bids collected by the auctioneer are empty bids. That is, all agents that bid in the last round are included in the current provisional schedule. No agent is unhappy. After the auction terminates, the auctioneer implements the final schedule and the agents pay their bidding prices.

4.3.3. Winner Determination and Price Update

The auctioneer needs to compute a new provisional schedule in each round as long as the auction is not terminated. At round t, the new provisional schedule S_t solves:

$$\max_{S_t} \sum_{\substack{g \in N,\, S_t^g \in S_t \\ eft_k < C_{\max}(S_t^g) \le lft_k}} p_{bid,t}^{g}(eft_k, lft_k)$$

where is $p_{bid,t}^{g}(eft_k, lft_k)$ is the bidding price of agent g at round t for the completion time interval $(eft_k, lft_k]$, S_t^g is the part of S_t that represents the schedule of the job from agent g at round t, and $C_{max}(S_t^g)$is the completion time of S_t^g.

In A*d*SCHE, prices are attached to the job processing requirements of agents. There is an ask price for each makespan-interval that an agent may require. For the makespan-interval $(eft_k, lft_k]$, of agent g, the ask price set by the auctioneer at the end of round t is $p_{ask,t}^{g}(eft_k, lft_k)$. At the beginning of round $t+1$, if agent g does not bid on $p_{ask,t}^{g}(eft_k, lft_k)$, the price for $(eft_k, lft_k]$ of agent g will not be updated in round $t+1$, that is $p_{ask,t+1}^{g}(eft_k, lft_k) = p_{ask,t}^{g}(eft_k, lft_k)$. On the other hand, if agent g bids on completion time interval $(eft_k, lft_k]$ in round $t+1$ with price $p_{bid,t+1}^{g}(eft_k, lft_k)$, the auctioneer will update the ask price for $(eft_k, lft_k]$ based on the result of winner determination and whether agent g has taken a discounted bid:

- If agent g wins, the new ask price $p^{g}_{ask,t+1}(eft_k, lft_k) = P^{g}_{bid,t+1}(eft_k, lft_k)$.
- If agent g loses, the new ask price $p^{g}_{ask,t+1}(eft_k, lft_k) = P^{g}_{bid,t+1}(eft_k, lft_k) + \varepsilon$.

If $p^{g}_{ask,t}(eft_k, lft_k) > p^{g}_{bid,t+1}(eft_k, lft_k) > p^{g}_{ask,t}(eft_k, lft_k) - \varepsilon$, no matter agent g wins or loses, the auctioneer will not update the ask price for $(eft_k, lft_k]$ because agent g has taken a discounted bid which the auctioneer considers as the final bid from agent g for makespan interval $(eft_k, lft_k]$. In this case, agent g will not be allowed to bid on $(eft_k, lft_k]$ in future rounds.

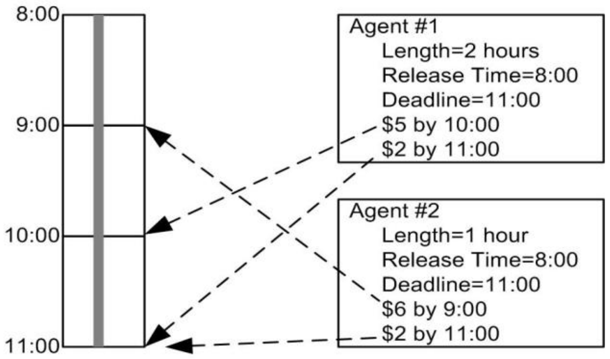

Fig. 3 An example of decentralized scheduling problems, in which two agents compete for the processing of their one-operation jobs

4.3.4. An Example

This section presents a worked example of A*d*SCHE. As shown in Figure 3, a three hour time window (from 8:00-11:00) of a resource is being scheduled. Agents have single-operation jobs to be processed. An agent's job is defined by its duration (length), its release time, its deadline, and the prices (expressed in dollars) the agent places on the different completion times of the job. For example, if the job of Agent#1 can be finished by 10:00, the agent is willing to pay $5; if the job is completed anywhere between 10:00 and 11:00, the agent is willing to pay $2. To complete its job, the agent must acquire a period of processing times no less than the length of its job, within its feasible time window (the time period between its release time and its deadline).

