NRC Publications Archive Archives des publications du CNRC

Constraint-based winner determination for auction-based scheduling

Wang, C.; Ghenniwa, H.; Shen, W.

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version. / La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version acceptée du manuscrit ou la version de l'éditeur.

For the publisher's version, please access the DOI link below. / Pour consulter la version de l'éditeur, utilisez le lien DOI ci-dessous.

Publisher's version / Version de l'éditeur:

<https://doi.org/10.1109/TSMCA.2009.2013194>

IEEE Transactions on Systems, Man and Cybernetics, Part A, 39, 3, pp. 609-618, 2009-05-01

NRC Publications Record / Notice d'Archives des publications de CNRC:

<https://nrc-publications.canada.ca/eng/view/object/?id=2b69f79c-187e-497a-aea1-0185ae8f6a5e>

<https://publications-cnrc.canada.ca/fra/voir/objet/?id=2b69f79c-187e-497a-aea1-0185ae8f6a5e>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at

<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site

<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at

PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

<http://www.nrc-cnrc.gc.ca/irc>

Constraint-based winner determination for auction-based scheduling

NRCC-50313

Wang, C.; Ghenniwa, H.; Shen, W.

May 2009

A version of this document is published in / Une version de ce document se trouve dans:
IEEE Transactions on Systems, Man and Cybernetics, Part A, 39, (3), pp. 609-618, May 01, 2009, DOI: [10.1109/TSMCA.2009.2013194](https://doi.org/10.1109/TSMCA.2009.2013194)

The material in this document is covered by the provisions of the Copyright Act, by Canadian laws, policies, regulations and international agreements. Such provisions serve to identify the information source and, in specific instances, to prohibit reproduction of materials without written permission. For more information visit <http://laws.justice.gc.ca/en/showtdm/cs/C-42>

Les renseignements dans ce document sont protégés par la Loi sur le droit d'auteur, par les lois, les politiques et les règlements du Canada et des accords internationaux. Ces dispositions permettent d'identifier la source de l'information et, dans certains cas, d'interdire la copie de documents sans permission écrite. Pour obtenir de plus amples renseignements : <http://lois.justice.gc.ca/fr/showtdm/cs/C-42>

National Research
Council Canada

Conseil national
de recherches Canada

Canada

Constraint-Based Winner Determination for Auction-Based Scheduling

Chun Wang, *Member, IEEE*

Hamada H. Ghenniwa

Department of Electrical and Computer Engineering,

The University of Western Ontario

London, Ontario, Canada, N6A5B9

+1 519 6612111-80334, +1 519 6612111-88262

chun.wang@ieee.org, hghenniwa@eng.uwo.ca

Weiming Shen, *Senior Member, IEEE*

National Research Council Canada

800 Collip Circle, London, Ontario, Canada

N6G4X8

Tel: +1 519 4307134

Weiming.shen@nrc.ca

*** An earlier version of this paper was presented in the International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), May 14-18, 2007, Honolulu, Hawaii.

Abstract — This paper presents a formulation and an algorithm for the winner determination problem in auction-based scheduling. Without imposing a time line discretization, the proposed approach allows bidders to bid for the processing of a set of jobs under scheduling constraints using a requirement-based bidding language. The proposed winner determination algorithm uses a depth first branch and bound search. The search branches on bids and a constraint directed scheduling procedure is used at each node to verify the feasibility of the temporary schedule. The performance of the proposed algorithm is evaluated through experiments versus a set of auction-based scheduling winner determination problems generated based on a suite of job shop constraint satisfaction benchmark problems from the literature. Experimental results show that the proposed algorithm is on average more than an order of magnitude faster than the commercial optimization package CPLEX 10.0 over the tested problem sets.

Index Terms— Auction-based scheduling, winner determination, constraint directed search

1. Introduction

The Internet has presented tremendous potential in changing the way companies buy and sell goods, integrate their supply chains, and collaborate with their business partners. eMarkets are now emerging as a means of achieving greater efficiency in today's globalized economic environment. Among various protocols governing the negotiation between eMarket participants, auctions are a particularly successful multi-lateral negotiation protocol and account for an enormous volume of transactions on the Internet [1]. In many auctions in eMarkets, the goods to be sold are processing times of resources,

e.g. landing timeslots of airport runways [2], machine processing times of a factory [3], computation and network accessing times of internet resources [4], and the right to use railroad tracks [5], [6]. In these settings, bidders have jobs to be completed during specific time windows and they compete with each other for the resources to schedule their jobs according to respective objectives, and auctions are used as a mechanism to schedule resource times to bidders. We may refer to this type of auction designed for scheduling environments as auction-based scheduling.

In auction-based scheduling, bidders' valuations often exhibit complementarities. For example, due to scheduling constraints, a bidder may need to obtain a set of specific combinations of time periods on resources to process its jobs. The complementarities of bidders' valuations present particular challenges for the design of bidding languages in terms of expressiveness, conciseness, and naturalness. Logical languages, which use either logical combinations of items as formulae (L_G) [7] or logical combinations of bundles as formulae (L_B) [8, 9], have been proposed to address this challenge. However, these languages cannot be applied directly to scheduling problems because they are designed for auctioning discrete goods, not goods that exhibit continuity, such as processing time on resources in scheduling problems. In order to apply L_G or L_B , the scheduling timeline of resources needs to be discretized into fixed time units and these units are treated as distinct items sold in the auctions [3, 10]. With *timeline discretization*, bidders can express their time-related scheduling requirements, such as release times and due dates indirectly by attaching values to various time units combinations. However, this 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 these types of languages as requirement-based languages (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[6], Parkes and Ungar designed a requirement-based language, which allows train agents to specify the accessing and leaving time on a railroad track in their bids. Comparing with L_B , the L_R languages do not require *timeline discretization* and therefore, save the bidders' computation on the evaluation of the combinations of resource time units. However, as stated in Theorem 1 (Section 2.2), the winner determination problems formulated using L_R remain NP-hard in general.

