Decentralized Multi-Project Scheduling via Multi-Unit Combinatorial Auction

Wen Song, Donghun Kang, Jie Zhang[a], Hui Xi[b]
Rolls-Royce@NTU Corp Lab, Nanyang Technological University, Singapore
[a]School of Computer Engineering, Nanyang Technological University, Singapore
[b]Rolls-Royce Singapore Pte Ltd, Singapore
{songwen, donghun.kang, [a]zhangj}@ntu.edu.sg, [b]hui.xi@rolls-royce.com

ABSTRACT

In industry, many problems are considered as the Decentralized Resource-Constrained Multi-Project Scheduling Problem (DRCMPSP). Existing approaches encounter difficulties in dealing with large problems while preserving information privacy of project agents. In this paper, we propose a novel approach to solve DRCMPSP based on the multi-unit combinatorial auction, which can efficiently solve the problem without violating information privacy. It adopts a greedy resource allocation strategy with fixed resource cost to simplify computation required for project (bidder) and auctioneer agents. In addition, a bid modification step is incorporated to allow project agents to better utilize resources. Analysis and empirical results indicate that our approach outperforms state-of-the-art decentralized approaches in minimizing average project delay, and scales well to large problems with thousands of activities from tens of projects.

Keywords

Multi-Project Scheduling, Multi-Unit Combinatorial Auction, Resource Allocation

1. INTRODUCTION

Most business firms need to manage multiple projects simultaneously. Usually, these projects have temporal constraints on activities and capacity constraints on resources [23]. With the rapid growth of globalization and information technology, nowadays it is quite common for firms to conduct intra- and inter-firm collaborations to improve efficiency and reduce cost [21]. This trend brings the traditional multi-project scheduling into a decentralized environment, where projects are distributed and controlled by different self-interested decision makers. In order to achieve individual objectives, these decision makers usually need to compete for some shared global resources having limited capacities. Moreover, they may be reluctant to reveal private information on the projects, especially when they are rivals in the market. In this decentralized and information asymmetric environment, the traditional multi-project scheduling problem has evolved into the *Decentralized Resource-Constrained Multi-Project Scheduling Problem* (DRCMPSP) [4].

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

Many real-world problems in manufacturing and service operations with complex product/supply structures are considered as DRCMPSP. In [11], an intra-firm scenario is given where different product managers in an electronics manufacturing company must share and compete for common production resources (e.g. automated production lines) in a regular basis. To satisfy their own customers' requirements, these managers need to deal with uniquely different sets of activities and constraints (e.g. lead time pressure, processing requirements). In terms of inter-firm scenarios, a typical example can be found in the Aero Repair and Overhaul industry [20]. In this example, different fleet managers who are dedicated to manage the aero engines of an airline, compete for several overhaul bases heaving limited repair capacities with other fleet managers. These managers need to schedule the overhaul base visit time for each controlled engine, to maximize their operating revenue at the same time. Similar inter-firm cases can also be found in the airport ground handling service scheduling [15] and supply chain scheduling [12]. As stated in both [12] and [20], approaches that respect the privacy of self-interested decision makers are more preferable in decentralized scheduling, since these decision makers may as well be competitors in the same marketplace.

Due to its decentralized nature, DRCMPSP is usually modeled as a mediated multi-agent system where each project is represented by a Project Agent (PA) and all PAs are coordinated by a Mediator Agent (MA). The core problem in DRCMPSP is how to allocate the shared global resources to each decision maker in a decentralized manner. Market-based approach, especially combinatorial auction, is an ideal choice for decentralized scheduling since it well supports the decentralized decision making and information privacy of the decision makers [22]. However, as will be detailed in Section 2, current combinatorial auction based approaches either cannot fully satisfy the requirements of DRCMPSP (e.g. not fully decentralized, not allow multiple activities per project), or underperform both in solution quality and computational efficiency, especially on large problem cases with thousands of activities from tens of projects.

In this paper, we propose a novel decentralized approach for solving DRCMPSP based on the multi-unit combinatorial auction. We simplify the computation of the agents by adopting a greedy resource allocation strategy and fixed resource cost, and incorporate a bid modification step to provide possibilities for the project agents to better utilize global resources. Following previous research on DRCMPSP, we assume that the agents are willing to provide truthful information required by the approach, i.e. currently

we do not handle the incentive compatibility issue[1]. However, by limiting the exchanged information to bids and demand ratio (defined in Section 4), this approach does not require private project information from the project agents.

Our complexity analysis shows that this approach terminates within a maximum number of auctions that increases linearly along with the number of projects, regardless of the number of activities, and requires polynomial computation for all agents. Empirical results on cases with multi-unit resources (obtained from the public benchmark MPSPLIB[2]) and single-unit resource (generated from MPSPLIB) show that our approach outperforms state-of-the-art decentralized approaches in minimizing average project delay, one of the most widely adopted performance measure of DRCMPSP, within reasonable computational time.

2. RELATED WORK

In the literature, combinatorial auction has been applied to solve simple decentralized scheduling problems [11, 22, 4]. In [11], Kutanoglu and Wu introduce a combinatorial auction based distributed scheduling framework, and show that combinatorial auction with a certain payment function is corresponding to the lagrangean decomposition of the centralized problem. Nevertheless, their approach is not fully decentralized, since a centralized algorithm is needed to clear the market and generate a feasible solution. In [22], Wellman et al. give concrete theoretical results on the equilibrium and efficiency of applying combinatorial auction to schedule activities from different agents on a shared resource. However, their analysis is based on the model of a simple factory scheduling problem where each agent is only in charge of one activity. In large DRCMPSP cases, each project agent (PA) could be in charge of hundreds of activities with complex precedence relations. Confessore et al. propose another combinatorial auction based approach in [4]. This approach is based on an iterative combinatorial auction named *i*Bundle [17], which has been applied to solve problems such as multi-agent pathfinding [2] and train scheduling [18]. The nice properties of *i*Bundle, including optimality guarantee and strategy-proofness, are based on the assumption that all bidders can give Myopic Best-Response (MBR) in each round of auction. However, when applied to DRCMPSP, MBR is difficult to satisfy due to the intractability of local bidding problems. When approximated algorithms without optimality guarantee are used for bidding, as in [4], both optimality and strategy-proofness on the final solution will be compromised. In addition, when MBR cannot be satisfied, *i*Bundle cannot guarantee that all the bidders will be granted a bid upon termination. Finally, *i*Bundle could take a large number of iterations to terminate, which requires intense computation for bidders and the auctioneer, resulting in inefficiency on large-scale cases. Besides their own limitations, all approaches in [11, 22, 4] can only be used to schedule single-unit global resources. Moreover, test cases are rather small with tens of activities in total from several PAs. In contrast, our approach can efficiently handle multi-unit resources and large problem cases.