Table 2 Bidding process of applying A*d*SCHE to an example decentralized scheduling problem

Round #	Ask Prices				Bid Prices				Allocation		Auctioneer Revenue	Objective value.
	Agent-1		Agent-2		Agent-1		Agent-2					
	(8,10] $5	(10,11] $2	(8,9] $6	(9,11] $2	(8,10] $5	(10,11] $2	(8,9] $6	(9,11] $2	Agent-1	Agent-2		
1	2	2	1	1	2		1		(8,10]		2	5
2	2	2	3	1	2		3			(8,9]	3	6
3	4	2	3	1	4		3		(8,10]		4	5
4	4	2	5	1	4		5	1		(8,9]	5	6
5	6	2	5	1	5	2	5	1	(10,11]	(8,9]	7	8

For this example and other decentralized scheduling problems, we may construct an auction in which an agent can bid time periods or their combinations within its feasible time window for processing its jobs. By doing so, we are mapping the decentralized scheduling problem to a type of combinatorial auction problems, for which the goods to be sold are continuous processing time periods and no time window discretization is imposed on resources. AdSCHE is the auction framework designed for this type of combinatorial auction problems. In Figure 3, Agent #1's valuation can be expressed by XOR-Bid: $\langle j_1, 8,10, \$5\rangle XOR\langle j_1, 10,11, \$2\rangle$, where $j_1$is the job of Agent #1; Agent #2's valuation can be expressed by XOR-Bid $\langle j_2, 8,9, \$6\rangle XOR\langle j_2, 9,11, \$2\rangle$ where $j_2$is the job of Agent #2. Assume that the resource has the reserve price of 1 dollar an hour and the price increment $\varepsilon = \$2$. The ask prices, bid prices and allocation of each round of the auction are shown in Table 2. The process is described as follows:

1) At round #1 the agents use the reserve prices for their job requirements as the ask prices. Agent1 bids on makespan-interval (8,10] and Agent2 bids on makespan-interval (8,9] because given the current ask prices, these two makespan-intervals maximize agents' utility functions. The auctioneer includes only Agent1 into the provisional schedule because the two bids from agents cannot coexist in a schedule and Agent1's bid maximize auctioneer's revenue. The auctioneer increases the ask price of Agent2's makespan-interval (8,9] to \$3.

2) Round #2&#3 repeat the process of round #1. In these rounds each agent only has one makespan-interval that maximizes its utility function. So it only submits a bid at each round. However, in round #4, both makespan-intervals (8,9] and (9,11] maximize Agent #2's utility function, therefore, at this round, Agent #2 submits an XOR-Bid:$\langle j_2, 8,9, \$5\rangle XOR\langle j_2, 9,11, \$1\rangle$.

3) At round #5, Agent #2 repeats its bid because it was included in the provisional schedule in round #4. The ask price for Agent1's (8,10] has been increased to \$6 which is \$1 above its value on (8,10] because the minimum price increment is \$2. In this case Agent1 can place a bid on (8,10]at any price from \$4 to \$5. However it will not be allowed to increase the price in the rounds after. As shown in Table 2, Agent #1 bids \$5 on (8,10] and the final schedule turns out to be that Agent #2 precedes Agent #1, which is optimal.

4.4. Computational Evaluation

This section evaluates A*d*SCHE through computational analysis. We first introduce the metrics used to evaluate A*d*SCHE's performance.

4.4.1. Metrics

As mentioned at the beginning of this section, we evaluate the iterative bidding framework in terms of efficiency, computation (running time), revenue and information revelation. These metrics were developed in (Parkes, 1999) for testing the performance of iBundle, an iterative combinatorial auction for general combinatorial auction problems. We redefine them in the context of decentralized scheduling.

Efficiency of Scheduling, $eff(S)$, is measured as the ratio of the value of the final schedule S to the value of the optimal schedule that maximizes total value across the agents:

$$eff(S) = \frac{\sum_{lft_{j,i} \in S} v_j(lft_{j,i})}{\sum_{lft_{j,i} \in S^*} v_j(lft_{j,i})} ,$$

where S^* is the optimal schedule given customers' valuations.