This paper is concerned with the computational challenges of winner determination problems (WDP) formulated using L_R in the context of auction-based scheduling. As one cannot hope for a general purpose algorithm that can efficiently solve every instance of the WDP [11], we developed a scheduling domain specific winner determination algorithm using a constraint-based scheduling approach for the node feasibility validation in a bid-driven branch & bound search procedure. We did not discuss the concern that bidders might not report their true valuations because of strategic considerations. The

WDP is assumed to be inherited from some mechanisms, such as Vickrey-Clarke-Groves (VCG) mechanism [12] or an iterative implementation of VCG.

The rest of the paper is organized as follows: Section 2 formulates the WDP in auction-based scheduling using two type of bidding languages; L_B and L_R . In Sections 3 and 4, we propose a constraint-based winner determination algorithm for the L_R WDP formulation and compare our algorithm with CPLEX 10.0 on a suite of benchmark test problems. We further refine the algorithm in Section 5. Section 6 concludes the paper and points out future research directions.

2. WDP Formulations for Auction-Based Scheduling

The structure of the WDP in combinatorial auctions is affected by the bidding languages adopted. In this section, we formulate the WDP for auction-based scheduling using the general bidding languages L_B and the scheduling specific bidding languages L_R . As auction-based scheduling is a type of combinatorial auction (due to the complementarities of bidder's valuations), for the sake of completeness, we start by briefly introducing combinatorial auctions.

2.1. Preliminaries on combinatorial auctions

Auctions are a mediated negotiation process that allocates goods to bidders according to predefined rules based on bidders' bids. Combinatorial auctions are those auctions in which bidders can place bids on combinations of items, called "bundles," rather than just individual items. The advantage of combinatorial auctions is that bidders can more fully express their preferences. This is particularly important when items are complements,

that is, a set of items has greater value than the sum of the values for the individual items. For example, a pair of shoes is worth more than the value of a left shoe alone plus the value of a right shoe alone. Allowing bidders more fully to express preferences often leads to improved economic efficiency (allocating the items to those who value them most) and greater auction revenues.

However, alongside their advantages, combinatorial auctions raise a host of questions and challenges. An important one is the high computation demanded for winner determination. Given a set of bids in a combinatorial auction, the winner determination problem (WDP) is to find an allocation of items to bidders (the auctioneer can keep some of the items) that maximizes the auctioneer's revenue, and no items are assigned to more than one bidder. The problem is inherently complex. Specifically, it is NP-hard [13], meaning that a polynomial-time algorithm that is guaranteed to compute the optimal allocation is unlikely to exist. Even worse, the problem is not uniformly approximable, in the following sense: almost certainly there does not exist a polynomial-time algorithm and a constant d that, for all inputs, the algorithm produces an answer that is at least $1/d$ of the correct optimal answer [11].

Different bidding languages adopted by a combinatorial auction may result in fundamentally different WDP formulations. General bidding languages proposed in the literature use either logical combinations of items as formulae (L_G) [7, 14] or logical combinations of bundles as formulae (L_B) [8, 9]. L_G languages allow bids that are logical formulae where items are taken as atomic propositions and combined using logical connectives. L_B languages use bundles of items with associated prices as atomic propositions and combines them using logical connectives. In auction-based scheduling,

the outcomes of an auction are schedules, which allocate time units of resources to bidders' jobs. For a specific schedule, a bidder attaches his/her value to the bundle of all time units (not the individual time unit) assigned in a schedule. Therefore, in auction-based scheduling, L_B is more natural than L_G in terms of capturing bidders' utility functions. We present the WDP formulation for auction-based scheduling using L_B in the following section.

2.2. The WDP Formulation Using L_B

To illustrate the idea of formulating auction-based scheduling using L_B , we first present an example, which is a slight modification (release times of jobs are added) of the factory scheduling economy example adopted in [3]. Consider an unscheduled day shift of a factory (as shown in Fig. 1), which is divided into eight one-hour time units, labeled 9:00 to 16:00 according to their respective end times. In this example, one hour is the basic time unit, which cannot be further divided. Units are treated as distinct items that can be allocated for the production of customer orders. Assume each customer has one single-operation job to be completed. A customer's job is defined by its duration (length), its release time (the time when the job is available for processing), its deadline, and the valuation (expressed in dollars) the customer places on the job. To complete his/her job, the customer must acquire a number of time units no less than the length (not necessarily contiguous¹), within its feasible time window (the time period between its release time and its deadline).

For this example, we may construct an auction in which a customer can bid any combination of time units (with the sum of units equal or greater than the length of its

¹ In this example, we allow preemption at operation level.

job) within his/her feasible time window. For example, customer 1 can submit 4 XOR bids which are $\{9, 10, 11\}$, $\{10, 11, 12\}$, $\{9, 11, 12\}$ and $\{9, 10, 12\}$. In this case, the WDP is to allocate time units to customers such that no unit is allocated to more than one customer, no customer receives more than one bid and the revenue (sum of winning customers' prices) is maximized.

In the above example, we have treated each time-stamped unit as a distinct item to be sold in the auction. By doing this, we convert an auction-based scheduling problem to a *combinatorial auction problem* [14]. In the following, we present the standard WDP formulation for *combinatorial auction problems* and analyze its complexity in the context of auction-based scheduling.