More recently, several approaches have been proposed to solve much larger DRCMPSP cases with hundreds to thousands of activities from tens of projects sharing several multi-unit global resources [15, 7, 1, 24]. In [15], Mao et al. propose a market-based approach to schedule the airport ground handling services, but the resource capacities are assumed to be infinite which is hardly found in practice. In [7], Homberger presents an evolutionary computation based negotiation approach, but it is outperformed by a centralized approach SASP [10], one of the best priority rule based multi-project scheduling algorithms in minimizing average project delay. In [1], Adhau et al. introduce an approach named DMAS/ABN, which conducts an auction-based negotiation on each time slot for each activity. In [24], Zheng et al. propose an approach named DMAS/EM which employs an activity elimination algorithm to fix an infeasible solution. Both DMAS/ABN and DMAS/EM outperform SASP in minimizing average project delay. One common feature of the approaches in [15, 7, 1, 24] is that, they are based on activity-level negotiation to generate or fix a solution, which has two major drawbacks. Firstly, activity information (e.g. start time, duration, resource requirements) is inevitably required by the mediator. Secondly, when the individual objective of each PA cannot be decomposed precisely to each activity (e.g. project makespan, delay cost), the decision on global resource allocation to each activity can only rely on the estimated objective value, which could result in unsatisfactory solution quality. On the contrary, global resource allocation in our approach is purely on project level, which can satisfy the private information requirement and provide more precise information for allocating global resources.

3. PROBLEM STATEMENT

In DRCMPSP, a number of projects that share global resources will be scheduled in a time horizon of T consecutive time slots. Each project P_i, $i \in \{1, ..., N\}$, consists of non-preemptive activities a_{ij}, $j \in \{1, ..., J_i\}$. Let s_{ij} and d_{ij} be the start time and duration of an activity a_{ij}, respectively. The precedence constraint $a_{ij} \prec a_{ik}$ indicates that an activity a_{ik} cannot start before the completion of its preceding activity a_{ij} (i.e. $s_{ik} \geq s_{ij} + d_{ij}$). Each project P_i has two time-related attributes, an earliest start date ed_i and an expected due date dd_i. For a project P_i, a Project Agent PA_i is assigned to control P_i. Each PA_i can receive a revenue rv_i upon the completion of P_i.

Each activity requires certain amounts of resources. A set of G global resources are shared by all the projects. Each global resource R_g, $g \in \{1, ..., G\}$ has a limited capacity C_{gt} in time t, $t \in \{1, ..., T\}$, and has a fixed utilization cost c_g per unit at each time slot. The global resources are managed by a Mediator Agent (MA). Also, each PA_i may own a set of L_i local resources[3] that are dedicated to P_i. Each local resource R_{l_i}, $l_i \in \{1, ..., L_i\}$, has a limited capacity $C_{l_i t}$ in time t. Each activity a_{ij} requires r^g_{ij} units of global resource R_g and $r^{l_i}_{ij}$ units of local resource R_{l_i} during each time slot of its processing period.

Let S be a solution of a DRCMPSP and S_i be a schedule of a project P_i. Assume that S_i is a vector of individual activity's start times, namely $S_i = (s_{i1}, ..., s_{iJ_i})$. Then, the solution S can be defined as a vector of individual project's

[1]Note that privacy and truthfulness are two separate issues. Even though an agent is willing to tell the truth, it may still not be willing to disclose their sensitive information [5].

[2]http://www.mpsplib.com/

[3]Here we ignore the cost of local resources since they are assumed to be already owned by the PA and do not need to be allocated to each PA. However, our approach can also work when local resource cost is considered.

schedules: $S = (S_1, ..., S_N)$. A feasible solution S cannot violate any precedence or global/local resource constraints, but the project due dates are soft constraints that can be violated. However, for PA_i, any violation on the due date caused by schedule S_i will impose a delay cost $dc_i(S_i)$, which is a function that monotonically increases with the project completion time $ct_i(S_i) = \max_j\{s_{ij} + d_{ij}\}$. Here we use a linear cost function which is frequently used in the literature: $dc_i(S_i) = p_i \cdot \max\{0, ct_i(S_i) - dd_i\}$, where p_i is the unit delay penalty. Nevertheless, our approach can also adopt non-linear cost functions with the property of monotonic increasing. Also following the previous research, we set the optimality criteria on solution S as the minimization of *Average Project Delay* (APD):

$$APD(S) = \frac{\sum_{i=1}^{N} \max\{0, (ct_i(S_i) - dd_i)\}}{N}. \quad (1)$$

The information of the activities (e.g. d_{ij}, s_{ij}, r^g_{ij}, $r^{l_i}_{ij}$) and local resources (e.g. R_{l_i}, $C_{l_i t}$) are considered as private to each project agent.

4. OUR AUCTION-BASED APPROACH

In general, our approach contains a series of multi-unit combinatorial auctions, and incrementally allocates global resources to PAs. We formulate the auctions following a paradigm similar to [4], where PAs act as bidders, the MA acts as the auctioneer, and the goods to sell are the global resources at each time slot. However, we make the following modifications: (i) we adopt multi-unit combinatorial auction instead of single-unit, since multi-unit global resources need to be handled; (ii) we define the bidder utility based on fixed resource cost instead of changing bundle prices; (iii) we set the objective of the auctioneer as maximizing social welfare instead of revenue, since optimizing revenue may lead to unsatisfactory allocation efficiency [13].

Given G global resources and the scheduling horizon T, there are $G \times T$ goods to sell. We denote the available units of global resource R_g at time t as Ψ_{gt}. Initially, $\Psi_{gt} = C_{gt}$; as the approach proceeds, it is possible that $\Psi_{gt} \leq C_{gt}$. A *multiset* [9] of goods is denoted as a matrix $\Lambda = [\lambda_{gt}]_{G\times T}$, where λ_{gt} is the units of R_g at time t. In each auction round, each participating PA needs to place one bid on the multiset that can (approximately) maximize its *utility*. For PA_i, utility is a function $u_i : \mathbf{\Lambda} \mapsto \Re^+$, $\mathbf{\Lambda} = \{\Lambda | 0 \leq \lambda_{gt} \leq \Psi_{gt}\}$, defined as a quasi-linear function $u_i(\Lambda) = v_i(\Lambda) - RC_i(\Lambda)$, where $v_i(\Lambda)$ is its valuation for Λ and $RC_i(\Lambda)$ is the cost for utilizing Λ. Under fixed unit global resource cost, $RC_i(\Lambda)$ can be calculated as follows:

$$RC_i(\Lambda) = \sum_{g=1}^{G} c_g \sum_{t=1}^{T} \lambda_{gt}. \quad (2)$$

The valuation function will be further detailed in Section 4.1. A bid from PA_i is a pair $B_i =< \Lambda, v_i(\Lambda) >$, where the interested multiset and its valuation are specified. In each round of auction, the MA needs to solve a Winner Determination Problem (WDP) to optimize social welfare, which is the summation of the valuation of the bidders, under the current global resource capacities. Given a set of bids $\mathscr{B} = \{B_\xi | \xi \in \{1, ..., N_b\}\}$ submitted by N_b bidders, a WDP of single-minded bidders (e.g. one bidder is interested in one multiset [13]) can be formulated as:

$$\begin{aligned} \max \quad & \sum_{\xi=1}^{N_b} v_\xi \cdot x_\xi \\ \text{s.t.} \quad & \sum_{\xi=1}^{N_b} \lambda^\xi_{gt} \cdot x_\xi \leq \Psi_{gt}, \ \forall g, t \\ & x_\xi \in \{0, 1\}, \ \forall \xi \end{aligned} \quad (3)$$