Running Time of Auction refers to the computation time needed to terminate the auction on a decentralized scheduling problem instance.

Information Revelation for customer j, $inf(j)$, is measured as the sum of the final price bid by the customer for all due dates in its valuation function, as a fraction of the sum of the true values of each due date.

$$inf(j) = \frac{\sum_{i=0}^{m_j} \overline{p}_j(lft_{j,i})}{\sum_{i=0}^{m_j} v_j(lft_{j,i})},$$

The overall auction information revelation is computed as the average information revelation over all agents. The auction often terminates before agents have revealed complete information about their values for due dates. The information revelation metric is designed to measure the extent to which an agent has revealed its value for each due dates to the auctioneer during the auction.

4.4.2. Comparison Results

A*d*SCHE is tested on a set of scheduling auction problems with multiple completion-time-interval valuations, which are generated based on the single completion-time-interval benchmark problem set used in Wang *et al.* (2009b). We take problem instances of the single completion-time-interval set and add two more completion-time-interval valuations to each problem instance to generate a multiple completion-time-interval set. The first completion-time-interval added represents a delay up to 20% and the agent's value on the delayed makespan decreases 20%. The second added represents a 20% to 40% delay and, accordingly, the agent's value decreases 40% on the delayed makespan. For example, if the single completion-time-interval valuation of agent g on a set of jobs J^g can be represented as an atomic bid, $\langle J^g, 8,10,\$10\rangle$ the multiple-makespan-interval valuation of the agent can be represented as an XOR-Bid $\langle J^g, 8,10,\$10\rangle XOR\langle J^g, 10,12,\$8\rangle XOR\langle J^g, 12,14,\$6\rangle$. By this way, we generated 9 groups of decentralized scheduling problem instances.

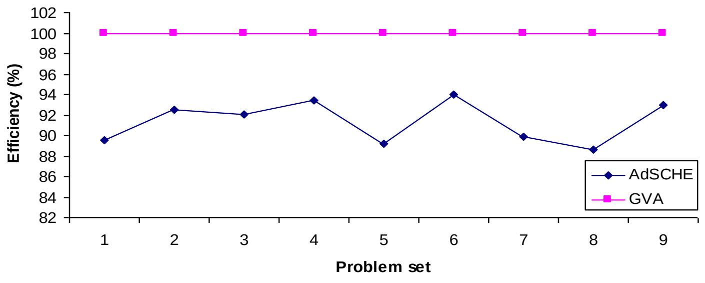

Fig. 4 Efficiency performance of AdSCHE over 9 problem sets with bid increment

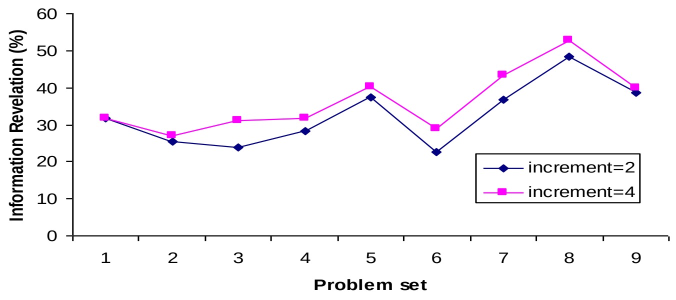

Fig. 5 Information revelation performance of AdSCHE over 9 problem sets with bid increment $\varepsilon = 4$ and $\varepsilon = 2$

We compare A*d*SCHE with a Generalized Vickery Auction (GVA) in which agents report their complete valuations over different makespan-intervals at the beginning of the auction and the auctioneer computes the optimal schedule for agents. In both A*d*SCHE and the GVA, We have used the same constraint-based winner determination algorithm. Therefore, the comparison results presented in this section only reflect the difference between the iterative bidding structure (in A*d*SCHE) and the one-shot bidding structure (in GVA) in the context of auction-based decentralized scheduling.