Let Ω be the set of time units of resources (goods to be sold). For every subset B of Ω , let $price_j(B)$ be the price that the bidder j ($j = 1, \dots, n$) is willing to pay for B . Let $x_j(B) = 1$ if the bundle $B \subseteq \Omega$ is allocated to bidder j and zero otherwise. The problem can be formulated as an integer programming.

$$\max \sum_{B \subseteq \Omega} \sum_{j=1}^n x_j(B) price_j(B) \quad (1)$$

$$\text{s.t. } \sum_{B \subseteq \Omega} x_j(B) \leq 1, \quad j = 1, \dots, n \quad (2)$$

$$\sum_{B \subseteq \Omega, i \in B} \sum_{j=1}^n x_j(B) \leq 1, \quad \forall i \in \Omega \quad (3)$$

$$x_j(B) = \{0, 1\}, \quad B \subseteq \Omega, j = 1, \dots, n \quad (4)$$

In auction-based scheduling the number of feasible bids that a bidder can submit is restricted by the scheduling constraints, such as release time, deadline, and number of time units that a bidder needs to process its job. For example, in Figure 1, bidder 1 will

not bid on the time units after his/her deadline (12:00) because any time unit after 12:00 has no value to him/her. If a job cannot be interrupted during processing, we can further restrict the feasible bids by requiring that a bidder can bid only on time units that are adjacent. However, even with the scheduling constraints, as shown in the following theorem, the WDP remains NP-hard.

Theorem 1: *The WDP remains NP-hard even if we restrict to instances where the items to be sold are a set of linearly ordered time slots of a single resource; a bid has only one operation to be scheduled on the resource between its release time, r_j and deadline, d_j ; pre-emption is not allowed; all operations have identical processing time τ ; and $d_j - r_j - \tau \leq 2$.*

Proof: To show that the WDP with restrictions is NP-hard, we prove its decision version is NP-complete by constructing a polynomially computable reduction from the following variation of the job interval selection problem (JISP), which is NP-complete [15]. Given a set of $J = \{J_1, J_2, \dots, J_n\}$ jobs in a JISP, each of which requires the same execution time τ , and J_j may be started for execution at any of a set of given discrete starting times $S_j = \{s_j, \dots, s_j + k_j\}$ where $k_j \leq 2$, $1 \leq j \leq n$, we can construct an instance of the WDP as follows. The bidder j has job J_j to be executed with the release time $r_j = s_j$ and deadline $d_j = s_j + k_j + \tau$. v_j is the price that the bidder j is willing to pay for any bundles that satisfy its constraints. Because $k_j \leq 2$, $d_j - r_j - \tau \leq 2$ is satisfied. Let $V = \sum_{j=1}^n v_j$. To answer the question if the WDP has a solution with revenue V is equivalent to solving the NP-complete IJSP ■. This sentence doesn't read clearly.

The *time discretization* approach allows the WDP in auction-based scheduling to be

formulated using the general bidding language L_B . However, this language normally requires a large number of bids to represent a bidder's preferences in auction-based scheduling. As mentioned in the Introduction, a one week time window on 10 resources can be discretized into more than 100,000 time units to be sold, if the time accuracy we need is in minutes. In combinatorial auctions, for a bidder, the number of non-empty bids to be submitted is $2^m - 1$, where m is the number of items to be sold. To fully express his/her preferences over the combinations of the 100,000 time units using L_B , a bidder will have $2^{10^5} - 1$ bids to be submitted in this particular case. According to Theorem 1, the WDP remains NP-hard with scheduling domain restrictions. A large number of bids leads to larger problem size, which demands exponentially growing computation time.

To deal with this problem, we introduce a scheduling specific family of bidding languages, called requirement-based languages (L_R), in which the atomic propositions attach prices to the performance requirements of processing a job rather than items (as in L_G) or a bundle of items (as in L_B). The requirement can be the completion time of the job or other scheduling objectives.

2.3. A Requirement-Based Bidding Language

A good bidding language for a problem domain needs to capture the intrinsic structure of the problem, such that bidders' valuations can be expressed naturally and concisely using the language. In auction-based scheduling, bidders derive values based on the levels at which their performance requirements have been satisfied. Based on this valuation structure, we present a type of requirement-based bidding language in which atomic propositions attach prices to the completion time requirements of processing jobs.

We call the language completion time-based L_R .

An atomic proposition of the completion time-based L_R consists of a set of jobs to be scheduled, the required completion time before which the jobs need to be completed and the price that the bidder is willing to pay, given the completion time is satisfied. We refer to this atomic proposition as CBid (Completion time-based bid).

CBid is a 4-tuple $\langle J, C_{\max}, lft, p \rangle$ where J is a set containing the descriptions of the jobs and constraints; $C_{\max}$ defines that the measure of performance being used is makespan (completion time of all jobs in J); lft is the level of completion time required and p is the price that the bidder is willing to pay for $C_{\max} \leq lft$. CBids can also be connected by logical connectives. For example, if a bidder is willing to pay \$100 for the guarantee of completing its jobs before 4:00PM or \$60 for completing its job before 6:00PM, the bidder can express this valuation by submitting an XOR-CBid, $\langle J, C_{\max}, 4:00PM, \$100 \rangle XOR \langle J, C_{\max}, 6:00PM, \$60 \rangle$.

2.4. The WDP Formulation Using L_R

In the L_R model, the WDP involves the selection of a subset of CBids such that the sum of the prices of the selected bids is maximized and all scheduling constraints are satisfied. L_R WDP is different from the job shop scheduling problem model studied in classical scheduling theory. It is assumed in the job shop model that all jobs need to be included in the final schedule. L_R WDP expands the job shop model by adding the winner determination part to it. In L_R WDP, not only jobs need to be scheduled, the set of winning bids needs to be determined at the same time. We formulate the L_R WDP by

incorporating scheduling specific modeling techniques into the general *combinatorial auction problem* structure.