Algorithm 1: Our Auction-based Approach

Input: A DRCMPSP case
Output: A solution of the case

1 MA initializes the auction environment, by setting $\overline{SS} \leftarrow \emptyset$, $\overline{IW} \leftarrow \emptyset$, $\overline{US} \leftarrow \{PA_i | i \in \{1, ..., N\}\}$;
2 **while** $\overline{US} \neq \emptyset$ **do**
3 **forall the** $PA_i \in \overline{US}$ **do**
4 Solve a bidding problem, generate a schedule S^*_i and a bid B^*_i ;
5 MA solves a WDP, updates $\overline{IW}$ and $\overline{US}$;
6 MA calculates the demand ratio;
7 **while** $\overline{IW} \neq \emptyset$ **do**
8 **forall the** $PA_i \in \overline{IW}$ **do**
9 Solve a bid modification problem, generate a schedule $S^{*'}_i$ and a bid $B^{*'}_i$;
10 MA solves a WDP, grants global resources to the final winners, and updates $\overline{SS}$ and $\overline{IW}$;
11 Final winners confirm their schedules $S^{*'}_i$;
12 MA updates global resource capacities;
13 **return** $S^* \leftarrow \{S^{*'}_1, ..., S^{*'}_N\}$

Our approach is described in Algorithm 1. Throughout the whole process, three sets of PAs are maintained by the MA: *Scheduled Set* $\overline{SS}$, *Unscheduled Set* $\overline{US}$, and *Initial Winner Set* $\overline{IW}$. Initially, all PAs belong to $\overline{US}$, while $\overline{SS}$ and $\overline{IW}$ are empty. Each round of auction consists of two successive phases: *initial phase* (Line 3 to 6) and *final phase* (Line 7 to 12). In the initial phase, the MA initiates an auction among the unscheduled PAs to determine the *initial winners*. Each PA in $\overline{US}$ generates a bid by solving a bidding problem, then the MA solves a WDP. The initial winners will be moved from $\overline{US}$ to $\overline{IW}$. In the final phase, several rounds of auctions among the initial winners are initiated by the MA to finalize the global resource allocation. In each round of these auctions, each PA in $\overline{IW}$ modifies its bid according to the *demand ratio* (see Definition 5) published by the MA. Then the MA solves a WDP to determine the *final winners* who will be granted the amounts of global resources specified in its bid, and will be moved from $\overline{IW}$ to $\overline{SS}$. Intuitively, the initial phase is used to choose the most "promising" PAs given the current global resource capacities, and the final phase tries to guide these chosen PAs to tune their demand profile in order to provide chances for unscheduled PAs to improve their utilities. Next, we will explain the major components of Algorithm 1 in detail.

4.1 Bid Generation

Here we show how bidders find the multisets that can (approximately) maximize their utility by solving a correspond-

ing single Resource-Constrained Project Scheduling Problem (RCPSP). More formally, a bidder needs to find an optimal or near-optimal solution Λ_i^* of the *bidding problem*:

$$\Lambda_i^* = \underset{\Lambda \in \boldsymbol{\Lambda}}{\operatorname{argmax}}\ u_i(\Lambda), \tag{4}$$

and calculate its valuation $v_i(\Lambda_i^*)$. First we discuss how to evaluate a multiset by introducing the following definition:

Definition 1. A multiset Λ is said to be *feasible* to PA_i, if there exists a feasible schedule S_i that satisfies all the precedence and local/global resource constraints of P_i, when PA_i obtains the amount of global resources specified in Λ.

Now we can define the valuation function as follows:

$$v_i(\Lambda) = \begin{cases} rv_i - DC_i(\Lambda), & \text{if } \Lambda \text{ is feasible} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where $DC_i(\Lambda)$ is the delay cost of Λ. To show how to calculate $DC_i(\Lambda)$, the following definition is needed:

Definition 2. Given a feasible multiset Λ of PA_i, the set $\boldsymbol{S_i^{\Lambda}}$ which contains all feasible schedules of P_i given the amounts of global resources specified in Λ, is called a *solution set*. A solution $\widehat{S}_i^{\Lambda} \in \boldsymbol{S_i^{\Lambda}}$ that has the minimum completion time $ct_i(\widehat{S}_i^{\Lambda})$ is called a *primal schedule* of Λ to PA_i.

Assume a bidder can find a primal schedule $\widehat{S}_i^{\Lambda}$ of Λ, it can safely choose $\widehat{S}_i^{\Lambda}$ and discard other schedules in $\boldsymbol{S_i^{\Lambda}}$, since the delay cost cannot be reduced further by choosing another schedule. Hence, the delay cost of a multiset can be calculated as the delay cost of its primal schedule:

$$DC_i(\Lambda) = dc_i(\widehat{S}_i^{\Lambda}). \tag{6}$$

It seems straightforward to evaluate Λ using Equation (5) and (6). However, it is computationaly intractable to solve these two equations, since firstly in Equation (5) it is hard to determine if a schedule is feasible (equivalent to solving a feasibility problem of RCPSP which is NP-hard [16]), and secondly in Equation (6) it is hard to find the primal solution (equivalent to solving a RCPSP which is NP-hard [3]). On the other hand, to generate a feasible multiset from a feasible schedule is quite easy, as stated in the following definition:

Definition 3. Given a schedule $S_i = (s_{i1}, ..., s_{iJ_i})$ of P_i, the multiset $\widehat{\Lambda}_i = \widehat{\Lambda}_i(S_i)$ calculated as:

$$\widehat{\lambda}_{gt}^i(S_i) = \sum_{a_{ij} \in A_i(t)} r_{ij}^g, \tag{7}$$

where $A_i(t) = \big\{a_{ij}, j \in \{1, ..., J_i\} \big| s_{ij} \le t, (s_{ij} + d_{ij}) > t\big\}$, is called a *core*[4] of PA_i.