Figure 4 plots the efficiency of A*d*SCHE over the 9 problem sets with bid increment $\varepsilon = 4$. Compared to GVA (100% efficiency), on average, A*d*SCHE can achieve more than 90% efficiency. This efficiency level is comparable to some iterative auctions such as AUSM (Ledyard, et al., 1997) and RAD (DeMartini et al., 1999). However, it is not as good as *i*Bundle. Figure 5 plots the Information Revelation performance of A*d*SCHE. Compared to GVA which requires 100% Information Revelation, A*d*SCHE requires less than 50% at increment=2 and increment=4. Bigger increment value requires slightly more Information Revelation. This makes sense because bigger increments may pass some low price equilibrium point which smaller increments may find.

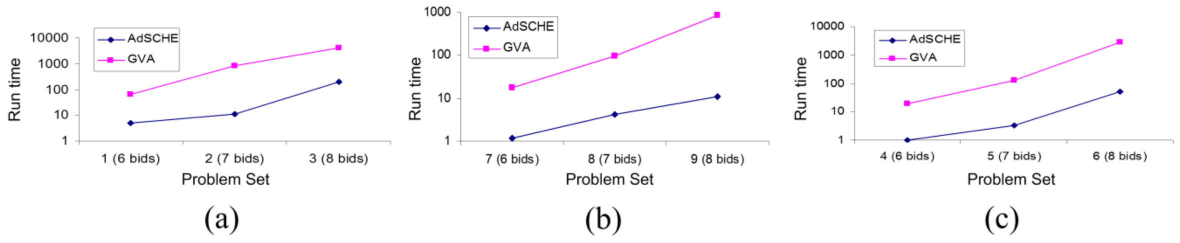

Fig. 6 Run time (in seconds) of GVA and AdSCHE as the problem difficulty is increased. (a) problem sets #1, #2, and #3; (b) problem sets #4, #5, and #6; (c) problem sets #7, #8, and #9)

Figure 6 compares the run time increases between A*d*SCHE and GVA as the number of bids increases from 6 to 8. We classify the problem sets into 3 groups as shown in (a), (b), and (c). All 3 groups demonstrate similar run time increasing pattern as problem difficulty (number of bids) increases. On average, A*d*SCHE is more than 10 times faster than GVA with the cost of losing 6%-10% efficiency as shown in Figure 6. We have set an 8000 seconds time limit. The computation times of GVA for half of 8-bid instances in group (b) and all 8-bid instances in group (a) have reached the limit. We did not test instances with more than 8 bids.