We consider a class of the WDPs for auction-based scheduling, which consist of a set of n bidders. Each bidder j ($j=1,...,n$) requires the processing of a job, which consists of a sequence of operations $o_{j,k}$ ($k=1,...,n_j$). An operation $o_{j,k}$ ($1 \leq j \leq n, 1 \leq k \leq n_j$) has a specified processing time, $p_{j,k} \in R^+$, and its execution requires the exclusive use of a designated resource for the duration of its processing. There are precedence constraints among operations of a job, that is, $o_{j,k}$ must precede $o_{j,k+1}$. If $o_{j,k}$ and $o_{\hat{j},\hat{k}}$ need to be processed on the same resource $q_{j,k,\hat{j},\hat{k}}=1$, otherwise $q_{j,k,\hat{j},\hat{k}}=0$. A bidder j has a job release time r_j after which the job is available for processing. The performance requirement of processing the job of bidder j is defined by the job's completion time d_j , by which the job needs to be finished. v_j is the price that bidder j is willing to pay for his/her job to be completed before d_j . In this case, a bid submitted to the auction can be represented using a CBid: $\langle J, C_{\max}, d_j, v_j \rangle$. The WDP involves the selection of a subset of bids such that the scheduling constraints for all selected bids are satisfied and at the same time, the sum of bid prices is maximized. Using the following variables:

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

$$Z_j = \begin{cases} 1 & \text{if bidder } j \text{ wins} \\ 0 & \text{otherwise.} \end{cases}$$

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

the WDP can be formulated as follows.

$$\max \sum_{j=1}^n Z_j v_j \quad (5)$$

Subject to

$$S_{j,1} \geq r_j Z_j \quad j = 1, \dots, n \quad (6)$$

$$S_{j,n_j} + p_{j,n_j} \leq d_j \quad j = 1, \dots, n \quad (7)$$

$$S_{j,k-1} + p_{j,k-1} \leq S_{j,k} \quad j = 1, \dots, n \quad 1 < k \leq n_j \quad (8)$$

$$\begin{aligned} S_{j,k} + p_{j,k} - S_{\hat{j},\hat{k}} + Hq_{j,k,\hat{j},\hat{k}} + \\ HZ_j + HZ_{\hat{j}} + HY_{j,k,\hat{j},\hat{k}} \leq 4H, \end{aligned} \quad (9)$$

$$j = 1, \dots, n, \hat{j} = 1, \dots, n, j \neq \hat{j}, 1 \leq k \leq n_j, 1 \leq \hat{k} \leq n_{\hat{j}}$$

$$\begin{aligned} Y_{j,k,\hat{j},\hat{k}} + Y_{\hat{j},\hat{k},j,k} + 3H \geq 1 + HZ_j + HZ_{\hat{j}} + Hq_{j,k,\hat{j},\hat{k}}, \\ j = 1, \dots, n, \hat{j} = 1, \dots, n, j \neq \hat{j}, 1 \leq k \leq n_j, 1 \leq \hat{k} \leq n_{\hat{j}} \end{aligned} \quad (10)$$

$$\begin{aligned} Y_{j,k,\hat{j},\hat{k}} + Y_{\hat{j},\hat{k},j,k} + HZ_j + HZ_{\hat{j}} + Hq_{j,k,\hat{j},\hat{k}} \leq 1 + 3H, \\ j = 1, \dots, n, \hat{j} = 1, \dots, n, j \neq \hat{j}, 1 \leq k \leq n_j, 1 \leq \hat{k} \leq n_{\hat{j}} \end{aligned} \quad (11)$$

$$\begin{aligned} Y_{j,k,\hat{j},\hat{k}} \in \{0,1\}, \\ j = 1, \dots, n, \hat{j} = 1, \dots, n, j \neq \hat{j}, 1 \leq k \leq n_j, 1 \leq \hat{k} \leq n_{\hat{j}} \end{aligned} \quad (12)$$

$$Z_j \in \{0,1\}, \quad j = 1, \dots, n \quad (13)$$

$$S_{j,k} \geq 0, \quad j = 1, \dots, n, 1 \leq k \leq n_j \quad (14)$$

The set of constraints (6) and (7) ensures that the operations of a job do not start before its release time and finish after its completion time. The set of constraints (8) ensure that an operation does not start before the previous operation of the same job is completed. The set of constraints (9) is a set of logical constraints saying:

if two bidders j and $\hat{j}$ are selected in the schedule, and operations $S_{j,k}$ and $S_{\hat{j},\hat{k}}$ are to

be processed on the same resource ($q_{j,k,\hat{j},\hat{k}}=1$), and $S_{j,k}$ precedes $S_{\hat{j},\hat{k}}$ ($Y_{j,k,\hat{j},\hat{k}}=1$), then $S_{j,k} + p_{j,k} \leq S_{\hat{j},\hat{k}}$. These constraints ensure that, at most, one operation can be processed by a particular resource at a time, where H is a large positive constant, which is used for the linearization of the logical constraint “if”. The minimum value of H depends on the problem instance. In general, a $H = \max(d_j) + \max(p_{j,k})$, where $j = 1, \dots, n$ and $k = 1, \dots, n_j$, is large enough to enforce the logical “if” constraint. Constraints (10) and (11) ensure the values assigned to the two related variables $Y_{j,k,\hat{j},\hat{k}}$ and $Y_{\hat{j},\hat{k},j,k}$ are consistent, that is, if $S_{j,k}$ and $S_{\hat{j},\hat{k}}$ are to be processed on the same resource, then $Y_{j,k,\hat{j},\hat{k}} + Y_{\hat{j},\hat{k},j,k} = 1$. Constraints (12), (13), (14) are non-negative and integer constraints.

In this L_R WDP formulation, the number of bids in a WDP is the number of jobs to be processed, which is usually much smaller than $2^m - 1$ where m is the number of time units to be sold. Therefore, in terms of the number of bids in the problem, the size of the WDP formulated using L_R can be much smaller than that formulated using L_B . In the following section, we develop a constraint-based branch and bound algorithm for optimal winner determination for the WDPs formulated using L_R .