Definition 2 and 3 build connections between schedules and multisets. Given a feasible schedule, a feasible multiset (which is a core) can be obtained immediately using Equation (7); given a feasible multiset, there must exist a feasible schedule that is primal, since the solution set cannot be empty. In Section 5.1, we will show that a candidate multiset of the bidding problem must be a core. However, it is still hard to give the exact valuation of a core, due to the hardness on computing Equation (6). Here we estimate a core's delay cost using the delay cost of its corresponding schedule. Then the approximate valuation of core $\widehat{\Lambda}_i(S_i)$ is:

$$\widetilde{v}_i(\widehat{\Lambda}_i(S_i)) = rv_i - dc_i(S_i). \tag{8}$$

The approximate utility can be calculated as:

$$\widetilde{u}_i(\widehat{\Lambda}_i(S_i)) = rv_i - dc_i(S_i) - RC_i(\widehat{\Lambda}_i(S_i)). \tag{9}$$

Given a feasible schedule S_i, the computation of Equation (8) and (9) is trival. In Section 5.1, we will show that the above equation provides an approximation to the exact utility of the optimal solution of the bidding problem, with an error that monotonically increases with $ct_i(S_i)$. Hence, we can approximately solve the bidding problem by finding a sub-optimal schedule of the corresponding RCPSP (with the global resource capacities equal to Ψ_{gt}), then generate the corresponding core and calcualte the approximate valuation using Equation (7) and (8), respectively. Here we adopt the parallel schedule generation scheme with one of the best priority-rule *Latest Finish Time* [8], to generate a near-optimal schedule. However, all the resources considered in [8] have constant capacities over the whole scheduling horizon, which cannot be directly applied to our problem. Thus, we modify the activity selection step of the original algorithm by checking the global and local resource capacities over all the duration of an activity, instead of simply checking the first time slot.

4.2 Bid Modification

In a DRCMPSP case, requirements for different time slots of global resource are not even; some time slots may be demanded more heavily than others. Since winners will be granted their bids in the end of each round, if they can switch to other bids producing the same utility but deviated from heavily demanded time slots, other unscheduled PAs may have better chances to improve their utilities. Based on this idea, we design a bid modification step, where the winners try to modify their bids based on the *demand ratio* provided by the MA as demand status feedback. To define the bid modification problem, first we define the *equivalent set*:

Definition 4. For PA_i, given a core $\widehat{\Lambda}_i = \widehat{\Lambda}_i(S_i)$ generated by a schedule S_i, the *equivalent set* of $\widehat{\Lambda}_i$ is defined as $\mathscr{A}(\widehat{\Lambda}_i) = \Big\{\widehat{\Lambda}_i(S_i') \Big|\ ct_i(S_i') = ct_i(S_i)\Big\}$.

As will be shown in Lemma 1 in Section 5.1, the equivalent set consists of cores that produce the same approximate utility to a PA. Next we give the following definitions:

Definition 5. Given a set of bids $\mathscr{B}$, the *demand ratio* is defined as a matrix $D(\mathscr{B}) = [\delta_{gt}(\mathscr{B})]_{G\times\ T}$, where

$$\delta_{gt}(\mathscr{B}) = \left(\sum_{\xi=1}^{N_b} \lambda_{gt}^{\xi}\right) / \Psi_{gt}. \tag{10}$$

Definition 6. The *resource index* of a multiset Λ under demand ratio D is defined as:

$$RI(\Lambda, D) = \sum_{g=1}^{G}\sum_{t=1}^{T} \lambda_{gt}\delta_{gt}. \tag{11}$$

The intuition of Definitions 5 and 6 is that, a higher aggregate demand on a resource time slot with lower capacity results in a higher demand ratio, and a multiset with a lower resource index is more likely to avoid the heavily demanded time slots. Now we define the *bid modification problem*:

[4]Note that this definition is different from the commonly used definition of core in game theory.

Algorithm 2: Bid Modification

Input: A schedule S_i^* and demand ratio D
Output: Modified schedule $S_i^{*'}$ and bid $B_i^{*'}$

1 **if** *Global resource capacities changed* **then**
2 Solve the bidding problem;
3 Calculate the activity slackness, and put all the flexible activities in FJ_i;
4 **while** $FJ_i \neq \emptyset$ **do**
5 Virtually shift all the flexible activities within their slackness, and record the best shifting with the most reduction in resource index;
6 Apply the best shifting, and remove the selected activity from FJ_i;
7 Update the activity slackness;
8 Calculate a bid $B_i^{*'} \leftarrow < \Lambda_i^{*'}, u_i^{*'} >$ based on the modified schedule;
9 **return** $S_i^{*'}, B_i^{*'}$

Definition 7. Given demand ratio D and multiset Λ_i^* in the solution of the bidding problem, the problem of finding

$$\Lambda_i^{*'} = \underset{\Lambda \in \mathscr{A}(\Lambda^*)}{\operatorname{argmin}} \ RI(\Lambda, D) \tag{12}$$

is called the bid modification problem.

The above problem is NP-hard since it can be transformed to a RCPSP with an additional hard due date constraint. Here we design a polynomial-time algorithm to approximately solve this problem, as shown in Algorithm 2. This algorithm generates an equivalent multiset by shifting flexible activities in an existing schedule. An activity a_{ij} in a schedule S_i is flexible if its *slackness* $slk_{ij} = \{t | est_{ij} \leq t < lst_{ij}\}$ is not empty, where:

$$\begin{aligned} est_{ij} &= \max\left\{ad_i, \max_{a_{ij'} \prec a_{ij}} \{s_{ij'} + d_{ij'}\}\right\}, \\ lst_{ij} &= \min\left\{ct_i(S_i), \min_{a_{ij} \prec a_{ij'}} \{s_{ij'} - d_{ij}\}\right\}. \end{aligned} \tag{13}$$

In Algorithm 2, PA_i may need to update its bid before activity shifting procedures by solving a bidding problem, since the global resource capacities could change during the finalization phase (Line 1 to 2). In activity shifting procedures, first all the flexible activities are detected and put into a set FJ_i (Line 3). Then, an iterative process (Line 4 to 7) shifts $a_{ij} \in FJ_i$ to start at a global and local resource feasible time $t \in slk_{ij}$ which reduces the resource index value most, and removes a_{ij} from FJ_i. When FJ_i is empty, a new schedule S_i' is found, and a modified bid is generated (Line 8). A simple example of the bid modification is given in Figure 1, where two projects compete for one global resource with a capacity of 5. Both projects can start from time 0. Figure 1(a) shows the schedules generated by two successive rounds of auctions (without modification), where PA_1 wins the first round and PA_2 wins the second round. The demand profile of PA_1 and demand ratio of the first round is given in Figure 1(b). As in Figure 1(c), if PA_1 can shift its flexible activity a_{12} to start at time 2 according to the demand ratio, PA_2 can achieve a better schedule which completes at time 8, instead of time 10 in Figure 1(a). The modified demand profile of PA_1 is shown in Figure 1(d).