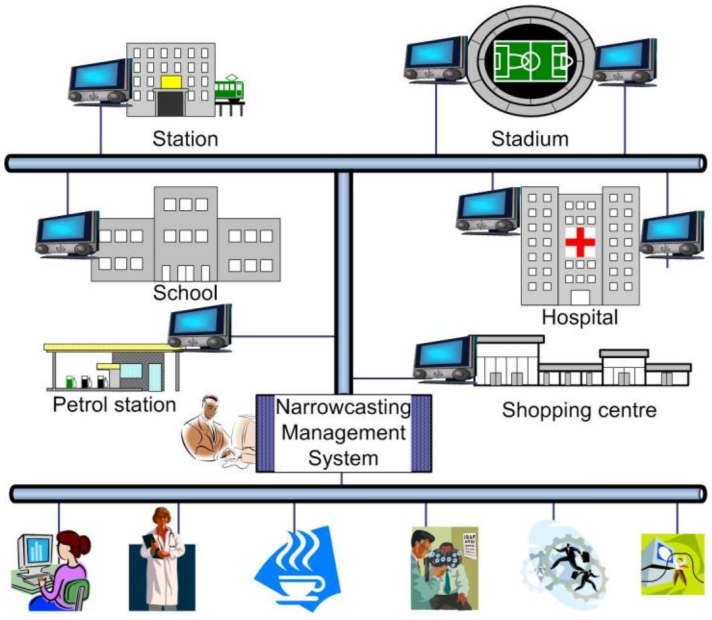

Fig. 7 A narrowcasting environment

5. A Case Study

This section presents a case study regarding the application of A*d*SCHE to decentralized media content scheduling in narrowcasting. Narrowcasting is a newly emerged technology which enables companies and organizations to target audiences with an unprecedented level of customization and timeliness through media devices such as commercial digital displays. Figure 7 shows a distributed narrowcasting environment in which a number of displays are located in various types of places in a community including stadiums, hospitals, shopping centers, gas stations, schools, and train stations, etc. The displays are connected to the Narrowcasting Management System through high speed network. Main functionalities of the management system include promotion planning, media content scheduling and media file distribution. Customers who want to use the network to deliver messages to their audiences are also connected to the management System.

Table 3 The media content scheduling problem configuration

Customer	Content	Location	Repetition limit	Earliest starting Time	Due Date	Length (in minutes)	Value
	1	1	4	8:00 AM	8:30 AM	8	$24
	2	5	2	8:00 AM	8:30 AM	4	$34
	3	4	2	8:00 AM	8:30 AM	6	$44
	4	3	2	8:00 AM	8:30 AM	14	$54
A	5	2	2	8:00 AM	8:30 AM	4	$24
	1	5	1	8:10 AM	8:30 AM	8	$50
	2	2	1	8:10 AM	8:30 AM	6	$50
	3	1	1	8:10 AM	8:30 AM	7	$50
	4	3	1	8:10 AM	8:30 AM	15	$50
B	5	4	1	8:10 AM	8:30 AM	10	$50
	1	4	1	8:00 AM	8:35 AM	5	$34
	2	1	1	8:00 AM	8:35 AM	4	$34
	3	2	1	8:00 AM	8:35 AM	9	$34
	4	3	1	8:00 AM	8:35 AM	9	$34
C	5	5	1	8:00 AM	8:35 AM	5	$34
	1	1	1	8:00 AM	8:35 AM	6	$52
	2	5	1	8:00 AM	8:35 AM	11	$52
	3	2	1	8:00 AM	8:35 AM	9	$52
	4	3	1	8:00 AM	8:35 AM	15	$52
D	5	4	1	8:00 AM	8:35 AM	3	$52

We demonstrate the applicability of A*d*SCHE to decentralized media content scheduling problems using a multiple display problem setting. Consider a decentralized content scheduling problem with 4 customers who need promotion service at 5 different locations for a morning advertising time window from 8:00AM to 8:45AM. We assume that one location has a single display to be scheduled. Each customer has a specific content file for a location. A content file has an earliest starting time, after which the content can be displayed, and a due date, which is the preferred time before which the displaying of the content is finished. The customer has a fixed value on a piece of content as long as it is displayed between its earliest starting time and its due date. The customer's value on a piece content decreases 20% if the finishing time of displaying delays by 20% according to the due date. The value will decreases 40% if the finishing time delays 40%. The customer's value reaches zero for any delays greater than 40% or any finishing times later than 8:45AM. A piece of content can be displayed multiple times within the advertising time window. However, the customer has a limit on how many times she wants the content to be displayed. There are no complements or substitutes in terms of a customer's valuation on her different pieces of media content and copies of the same piece of media content. The customer wants her contents to be displayed as much as possible as long as the cost of displaying is below her valuation. Table 3 shows the detailed configuration of this problem. The *value* column refers to the value that a customer has on the displaying of a piece of content between the earliest starting time and due date. The objective is to compute a promotion schedule over five displays such that the customers' location, earliest starting time, and deadline requirements are satisfied, the sum of customers' promotion values is maximized.

The bid structure in this case study is more complicated than that in the single display example in Section 4.3.4. Since customers have multiple pieces of content to be displayed and the customer's value varies based on the finishing times of displaying, they need to construct three bids for each piece of content to express the valuations on meeting due date, 20% delay, and 40% delay respectively. In addition, if a piece of content may be displayed multiple times, bids are also need to be constructed to represent the multiple copies of the content.