3. Constraint-Based Winner Determination

For optimal winner determination, branch and bound algorithms are often used. A branch and bound algorithm for the WDP are modeled either to branch on items [16, 17] or to branch on bids [18]. For the WDPs formulated by L_R , we propose a Branch and Bound Search (BBS) algorithm which branches on bids.

3.1. Branching structure of BBS

The algorithm is a depth-first tree search. Fig. 2 depicts a 3 bids search tree of BBS. The search starts with an empty *temporal schedule*, $TEMP$. Along the path $TEMP$ is expended by adding more bids from AV , which is a set that contains no 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 in terms of the value of the objective function of the WDP. Along the search path, the algorithm needs to check the feasibility of $TEMP$ after a new node is added. If the current $TEMP$ is not feasible, it is not necessary to search further along the path, and the algorithm backtracks. For the L_B formulation, as long as any two winning bids in $TEMP$ do not share an item, the $TEMP$ is feasible. However, in L_R formulation, validating the feasibility of $TEMP$ is equal to answering the question: given a set of bids in $TEMP$ with constraints, does a schedule exist that allocates the operations of bids on the resources, such that all constraints are satisfied? This decision problem is a job shop Constraint Satisfaction Problem (CSP) [19], which is known to be NP-complete [20]. We will propose a constraint-directed search algorithm for L_R feasibility checking later in this section. The branch and bound search is invoked by calling **BRANCH-BOUND-SCHEDULING** ($bids$). The algorithm is described as following.

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  $AV = \emptyset$  then return

     $h \leftarrow \sum_{bid \in AV} bidding - price - of(bid)$ 

    If  $sum + h \leq sum^*$  , then return

     $bid \leftarrow \text{SELECT-UNASSIGNED-BID} ( AV )$ 
     $TEMP = TEMP \cup \{bid\}$  ,  $AV = AV - \{bid\}$ 
     $sum \leftarrow sum + bidding - price - of(bid)$ 