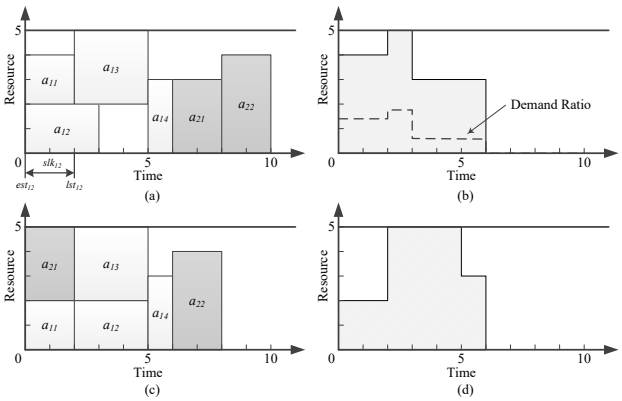

Figure 1: An example of bid modification

4.3 Winner Determination

The WDP is proved as NP-complete [19]. A polynomial-time algorithm with a proved upper-bound for solving WDP among single-minded bidders is provided in [6]. The algorithm first sorts all the bids based on a given criterion, then examines each bid in order and labels the bidder as winner if its bid does not conflict with the previous winners. Here we adopt this algorithm with the following sorting criterion:

$$\gamma(B_\xi) = \frac{v_\xi}{\sqrt{\sum_{g=1}^{G}\sum_{t=1}^{T} \lambda_{gt}^{\xi} / \Psi_{gt}}}. \tag{14}$$

A bid with a higher γ value is more likely to be a winner in the optimal solution. Hence, all bids are first sorted descendingly according to their γ values, then examined one by one to check if there is any violation on resource capacity.

5. ANALYSIS

We provide analysis about our approach in this section, including some formal proofs to support our design, and detailed analysis on its information exchange and complexity.

5.1 Theoretical Support for Our Design

In Section 4.1, we design an efficient method to approximately solve the bidding problem by generating a multiset (i.e. core) given a schedule using Equation (7). Here, we formally prove that the multiset generated in this way must be a candidate of the optimal solution of the bidding problem, and analyze the error of the approximate utility in Equation (9). We first prove that under fixed resource cost, a core satisfies the properties described in the following lemma:

Lemma 1 *For PA_i, the following statements hold: (1) all cores have the same resource cost; (2) the resource cost of a core is the lowest among all the feasible multisets; (3) cores generated from schedules with the same completion time has the same approximate utility.*

Proof Introduce an auxiliary variable $\overline{r}_{ij,t}^g$ for each a_{ij}:

$$\overline{r}_{ij,t}^g = \begin{cases} r_{ij}^g, & \text{if } s_{ij} \leq t < (s_{ij} + d_{ij}) \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

Then a core $\widehat{\Lambda}_i$ defined by Equation (7) can be rewritten as

$$\widehat{\lambda}_{gt}^i = \sum_{a_{ij} \in A_i(t)} r_{ij}^g = \sum_{j=1}^{J_i} \overline{r}_{ij,t}^g. \tag{16}$$

Putting the above equation into Equation (2), we have

$$RC_i(\widehat{\Lambda}_i) = \sum_{g=1}^{G} c_g \sum_{t=1}^{T} \sum_{j=1}^{J_i} \overline{r}_{ij,t}^{g} = \sum_{g=1}^{G} \sum_{j=1}^{J_i} c_g r_{ij}^{g} d_{ij}. \quad (17)$$

Since c_g, r_{ij}^g and d_{ij} are all constant, $RC_i(\widehat{\Lambda}_i)$ is also a constant regardless of $\widehat{\Lambda}_i$, the first statement holds. Assuming there exists a multiset $\widetilde{\Lambda}$ such that $RC_i(\widetilde{\Lambda}) < RC_i(\widehat{\Lambda}_i)$, according to Equation (17), at least one activity $a_{i\widetilde{j}}$, $\widetilde{j} \in \{1, .., J_i\}$ cannot obtain enough amount of global resource R_g during at least one time slot in $d_{i\widetilde{j}}$, which means $\widetilde{\Lambda}$ is not a feasible multiset. Hence the second statement holds. The third statement can be obtained immediately according to the first statement and Equation (9). □

Now we are ready to prove the following proposition:

Proposition 1 *The optimal multiset Λ_i^* of PA_i defined in Equation (4) must be the core generated by the schedule S_i^* which has the minimum completion time regarding the current global resource capacity Ψ_{gt}.*

Proof According to Equation (5), any infeasible multiset cannot be optimal. For any feasible multiset Λ, according to Lemma 1 and Definition 2, it is always true that $u_i(\widehat{\Lambda}_i(\widehat{S}_i^{\Lambda})) \geq u_i(\Lambda)$, hence an optimal solution must be a core. Since all cores have the same resource cost, the core Λ_i^* generated by S_i^* solves the bidding problem to optimal, since Λ_i^* results in the lowest delay cost. □

Proposition 1 shows that any multiset that is not a core cannot be optimal, and the optimal solution of the bidding problem is $\Lambda_i^* = \widehat{\Lambda}_i(S_i^*)$. Noted that the estimated utility $\widetilde{u}_i(\Lambda_i^*)$ calculated using Equation (9) is the same as its exact utility $u_i(\Lambda_i^*)$, since $DC_i(\Lambda_i^*) = dc_i(S_i^*)$. Hence, for any feasible schedule S_i and the estimated utility $\widetilde{u}_i(\widehat{\Lambda}_i(S_i^*))$, the error from the optimal utility $u_i(\Lambda_i^*)$ is $err = \widetilde{u}_i(\widehat{\Lambda}_i(S_i)) - \widetilde{u}_i(\Lambda_i^*) = dc_i(S_i) - dc_i(S_i^*)$.

5.2 Information Exchange

In our approach, only two kinds of information need to be exchanged between PAs and the MA, namely bids and demand ratio. Beyond them, PAs do not need to specify any private information regarding activities and local resources. More importantly, it is very hard to obtain these kinds of private information only from the bids and demand ratio. Nevertheless, they need to provide truthful valuation on the global resource requirements that they are interested in.