All bids from a customer need to be joined by OR and XOR connectives. For example, the valuation of customer A on content 2 can be represented using the following bid, in which the valuations for two copies of content A-2 are joined by an OR connective:

$$\begin{pmatrix} \langle A2, location5, 8:00AM, 8:30AM, \$34\rangle XOR \\ \langle A2, location5, 8:00AM, 8:36AM, \$27.2\rangle XOR \\ \langle A2, location5, 8:00AM, 8:42AM, \$20.4\rangle \end{pmatrix} OR \begin{pmatrix} \langle A2, location5, 8:00AM, 8:30AM, \$34\rangle XOR \\ \langle A2, location5, 8:00AM, 8:36AM, \$27.2\rangle XOR \\ \langle A2, location5, 8:00AM, 8:42AM, \$20.4\rangle \end{pmatrix}$$

A complete valuation of customer A consists of bids for all 5 pieces of content joined by OR connectives.

Figure 8 shows the promotion schedule generated by A*d*SCHE. In the schedule, content A-1 (content 1 from customer A), A-2, A-3, A-5 are displayed two times. B-4 losses in the bidding, therefore, it is not included in the schedule. The minimum bid increment is set to be \$4 and the auction terminates at round 31 with an overall value of \$929.

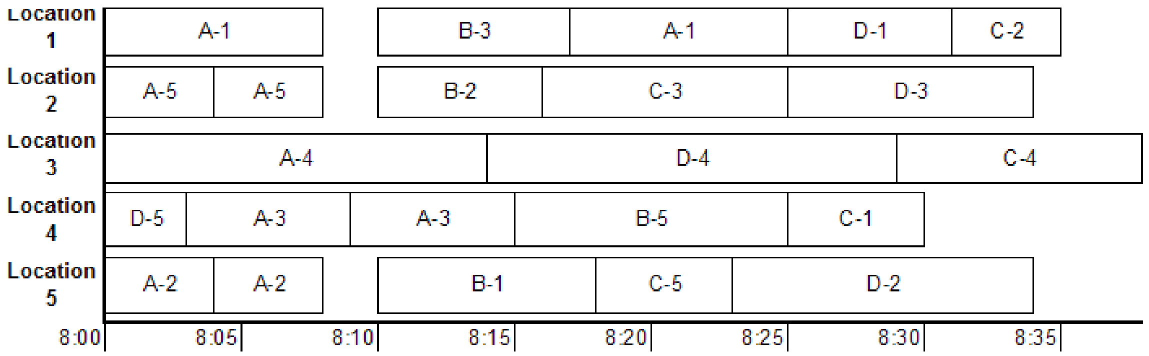

Fig. 8 Promotion schedule for the multiple display media content scheduling case

We have demonstrated how A*d*SCHE can be applied to the decentralized media content scheduling model. The advantage of applying A*d*SCHE to this domain is that the complicated customer content displaying requirements, such as location, earliest starting time, latest finishing time, and times to be displayed, can be concisely captured by the requirement based bidding language. Also, the language allows customers to express different promotion values based on the completion times of content displaying. However, the media content scheduling problem is less-constrained compared to other scheduling problems, such as decentralized job shop scheduling, since there are no precedence constraints between content copies. For the less-constrained problems, constraint-based winner determination may not be effective[4]. For large scale problems, the lack of efficiency in winner determination may lead to unpractical solving times. In future research, we will expand A*d*SCHE to meet the requirements raised from large industry scale decentralized media content scheduling problems. In particular, winner determination algorithms designed toward the media content scheduling formulation and approximate winner determination algorithms will be considered.

6. Conclusion and future work

In many combinatorial auction problems, the items to be sold are processing times of resources. As these problems primarily concern the allocation of jobs on resources overtime in decentralized environments, they fall into the category of decentralized scheduling problems. The proposed A*d*SCHE framework promises to outperform general combinatorial auctions on decentralized scheduling problems by utilizing scheduling domain specific bidding languages and winner determination algorithms. Our analysis and experiments show that A*d*SCHE posses good computational properties in terms of agents' valuation, communication and winner determination. We outline three directions that can be pursued in terms of expanding the current work from the perspective of improving its applicability to real world scale applications.