    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

```

where CHECK-FEASIBILITY ($TEMP$) is the sub-algorithm for the feasibility checking (which will be described in the following subsection) and SELECT-UNASSIGNED-BID (AV) is the sub-algorithm that implements the bid ordering heuristics (which will be described in subsection 3.3).

3.2. Constraint-Based Feasibility Validation

We describe the constraint-based feasibility validation algorithm based on a disjunctive graph representation of the job shop CSP. Consider a directed graph G with a set of nodes N and two sets of arcs A and B . A node corresponds to an operation $o_{j,k}$ and its set of eligible resources $e(o_{j,k})$, denoted by $(e(o_{j,k}), o_{j,k})$. $e(o_{j,k})$ is a set of resources which has the capability of processing $o_{j,k}$. In job shop CSP, $|e(o_{j,k})| = 1$, that is, only one resource is capable of processing a specific operation. We can have a one to one mapping from

$e(o_{j,k})$ to a specific resource. For example, in Fig. 3, $e(o_{1,1}) = R_1$. The conjunctive arcs (solid arcs in Fig. 3) A represent the precedence constraints of two operations belonging to the same job. If an arc from node $(e(o_{j,k}), o_{j,k})$ to node $(e(o_{j,\hat{k}}), o_{j,\hat{k}})$ is part of A , then $o_{j,k}$ has to be processed on $e(o_{j,k})$ before $o_{j,\hat{k}}$ is processed. $e(o_{j,\hat{k}})$. Two operations that belong to two different jobs and that have to be processed on the same resource are connected by two disjunctive arcs (dashed arcs in Fig. 3) that go in opposite directions. The disjunctive arcs B form m cliques of double arcs, one clique for each resource. All operations in the same clique have to be done on the same resource. All arcs emanating from a node have as length the processing time of the operation that is represented by that node. In addition, there is a source U (representing time zero) and a sink V (representing an upper bound T of the minimal makespan of the problem), which are dummy nodes. The first operation of each job is connected to U by a conjunctive arc emanating from U with as length the release time of that job. The last operation of each job is connected to V by a conjunctive arc emanating to V with as length of $p_{j,n_j} + T - d_j$. This graph is denoted by $G = (N, A, B)$. A feasible schedule of job shop CSP corresponds to a selection of one disjunctive arc from each pair such that the resulting directed graph $\bar{G} = (N, A, \bar{B})$ is acyclic and the longest path in $\bar{G}$ from the source U to the sink V is not longer than T , where $\bar{B}$ denotes the subset of the selected disjunctive arcs. Fig. 2 shows an example of the disjunctive graph representation of a small job shop CSP.

If we model each disjunctive arc pair in B as a variable and the two opposite directions as two possible values, this problem can be solved by a constraint directed backtrack search procedure. The solution is incrementally extended through the repeated binding of

a direction to unconstrained disjunctive arc pairs. 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 [21], *Precedence Constraint Posting* (a commitment heuristic, developed in [19], and a chronological backtracking. The validation function is invoked by calling CHECK-FEASIBILITY (*bids*).

Algorithm 2

function CHECK-FEASIBILITY (*bids*) **returns** *pass* or *failure*

$N \leftarrow \{\text{operations in bids}\}$

$A \leftarrow \{\text{conjunctive arcs in bids}\}$

$B \leftarrow \{\text{disjunctive arcs in bids}\}$

$G = (N, A, B)$

return BACKTRACKING (G)

function BACKTRACKING (G ,) **returns** *pass* or *failure*

if inconsistent assignments detected in G

then return failure.

if $\{\text{uncommitted pairs in } B\} = \emptyset$ **then return** *pass*

*/*Constraint propagating*/*

CONSTRAIN-BASED-ANALYSIS (G)

if $\{\text{uncommitted pairs in } B\} = \emptyset$ **then return** *pass*

*/*Variable ordering*/*

$\text{pair} \leftarrow \text{PRECEDENCE-CONSTRAINT-POSTING} (\{\text{uncommitted pairs in } B\})$

*/*Value ordering*/*

direction-sequence =PRECEDENCE-CONSTRAINT-POSTING (*pair*)

for each *direction* **in** *direction-sequence* **do**

commit *direction* to *pair*

result $\leftarrow$ BACKTRACKING (*G*)

if *result* $\neq$ *failure* **then return** *pass*

de-commit *direction* to *pair*

return *failure*

3.3. Bid ordering heuristics

The performance of BBS can be improved by incorporating bid ordering heuristics in the function SELECT-UNASSIGNED-BID (AV). We give bids that contribute more to the revenue (have higher bidding prices), and higher flexibility (with less constraints) more weight. It is clear that the price of bids is a measure of contribution to the revenue. We use the *flexibility* of a bid to measure the possibility that it leaves to other bids to contribute. For a bid j , the *flexibility* of j is denoted by $flex_j = slack_j / (d_j - r_j)$ where

$slack_j = d_j - r_j - \sum_{k=1}^{n_j} p_j$. By its definition, $slack_j$ represents the free space in bid j 's time

window $(d_j - r_j)$. It enables the operations of bid j 's to be feasibly shuffled around in that

window. A bid with high *flexibility* can adjust the processing of its operations to

accommodate other bids in the schedule without violating its constraints. At a search

node, a bid that maximizes $v_j \exp(flex_j / Q)$ is added to *TEMP*. Q is a scaling parameter that

can be determined empirically. Intuitively, a large Q gives too much preference to bids

with high prices. However, a smaller Q gives more preference to bids with high

flexibilities.

4. A Computational Study

This section validates the effectiveness of the constraint-based winner determination algorithm through a computational study. The proposed algorithm is compared with CPLEX 10.0 (<http://www.ilog.com/products/cplex/>) in terms of the running time for computing optimal solutions to a set of benchmark testing problems.

4.1. Design of the Testing Data

There are common combinatorial auction benchmarks distributions, such as those presented in [17, 22] developed for L_B formulation. We design our testing problems for L_R formulation based on a suite of job shop CSP benchmark problems developed in [23]. Two parameters were adjusted to cover different scheduling conditions in [23]. The first one is a range parameter, RG , that controls the distribution of job deadlines and release times. The second is a bottleneck parameter, BK , that controls the number of major bottleneck resources. In our decentralized scheduling test problems, we have introduced a third parameter, P , to control the distribution of the prices of bids. Deadlines are randomly drawn from a uniform distribution $M * U(1 - RG, 1)$, where $U(a, b)$ represents a uniform probability distribution between a and b , and M is an estimate of the minimum makespan of the problem, which is determined by the average duration of all operations and the average duration of the operations requiring bottleneck resources. This estimate was first suggested in [24]. Similarly, release times are randomly drawn from a uniform distribution of the form: $M * U(0, RG)$. The price of bid j is randomly drawn from a uniform distribution on $U(Pdu_j, du + Pdu_j)$, where du is the average duration of all bids, and

du_j is the duration of bid j . In the original job shop CSP benchmark design, M is inflated to $(1+S)M$, where S is called slack parameter which is a function of RG and BK . The rationale of using S is to make sure that most of the problems generated remain feasible (including all jobs). In our test problem design, we set $S=0$ because if all bids can be included in an optimal solution, the winner determination problem becomes trivial for the BBS. By considering the different values of the parameters, seven groups of 60 problems of different sizes (number of bids) were randomly generated. The details are summarized in Table 1.