Besides privacy preserving, another advantage of our design is that, the bids provide more precise information for the decisions on global resource allocation. Though utilities are approximated using Equation (8) and (9), the actual utility a bidder will gain after being granted $\widehat{\Lambda}(S_i)$ cannot be lower than the approximate value, assuming it will not apply a schedule with a larger completion time than $ct_i(S_i)$. On the contrary, activity-level approaches directly allocate global resources to individual activities according to some activity-level criteria. However, these criteria cannot exactly reflect the individual objectives (e.g. makespan, revenue) of each project, since these objectives can only be known exactly when all the activities are scheduled. This is more evident under tighter resource constraints, since it is more difficult for a PA to predict its objective without knowing the global resource demand status in the following decision process. This is also confirmed by our experiments in Section 6.1, as our approach improves the results of DMAS/EM more on cases with tighter resource constraints.

5.3 Complexity Analysis

The complexity of our modified RCPSP algorithm in Section 4.1 is $O\left(J_i^2(G+L_i)d_i^*\right)$, where $d_i^* = \max\{d_{i1}, ..., d_{iJ_i}\}$. For Algorithm 2, the worst-case complexity of calculating activity slackness is $O(J_i^2)$, hence the worst-case complexity of Algorithm 2 is $O(J_i^2\sigma_i)$, where $\sigma_i = \max\{(G+L_i)d_i^*T, J_i\}$. For the WDP algoritm, the complexity to calculate Equation (14) is $O(GT)$, hence the worst-case complexity of this algorithm is $O(N\omega)$, where $\omega = \max\{GT, \log N\}$.

If the scheduling horizon is too small, it is possible that some PAs cannot find a feasible solution for the bidding problem. Whenever this happens, the scheduling horizon can be extended. Here we assume the scheduling horizon is enough for all PAs to generate feasible multisets until Algorithm 1 terminates. Hence, the WDP algorithm can guarantee that at least one winner will be found in each round of auction. We denote the number of outer rounds between Line 2 and 12 of Algorithm 1 as N', the number of inner rounds between Line 7 and 12 in outer round n as m'_n, and the number of initial winners in outer round n as m_n. Based on the operations to $\overline{SS}$, $\overline{IW}$ and $\overline{US}$, we have $N' \leq N$, $m'_n \leq m_n$, and $\sum_{n=1}^{N'} m_n = N$. Hence, the total number of WDPs the MA needs to solve is $N' + \sum_{n=1}^{N'} m'_n \leq N' + N \leq 2N$. This means that in the worst-case scenario, Algorithm 1 terminates within $O(N)$ rounds of auctions. This upper bound on the number of auction rounds is usually much less than iBundle. As shown in [17], in the worst case, the maximum number of auction rounds for iBundle to terminate is proportional to the number of possible multiset for all the PAs, which grows exponentially with the number of items in the auctions.

Based on the above analysis, MA need to solve $2N$ WDPs at most, leading to a worst-case complexity of $O(N^2\omega)$. Assume a PA becomes an initial winner at outer round i, then in the worst case it needs to participate $i + m_i$ WDPs. Since $m_i \leq N - \sum_{n=1}^{i-1} m_n \leq N - (i-1)$, we have $i + m_i \leq N + 1$. Hence the worst-case complexity of a PA is $O(NJ_i^2\sigma_i)$.

6. EMPIRICAL EVALUATION

We test our approach on two problem sets. The first one contains all the 140 cases from MPSPLIB. To the best of our knowledge, this is the only public benchmark of DRCMPSP. We compare our approach with state-of-the-art decentralized approaches DMAS/EM [24], DMAS/ABN [1], and a centralized approach SASP on this problem set. We also compare our approach with the other combinatorial auction based approach in [4] (denoted as Confessore's). Since this approach can only handle one single-unit global resource, we conduct experiments on the second problem set, which is generated from MPSPLIB by replacing the multi-unit global resources in each case with one single-unit global resources.

6.1 Experiments on the First Problem Set

In this section, we describe the experiments on the first problem set. The 140 cases from MPSPLIB are divided into 20 subsets named as "MP_J_N", where $J \in \{30, 90, 120\}$ is the number of activities per project, and $N \in \{2, 5, 10, 20\}$ is the number of projects. Thus the largest cases contain $20 \times 120 = 2400$ activities. Each case contains 4 resource

Table 1: Comparison with other approaches

Subset	Our Approach	DMAS/EM	DMAS/ABN	SASP
MP_30_2	13.6	**8.9**	15.9	22.4
MP_90_2	**6**	6.6	9.9	18.5
MP_120_2	**50.6**	59.4	67.2	69.1
MP_30_5	19.76	**17**	21.2	31.9
MP_90_5	10.72	**4.6**	11	23.8
MP_120_5	**45.92**	54.2	66.48	71.9
MP_30_10	**55.78**	66.4	87.5	90.2
MP_90_10	**39.02**	50.9	46.08	65
MP_120_10	**107.14**	119.6	130.96	139.6
MP_30_20	**116.1**	138	207.96	185.5
MP_90_20	**20.95**	27.4	30.22	48.6
MP_120_20	**24.27**	28.6	37.18	61.1
MP_90_2AC	**108.15**	126	144.15	158.6
MP_120_2AC	**37.75**	52.4	47	56.7
MP_90_5AC	**249.42**	284.6	384.08	404.6
MP_120_5AC	**181.3**	233.5	291.44	258.8
MP_90_10AC	**175.23**	223.3	313.33	283.9
MP_120_10AC	**103.74**	169.4	171.54	181
MP_90_20AC	**94.05**	126.7	146.37	161.8
MP_120_20AC	**163.36**	280.7	297.39	297.4

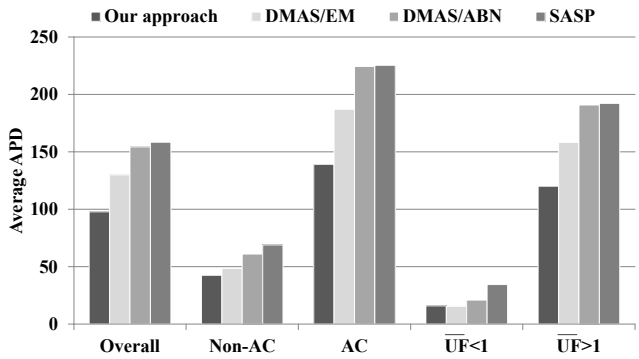

Figure 2: Comparison of average APD

per project, and the number of global resource G in each case is chosen between 1 and 4. The cases with no local resource, i.e. $G = 4$, are called "Agent Cooperation" cases, and a postfix "AC" is added to the subset names. Each AC subset contains 10 cases, while each non-AC subset contains 5 cases. To measure the tightness of global resource constraints, a Utilization Factor (UF) [7] value is calculated for each case. $UF < 1$ indicates a low to medium resource constraint, while $UF > 1$ indicates a medium to high constraint [14]. Average UF value $\overline{UF}$ of each subset can be found in [7]. In general, AC subsets have higher $\overline{UF}$ than non-AC subsets given the same N and J, which makes AC subsets harder to solve.