First, we will continue improving the efficiency of the constraint-based winner determination. The actual algorithms developed in the thesis have demonstrated good performance in auction-based decentralized scheduling. However, it can be further improved along several directions. For example, more sophisticated branching and bids ordering heuristics can be introduced. In addition, more heuristics from classical scheduling theory can be embedded to boost the performance of the approach on well studied scheduling problem models. We will also explore the possibility of introducing approximate and heuristic algorithms for the winner determination problem. While these algorithms can come with different flavors, those that preserve incentive compatibility are worth of investigation.

Another important future direction is to expand the current A*d*SCHE model to support online decentralized scheduling environments. In many decentralized scheduling environments, for example, supply chain or service industries, customers' requests for processing a set of jobs may be received over time. Furthermore, customers may need the response (e.g. price and due date quotations) to their requests immediately or within a short period of time. In these cases, it is not practical to synchronize the customers' requirements and ask all customers to start the bidding process at a predefined time point. By allowing customers to submit their requests at any time, we have introduced the dynamic version of decentralized scheduling problems. Developing scheduling systems for dynamic scheduling problems presents new challenges, that is, the system needs to repair the schedule or redo the scheduling whenever a dynamic event makes the original scheduling infeasible. For A*d*SCHE, the iterative bidding procedure adopted has provided a good potential in accommodating dynamic events. Since bidding proceeds in rounds, a newly arriving agent may pick a round in the process and join in the bidding. One avenue worth exploring is to build the support for dynamic scheduling within the iterative bidding structure.

At the current stage, we have considered the situation where agents participate in one auction at a time. However, in some scheduling applications, agents are typically interested in obtaining multiple schedulable resources from several markets. To complete the job, an agent needs to secure all the resources required. The strong complementarities among resources induced by such preferences pose strategic problems for bidding agents. While each market can be constructed using an A*d*SCHE

[4] It is observed in the computational study for constraint-based winner determination that problems with tight constraints tend to be solved quicker by the constraint-based winner determination algorithm (Wang *et al.*, 2009).

system, exploring agents' bidding policies, which allows coordination across separate auction-based scheduling markets, would be another step toward practical approaches to real world decentralized scheduling applications

7. References

Beck, J.C. and Fox, M.S. "A Generic Framework for Constraint-Directed Search and Scheduling," AI Magazine, vol. 19, No. 4, pp. 101-130, 1998.

Boutilier, C., Goldszmidt, M., and Sabata, B., "Sequential auctions for the allocation of resources with complementarities," in Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, 1999, pp. 527-534.

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

Brewer, P. J. and Plott, C. R., "A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks," International Journal of Industrial Organization, vol. 14, pp. 857-886, 1996

Buyya, R., Abramson, D., Giddy, J., and Stockinger, H., "Economic models for resource management and scheduling in Grid computing," Concurrency and Computation: Practice and Experience, vol. 14, pp. 1507-1542, 2002.

DeMartini, C., Kwasnica, A. M., Ledyard, J. O., and Porter, D., "A new and improved design for multi-object iterative auctions," Technical report SSWP 1054, CalTech, March 1999.

de Vries, S. and Vohra, R.V., "Combinatorial Auctions: Survey," INFORMS journal on Computing, vol.15, no.3, pp.284-309, 2003.

Engelbrecht-Wiggans, R. and Weber, R. J., "A sequential auction involving asymmetrically-informed bidders," International Journal of Game Theory, vol. 12, pp. 123-127, 1983.

Garey, M.R. and Johnson, D. S., Computers and Intractability, a Guide to the Theory of NP-Completeness, W. H. Freeman Company, 1979.

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

Kutanoglu, E. and Wu, S. D., "Incentive compatible, collaborative production scheduling with simple communication among distributed agents," International Journal of Production Research, vol. 44, pp. 421-446, 2006.

Ledyard, J.O., Porter, D., and Rangel, A., "Experiments testing multiobject allocation mechanisms," Journal of Economic and Management Strategy, vol. 6, pp. 639-675, 1997.

Mackie-Mason, J. K., Osepayshvili, A., Reeves, D. M., and Wellman, M. P., "Price prediction strategies for market-based scheduling," in Proc. 14th International Conference on Automated Planning and Scheduling, Whistler, BC, 2004.