4.2. **Experimental Results**

This section reports the performance of BBS over the decentralized scheduling testing problem set against CPLEX10.0. It was reported in [25] that as a general-purpose integer programming package, CPLEX 6.5 performs very well in terms of running time for many of the common benchmarks distributions (comparable to the special-purpose winner determination algorithms, such as those in [16, 17]). A detailed comparison between CPLEX8.0 and a recently developed sophisticated algorithm CABOB can be found in [18].

We tested BBS and CPLEX 10.0 on the first six groups of the test problems. There are 60 problem instances in a group. These instances are divided into 6 sub-groups according to the number of bids in an instance. The number of bids can be seen as a measure of the size of an instance. Each sub-group has 10 instances of the same size. The instance size scales from 5 to 10 in a group. We have imposed a 1000 second time limit for both algorithms. All the instances under size 8 can be solved by both algorithms within the time limit. When the size scales to 9 and 10, some of them cannot be solved within the

time limit. For a sub-group, if the number of the instances solved within the time limit is less than 8, we do not consider the sub-group in our comparison.

Fig. 4 shows the running times of BBS and CPLEX10.0 over the six groups of the test problems. BBS is around an order of magnitude faster (on average) than CPLEX10.0. However, there are exceptions. For the instances of size 9 and 10 in groups #3 and #5, CPLEX10.0 is close and some times faster than BBS. On the other hand, for the instances of all sizes in group #4, BBS is more than one order (around two orders in some cases) of magnitude faster than CPLEX10.0. Fig. 5 depicts the performance of BBS and CPLEX10.0 over three problem groups. Instances in these groups were generated with different values of RG . Values for other parameters were kept the same among groups. It is observed that BBS and CPLEX10.0 have similar trends in Fig. 5. That is, instances with tight constraints (bigger RG) tend to be solved quicker.

5. Constraint-based Pruning Scheme

While BBS performed quite well over the tested problem set, it does not in fact utilize all of the information provided by the L_R formulation. As shown in Fig. 6, the solutions of a L_R formulation with n ($n=4$ in Fig. 6) bids, can be organized using an n -level structure. A solution at level k ($1 \leq k \leq n$), contains exactly k bids and the number of solutions at level k is $n!/(n-k)!k!$. BBS adopts a top down approach (same as CABOB in [18]). In order to find a good solution quickly, it starts with larger size solutions (from right to left in the search tree shown in Fig. 3) trying to include as many bids as possible in the beginning. If a solution that includes all bids happens to be feasible, the search finds the optimal solution without a backtrack. However, if a solution that includes all or

a high number of bids is not feasible, before backtracking, BBS has to first exhaustively determine its infeasibility. This may lead to bad *any time performance* for BBS. We have observed in our experiments that BBS goes slowly in the early stages of a search.

In addition, BBS cannot effectively utilize the feasibility checking results of previously traversed nodes to prune search space. For example, suppose that B_2 and B_3 need to use a bottleneck resource at the same time and this conflict cannot be resolved according to their constraints. This implies, B_2 and B_3 cannot coexist in a feasible solution. Therefore, no feasible schedule can include B_2B_3 (such as $B_1B_2B_3$ and $B_2B_3B_4$). In light of this observation, we have developed an improved constraint-based pruning scheme to further prune the search space using a simple constraint propagator: any solution that includes an infeasible solution is also infeasible. The algorithm starts with smaller size solutions and tries to identify conflicts at an early stage and use the conflicts information to prune the search space. However this pruning scheme does not always lead to shorter computation time as for many problems there are no hard conflicts between bids. Even if there are some, they may have been pruned by the upper bounding scheme of BBS before they are identified. Preliminary experiments show that for some moderate size problems that usually take BBS 40-80 seconds, if a certain percentage (above 40%) of job shop CSPs is pruned by the scheme during search, usually a similar percentage of computation time can be saved. If the percentage is too low the constraint-based pruning scheme can cause longer computation time due to the cost associated with the propagating of infeasibility constraints. We leave further investigation of this pruning scheme to future research.

6. Related Work

Limited attention has been devoted to auctions for decentralized scheduling. Wellman et al. [3] investigated auction protocols for decentralized scheduling. They imposed a discretization of time into finite slots, while our approach allows bidders to use a requirement-based expressive bidding language to bid for the processing of a set of jobs under certain constraints. In [6], Parkes and Ungar presented an auction-based method for decentralized train scheduling. The bidding language designed in the train scheduling auction avoids the use of discrete time slots. Bids were expressed by specifying a price to enter a track line and a time window. The winner determination problem was formulated with mixed integer programming, with many domain-specific constraints, and solved with CPLEX. No domain-specific winner determination algorithm was proposed in this work. In [26] auctions over tasks with complex time constraints and interdependencies were proposed. The problem was not to schedule tasks the bidder has, but to produce a schedule of tasks that other bidders would do. The objective was to optimize the expected customer's utility before bids are submitted and schedules are finalized. In [10], combinatorial auctions were applied to the job shop scheduling problem. The focus of this work was to investigate the links between combinatorial auctions and Lagrangean relaxation, and to design auctions based on the Lagrangean decomposition. Another Lagrangean decomposition-based solution for web banner advertisements scheduling can be found in [27]. A "schedule selection game" was presented in [28] for collaborative production scheduling. The emphasis of this work was on the incentive compatibility of the mechanism rather than winner determination. While our approach is designed for optimal winner determination, there are distributed scheduling approaches which tackle

the combinatorial complexities of scheduling problems in a heuristic manner[29, 30]. In these approaches, interaction protocols among agents are usually clearly defined. However, the optimality of the schedules found through the interaction is not guaranteed. Many agent-based manufacturing scheduling approaches fall into this category. For comprehensive surveys in this area, the reader may refer to [31, 32]. Auction-based scheduling approaches were also applied to wireless systems for channel scheduling [33, 34]. Due to the high responsiveness requirement in wireless systems, these approaches deal with the combinatorial complexities of the WDP problem by either conducting single item auctions repeatedly or utilizing fast approximate algorithms. Recently proposed winner determination algorithms can be found in [16-18]. These algorithms target the WDP formulated using L_B and cannot be directly applied to our L_R based winner determination problem formulation in auction-based scheduling.