6.1.1 Performance Comparison

Our approach is implemented using Java 1.8, and runs on a single Intel Xeon Workstation (3.5GHz, 16GB). We set the scheduling horizon $T = 1500$ for all the cases. All the 140 cases are successfully solved within a total time of 170 seconds. We calculate the average APD of each subset in Table 1, where the best results are marked as bold. Among the 20 subsets, our approach outperforms other approaches in 17 subsets. For MP_120_20AC, one of the most complex subset with the tightest resource constraints, our approach outperforms DMAS/EM by 41.8%.

To evaluate the performance of our approach in different types of subsets, we first split the problem subsets into two groups consist of only non-AC and AC cases, and calculate the average APD obtained by the four approaches. Then

Table 2: Comparison with DMAS/EM

Type	Overall	Non-AC	AC	$\overline{UF} < 1$	$\overline{UF} > 1$
Our approach	**97.71**	**42.49**	**139.13**	15.88	**120.03**
DMAS/EM	129.89	48.47	187.08	**15.52**	158.26
Difference (%)	-24.77	-12.34	-25.63	2.32	-24.15

Table 3: Results with and without bid modification

Subset	Mod	NoMod	Subset	Mod	NoMod
MP30_2	**13.6**	13.7	MP90_20	**20.95**	21.28
MP90_2	6	**5.8**	MP120_20	**24.27**	24.39
MP120_2	**50.6**	50.7	MP90_2AC	**108.15**	108.35
MP30_5	19.76	**19.6**	MP120_2AC	**37.75**	38.3
MP90_5	**10.72**	11.2	MP90_5AC	**249.42**	249.72
MP120_5	**45.92**	46	MP120_5AC	**181.3**	181.76
MP30_10	55.78	**55.38**	MP90_10AC	**175.23**	175.75
MP90_10	**39.02**	39.34	MP120_10AC	**103.74**	104.6
MP120_10	**107.14**	108.24	MP90_20AC	**94.05**	95.29
MP30_20	**116.1**	117.38	MP120_20AC	**163.36**	163.56

we group the subsets according to if $\overline{UF} > 1$, and calculate the corresponding average APD. We plot these values along with the overall average APD of all the 140 cases in Figure 2. As presented, our approach produces the lowest average APD compared with other three approaches. Moreover, our approach generates better results for three types of subsets non-AC, AC, and $\overline{UF} > 1$; for $\overline{UF} < 1$, our result is comparable with the best result obtained by DMAS/EM. In Table 2, we compare our approach with DMAS/EM, which produces the closest results to ours. As shown, overall our approach outperforms DMAS/EM by 24.77%. For the two harder groups AC and $\overline{UF} > 1$, our approach gives 25.63% and 24.15% improvements against DMAS/EM, respectively. In general, our approach generates results with lower APD compared to other three approaches, and performs better on harder subsets with tighter global resource constraints.

To show the effectiveness of bid modification, we compare our approach (Mod) with the modified one (NoMod) that replaces the bid modification in Line 9 of Algorithm 1 with bid generation. In Table 3, our approach with bid modification shows signs of improvement on the APD in 17 out of 20 subsets, and has better results in all AC subsets. A possible reason for the marginal improvement is that the bids generated by our method already could provide effective information for allocating global resources, hence the room for bid modification to further improve the results is limited. Since Algorithm 2 is efficient, bid modification does not affect the scalability of this approach.

6.1.2 Scalability Evaluation

We evaluate the scalability of our approach according to the total time for solving one case, since it is simulated on a single computer. Since d_i^* and $G + L_i$ are constants in all the cases of MPSPLIB and T is fixed, the worst-case complexity of the MA and PA are $O(N^2 \log N)$ and $O(NJ_i^3)$. Thus, if Algorithm 1 runs on a single thread, the worst-case complexity to generate a solution is $O(N^2 \max\{\log N, J^{*3}\})$, where $J^* = \max\{J_1, ..., J_N\}$. We plot the average execution time against N and J^* in Figure 3, by fixing one parameter and increasing the other. The time increasing trends in Figure 3 are compatible with our complexity analysis and show that our approach can efficiently solve large problem cases with thousands of activities from tens of projects.

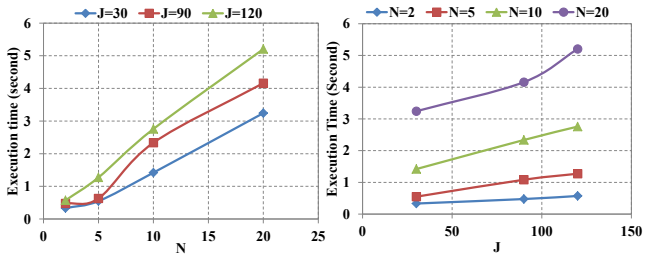

Figure 3: Execution time on MPSPLIB cases: (left) fix J^*, and (right) fix N

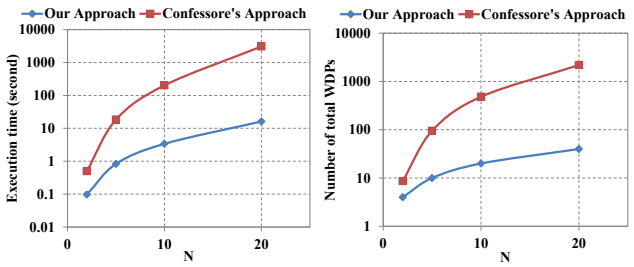

Figure 4: Scalability comparison on the second problem set with Confessore's approach

Table 4: Number of cases solved by Confessore's approach for each subset of the second problem set

Subset	Total	Solved	Ratio(%)	Subset	Total	Solved	Ratio(%)
MP30_2	5	5	100	MP30_10	5	4	80
MP90_2	15	15	100	MP90_10	15	10	67
MP120_2	15	15	100	MP120_10	15	14	93
MP30_5	5	5	100	MP30_20	5	4	80
MP90_5	15	12	80	MP90_20	15	3	20
MP120_5	15	14	93	MP120_20	15	9	60

Table 5: Comparison with respect to problem subset of the second problem set

Subset	Our Approach	Confessore's Approach	Diff(%)
MP30_2	**106.4**	122.8	-13.36
MP90_2	**354.27**	393.7	-10.02
MP120_2	**362.2**	439.6	-17.61
MP30_5	**167.28**	279.8	-40.21
MP90_5	**641.02**	869.67	-26.29
MP120_5	**993.36**	1281.24	-22.47
MP30_10	**393.3**	667.08	-41.04
MP90_10	**1504.75**	2111.3	-28.73
MP120_10	**1564.11**	2360.66	-33.74
MP30_20	**937.39**	1540.06	-39.13
MP90_20	**1591.27**	2800.17	-43.17
MP120_20	**3037.74**	5075.11	-40.14

6.2 Experiments on the Second Problem Set

In this section, we describe the experiments on the second problem set, which contains 140 cases generated from the MPSPLIB. More specifically, for each case in MPSPLIB, we first keep the global resource that has the minimum capacity and remove other ones, and then replace the capacity and requirements of each activity on that resource with 1. We classify the newly generated cases based on the number of activities and projects, hence 12 subsets can be obtained.