McAfee, R. P. and McMillan, J., "Analyzing the airwaves auction," The Journal of Economic Perspectives, vol. 10, pp. 159-175, 1996.

Milgrom, P., "Putting auction theory to work: the simultaneous ascending auction," Journal of Political Economy, vol. 108, pp. 245-272, 2000.

Parkes, D. C., "iBundle: An Efficient Ascending Price Bundle Auction," In Proc. 1st ACM Conf. on Electronic Commerce (EC-99), pp. 148-157, 1999.

Parkes, D. C. and Ungar, L. H., "Iterative combinatorial auctions: theory and practice," in Proc. 17th National Conference on Artificial Intelligence, Austin, TX. , 2000, pp. 74–81.

Parkes, D. C. and Ungar, L. H., "An auction-based method for decentralized train scheduling," in Proc. 5th International Conference on Autonomous Agents, Montreal, Quebec, 2001, pp. 43-50.

Pinedo, M., Scheduling: Theory, Algorithms, and Systems: Prentice Hall, 2002.

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.

Reeves, D. M., Wellman, M. P., MacKie-Mason, J. K., and Osepayshvili, A., "Exploring bidding strategies for market-based scheduling," Decision Support Systems, vol. 39, pp. 67-85, 2005.

Sandholm, T., "Distributed rational decision making," in Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, Cambridge, MA: MIT Press, 1999.

Sandholm, T., "Algorithm for optimal winner determination in combinatorial auctions," Artificial Intelligence, vol. 135, pp. 1-54, 2002.

Sandholm, T., Suri, S., Gilpin, A., and Levine, D., "CABOB: a fast optimal algorithm for winner determination in combinatorial auctions," Management Science, vol. 51, pp. 374-390, 2005.

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

Shen, W., Wang, L., and Qi, H., "Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 36, pp. 563-577, 2006.

Smith, A., The wealth of nations, Cannan edition (Modern Library, New York), 1937.

Smith, R. G., "The contract net protocol: High-level communication and control in a distributed problem solver," IEEE Transactions on computers, vol. 100, pp. 1104-1113, 1980.

Smith, S.F. and Cheng, C., “Slack-Based Heuristics for Constraint Satisfaction Scheduling,” In Proceedings of 11th National Conference on Artificial Intelligence. Washington D.C., July 1993.

Smith, S.F. and Becker, M.A., "An Ontology for Constructing Scheduling Systems", Working Notes of 1997 AAAI Symposium on Ontological Engineering, Stanford, CA, March. AAAI Press

Walsh, W. E. and Wellman, M. P., "Efficiency and equilibrium in task allocation economies with hierarchical dependencies," in Proc. 6th International Joint Conference on Artificial Intelligence, Stockholm, 1999, pp. 520-526.

Wang, C., Ghenniwa, H., and Shen, W., "Bidding languages for auction-based distributed scheduling," in Proc. 2009 IEEE International Conference on Systems, Man, and Cybernetics San Antonio, Texas, 2009a, pp. 4518-4523.

Wang, C., Ghenniwa, H. H., and Shen, W. Constraint-based winner determination for auction-based scheduling. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 39(3), 609-618, 2009b.

Wellman, M. P., "A computational market model for distributed configuration design," AI EDAM, vol. 9, pp. 125-133, 1995.

Wellman, M. P., W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason, “Auction Protocols for Decentralized Scheduling”, Games and Economic Behavior, vol.35, no.1-2, pp.271-303, 2001.

8. Authors’ Biographies

Dr. Chun Wang received the B.Eng. degree from Huazhong University of Science and Technology, Wuhan, China, in 1990, and the M.E.Sc. and Ph.D. degrees in computer engineering from The University of Western Ontario, in 2004 and 2007, respectively. He is currently an Assistant Professor with Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada. He has many years of experience in the software industry. From 1990 to 2000, he was a Software Engineer and a Project Manager with the China National Petroleum Company, Liaoning China. His research focuses on distributed systems, electronic supply chain management, services engineering, algorithmic mechanism design, and multiagent systems.