7. Conclusions

The objective of this research is to investigate how the domain specific properties of scheduling problems can facilitate efficient winner determination algorithms. We have presented a formulation and an algorithm for the winner determination problem of auction-based decentralized scheduling. We use a constraint-directed scheduling algorithm to verify the feasibility of the allocation at each node of the branch and bound search. We have also proposed a constraint-based pruning scheme which uses the domain-specific heuristics to further prune the search space. This work focuses on optimal winner determination. We have restricted the size of the testing problems to a level, such that they can be solved by both CPLEX and the constraint-based winner

determination algorithm under a reasonable time limit. For large scale WDPs in auction-based scheduling, to which optimal algorithms cannot be practically applied, we will develop constraint-based approximate algorithms in future research. This work has also focused only on the L_R WDP formulation. In fact, we have applied CPLEX to the L_B formulation of the same testing problem sets. We discovered that the L_R formulation was fundamentally superior to the L_B formulation in terms of solving speed and scalability. The detailed comparison data will be reported separately. The current L_R formulation restricts bids to one dimension, which is price only. We plan to extend this work to multi-attribute WDPs in the near future.

8. References

- [1] M. Bichler, "A roadmap to auction-based negotiation protocols for electronic commerce," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 2000. , 2000, pp. 1-10.
- [2] S. J. Rassenti, V. L. Smith, and R. L. Bulfin, "A Combinatorial Auction Mechanism for Airport Time Slot Allocation," *The Bell Journal of Economics*, vol. 13, pp. 402-417, 1982.
- [3] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason, "Auction Protocols for Decentralized Scheduling," *Games and Economic Behavior*, vol. 35, pp. 271-303, 2001.
- [4] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, pp. 1507-1542, 2002.
- [5] P. J. Brewer and C. R. Plott, "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.
- [6] D. C. Parkes and L. H. Ungar, "An auction-based method for decentralized train scheduling," *Proceedings of the fifth international conference on Autonomous agents*, pp. 43-50, 2001.
- [7] C. Boutilier and H. H. Hoos, "Bidding languages for combinatorial auctions," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001, pp. 1211-1217.
- [8] N. Nisan, "Bidding languages for combinatorial auctions " in *Combinatorial Auctions*, Cramton, Shoham, and Steinberg, Eds.: MIT Press, 2006.

- [9] T. Sandholm, "eMediator: A Next Generation Electronic Commerce Server " *Computational Intelligence*, vol. 18, pp. 656-676, 2002.
- [10] E. Kutanoglu and S. D. Wu, "On combinatorial auction and Lagrangean relaxation for distributed resource scheduling," *IIE Transactions*, vol. 31, pp. 813-826, 1999.
- [11] D. Lehmann, R. Muller, and T. Sandholm, "The Winner Determination Problem," in *Combinatorial Auctions*, P. Cramton, Y. Shoham, and R. Steinberg, Eds.: MIT Press, 2006.
- [12] P. C. Cramton, Y. Shoham, and R. Steinberg, *Combinatorial auctions*: MIT Press, 2006.
- [13] M. H. Rothkopf, A. Peke, and R. M. Harstad, "Computationally Manageable Combinational Auctions," *Management Science*, vol. 44, pp. 1131-1147, 1998.
- [14] H. H. Hoos and C. Boutilier, "Solving combinatorial auctions using stochastic local search," *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 22-29, 2000.
- [15] J. M. Keil, "On the complexity of scheduling tasks with discrete starting times " *Operations Research Letters*, vol. 12, pp. 293-295, 1992.
- [16] Y. Fujishima, K. Leyton-Brown, and Y. Shoham, "Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches," in *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1999, pp. 548-553.
- [17] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artificial Intelligence*, vol. 135, pp. 1-54, 2002.
- [18] T. Sandholm, S. Suri, A. Gilpin, and D. Levine, "CABOB: A Fast Optimal Algorithm for Winner Determination in Combinatorial Auctions," *Management Science*, vol. 51, pp. 374-390, 2005.
- [19] S. F. Smith and C. Cheng, " Slack-Based Heuristics for Constraint Satisfaction Scheduling," *Proceedings of 11th National Conference on Artificial Intelligence, Washington D.C.*, vol. July 1993.
- [20] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*: WH Freeman & Co. New York, NY, USA, 1979.
- [21] J. Erschler, F. Roubellat, and J. P. Vernhes, "Characterizing the set of feasible sequences for n jobs to be carried out on a single machine," *European Journal of Operational Research*, vol. 4, pp. 189-194, 1980.
- [22] K. Leyton-Brown, M. Pearson, and Y. Shoham, "Towards a universal test suite for combinatorial auction algorithms," in *The 2nd ACM conference on Electronic commerce*, 2000, pp. 66-76.
- [23] N. Sadeh and M. S. Fox, "Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem," *Artificial Intelligence*, vol. 86, pp. 1-41, 1996.
- [24] P. S. Ow, "Focused Scheduling in Proportionate Flowshops," *Management Science*, vol. 31, pp. 852-869, 1985.
- [25] A. Andersson, M. Tenhunen, and F. Ygge, "Integer programming for combinatorial auction winner determination," *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pp. 39-46, 2000.

- [26] A. Babanov, J. Collins, and M. Gini, "Scheduling tasks with precedence constraints to solicit desirable bid combinations," *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 345-352, 2003.
- [27] A. Amiri and S. Menon, "Scheduling Web Banner Advertisements With Multiple Display Frequencies," *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, vol. 36, pp. 245-251, 2006.
- [28] E. Kutanoglu and S. D. Wu, "Incentive compatible, collaborative production scheduling with simple communication among distributed agents," *International Journal of Production Research*, vol. 44, pp. 421-446, 2006.
- [29] B. P. C. Yen and O. Q. Wu, "Internet scheduling environment with market-driven agents," *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, vol. 34, pp. 281-289, 2004.
- [30] C. Wang, H. Ghenniwa, and W. Shen, "Distributed Scheduling for Reactive Maintenance of Complex Systems," *IEEE Transactions on System, Man, and Cybernetic, Part C*, in press, 2007.
- [31] W. Shen, "Distributed manufacturing scheduling using intelligent agents," *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol. 17, pp. 88-94, 2002.
- [32] W. Shen, L. Wang, and H. Qi, "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.
- [33] S. Jun, E. Modiano, and Z. Lizhong, "Wireless channel allocation using an auction algorithm," *Selected Areas in Communications, IEEE Journal on*, vol. 24, pp. 1085-1096, 2006.
- [34] S. Pal, S. Kundu, M. Chatterjee, and S. Das, "Combinatorial Reverse Auction based Scheduling in Multi-Rate Wireless Systems," *IEEE Transactions on Computers*, vol. 56, pp. 1329-1341, 2007.

Fig. 1 An Example of Decentralized Scheduling Problem

Fig. 2 A BBS search tree of 3 bids

Fig. 3 Disjunctive graph for a job shop CSP

Table 1 Configuration of the test problem set

Group	BK	RG	P	Bids	Instances
1	1	0.4	0	5-10	60
2	1	0.5	0	5-10	60
3	2	0.5	0	5-10	60
4	2	0.4	0	5-10	60
5	1	0.5	1	5-10	60
6	1	0.4	1	5-10	60
7	1	0.3	1	5-10	60

Fig. 4 Run times on 6 groups of test problems

Fig. 5 Run times of BBS and CPLEX on 3 groups of test problems with different RG values

Fig. 6 Improved pruning in BB