6.2.1 *Performance Comparison*

We implement Confessore's approach using Java 1.8, and run both Confessore's and our approach on the same workstation as in the previous section. During experimentation, we observe that Confessore's approach cannot converge on some cases, hence we limit the maximum iterations of this approach to 3000. The scheduling horizon T is set to 15000. All the 140 cases in this problem set are successfully solved by our approach within 730 seconds, while Confessore's approach can solve 79% (110/140) of all cases in about 15 hours. We list the number of cases solved by Confessore's approach for each subset in Table 4. From this table, we can further conclude that when the number of projects is 2, 5, 10, 20, Confessore's approach can solve 100%, 88.6%, 80%, 46% of the cases, respectively. This indicates that Confessore's approach may not be able to solve large cases where tens of projects are involved. In the following evaluation, we only consider those cases that are solved by both approaches.

We then evaluate the solution quality of these two approaches by calculating the average APD of each subset in Table 5, which shows that our approach consistantly achieves lower APD on all the subsets, and the improvement tends to increase on harder cases with more activities and projects.

6.2.2 *Scalability Comparison*

To compare the scalability of our approach and Confessore's approach, we calculate the average execution time and the number of total WDPs regarding the project number N. As shown in Figure 4 (vertical axes are in log scale), both the execution time and number of WDPs of our approach are much smaller than those of Confessore's approach. When N is larger than 10, our approach can be two orders of magnitude faster. These results clearly show that our approach is computationally frugal, and can efficiently solve large cases where tens of projects are involved.

7. CONCLUSION AND FUTURE WORK

In conclusion, this paper presents a novel decentralized approach for solving DRCMPSP based on multi-unit combinatorial auction. It does not require private information of projects, and can efficiently find solutions with much lower average project delay than state-of-the-art decentralized approaches. As our approach runs in polynomial time for all agents, it can easily scale to large problems with thousands of activities from tens of projects.

Though the model of DRCMPSP is general enough to describe a large class of decentralized scheduling problems, some real-world problems could be more complex (e.g. with hierarchical activity structure and additional constraints between global resources). For future work, we will extend our approach to handle these new problems. Experimental results show that our approach has good allocation efficiency. However, incentive compatibility remains an unsolved issue. It is of great importance to provide incentives for self-interested agents to provide truthful information, especially in a decentralized environment. Thus, another future research direction for us is to address this issue.

Acknowledgments

This work was conducted within the Rolls-Royce@NTU Corporate Lab with support from the National Research Foundation (NRF) Singapore under the CorpLab@University Scheme.

REFERENCES

[1] S. Adhau, M. Mittal, and A. Mittal. A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach. *Engineering Applications of Artificial Intelligence*, 25(8):1738–1751, 2012.

[2] O. Amir, G. Sharon, and R. Stern. Multi-agent pathfinding as a combinatorial auction. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 2003–2009, 2015.

[3] J. Blazewicz, J. K. Lenstra, and A. R. Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

[4] G. Confessore, S. Giordani, and S. Rismondo. A market-based multi-agent system model for decentralized multi-project scheduling. *Annals of Operations Research*, 150(1):115–135, 2007.

[5] A. Fink and J. Homberger. Decentralized multi-project scheduling. In *Handbook on Project Management and Scheduling Vol. 2*, pages 685–706. Springer, 2015.

[6] R. Gonen and D. Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *Proceedings of the 2nd ACM conference on Electronic Commerce (EC'00)*, pages 13–20, 2000.

[7] J. Homberger. A (μ, λ)-coordination mechanism for agent-based multi-project scheduling. *OR Spectrum*, 34(1):107–132, 2012.

[8] R. Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996.

[9] P. Krysta, O. Telelis, and C. Ventre. Mechanisms for multi-unit combinatorial auctions with a few distinct goods. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'13)*, pages 691–698, 2013.

[10] I. Kurtulus and E. Davis. Multi-project scheduling: Categorization of heuristic rules performance. *Management Science*, 28(2):161–172, 1982.

[11] E. Kutanoglu and S. D. Wu. On combinatorial auction and lagrangean relaxation for distributed resource scheduling. *IIE transactions*, 31(9):813–826, 1999.

[12] J. S. Lau, G. Q. Huang, K.-L. Mak, and L. Liang. Agent-based modeling of supply chains for distributed scheduling. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(5):847–861, 2006.

[13] D. Lehmann, L. I. Oćallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[14] A. Lova and P. Tormos. Analysis of scheduling schemes and heuristic rules performance in resource-constrained multiproject scheduling. *Annals of Operations Research*, 102(1-4):263–286, 2001.

[15] X. Mao, N. Roos, and A. Salden. Stable multi-project scheduling of airport ground handling services by heterogeneous agents. In *Proceedings of The 8th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'09)-Volume 1*, pages 537–544, 2009.

[16] K. Neumann, C. Schwindt, and J. Zimmermann. *Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions*. Springer Science & Business Media, 2003.

[17] D. C. Parkes and L. H. Ungar. Iterative combinatorial auctions: Theory and practice. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00)*, pages 74–81, 2000.

[18] D. C. Parkes and L. H. Ungar. An auction-based method for decentralized train scheduling. In *Proceedings of the 5th International Conference on Autonomous Agents (AAMAS'01)*, pages 43–50, 2001.

[19] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Svystems (AAMAS'02)*, pages 69–76, 2002.

[20] A. Stranjak, P. S. Dutta, M. Ebden, A. Rogers, and P. Vytelingum. A multi-agent simulation system for prediction and scheduling of aero engine overhaul. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'08): Industrial Track*, pages 81–88, 2008.

[21] W. E. Walsh and M. P. Wellman. Decentralized supply chain formation: A market protocol and competitive equilibrium analysis. *Journal of Artificial Intelligence Research*, pages 513–567, 2003.

[22] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and economic behavior*, 35(1):271–303, 2001.

[23] H. Xi, C. K. Goh, P. S. Dutta, M. Sha, and J. Zhang. An agent-based simulation system for dynamic project scheduling and online disruption resolving. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*, pages 1759–1760, 2015.

[24] Z. Zheng, Z. Guo, Y. Zhu, and X. Zhang. A critical chains based distributed multi-project scheduling approach. *Neurocomputing*, 143:282–293, 2014.