Robust Distributed Scheduling via Time-Period Aggregation

Shih-Fen Cheng[1] John Tajan Hoong Chuin Lau

School of Information Systems
Singapore Management University
80 Stamford Road, Singapore 178902
{sfcheng, jtajan, hclau}@smu.edu.sg

Abstract

In this paper, we evaluate whether the robustness of a market mechanism that allocates complementary resources could be improved through the aggregation of time periods in which resources are consumed. In particular, we study a multi-round combinatorial auction that is built on a general equilibrium framework. We adopt the general equilibrium framework and the particular combinatorial auction design from the literature, and we investigate the benefits and the limitation of time-period aggregation when demand-side uncertainties are introduced. By using simulation experiments on a real-life resource allocation problem from a container port, we show that, under stochastic conditions, the performance variation of the process decreases as the time frame length (time frames are obtained by aggregating time periods) increases. This is achieved without causing substantial deterioration in the mean performance.

The main driver for the increase in robustness is that longer time frames result in allocations where resources are assigned in longer contiguous time blocks. The resulting resource continuity allows bidders to shift schedules upon realization of stochasticity. To demonstrate the generality of the notion that resource continuity increases allocation robustness, we perform further experiments on a decentralized variant of the classical job shop scheduling problem. The experiment results demonstrate similar benefits.

Keywords: market-based resource allocation, uncertainty, auction, scheduling, robustness

1 Introduction

Allocating resources for problems with decentralized information and control is a challenge for both planners and agents. From the planner's perspective, s/he has to design protocols that would function well even with incomplete information from and limited control over these free-willed agents. From the agent's perspective, the challenge lies in the strategic nature of the resource allocation process; no matter what resource allocation protocol is chosen, every agent has to plan his/her acts considering the existence of other selfish and unpredictable agents. Adding in uncertainties that might come from either planners or agents, we have a problem domain that is well-sought-after but still a long way to go before being claimed well-understood.

The challenges posted by uncertainties can be elaborated more from both planner's and agent's perspectives. For planners, uncertainties might be introduced if managed resources are subject to unexpected breakdown or service disruptions. For agents, uncertainties might come from dynamic task arrivals or delayed (or canceled) task assignments. With the assumption of centralized control, a wide array of methodologies (e.g., stochastic programming and Markov decision process) are already shown to be quite effectively in handling uncertainties. However, in decentralized scenarios populated with selfish agents, the successes of these methodologies are much more restricted, mostly due to the absence of necessary information (agents might be unwilling to share private information) and the difficulty in enforcing resources allocation plan (agents are not likely to compromise their own interests).

Instead of trying to control agents using centralized approaches, we rely on the use of markets, which is arguably the most well-studied mechanism in addressing issues related to decentralized information and control [Wellman and

[1]Corresponding author

Wurman, 1998]. Agent's independence is respected in the market mechanism and a globally optimal allocation could be achieved if the market mechanism is designed properly [Ygge and Akkermans, 1999]. However, the market is not panacea for all decentralized resource allocation problems, and has to be used with care. One critical issue that determines the effectiveness of a market mechanism is the handling of uncertainties. From the agent's perspective, if it is exposed to significant risks of either buying useless resources or missing critical resources (both due to the uncertainty in its resource demand) by participating in the market, it might choose to back off. An agent certainly has to manage these risks by adopting more advanced bidding or demand prediction algorithms; however, the designer of the market mechanism should also help agents eliminate some of these risks whenever possible. Traditionally, the market designer can help agents by either allowing *decommitment* or providing *secondary markets*. By allowing decommitment, agents could forfeit their earlier commitments in using certain resources by paying penalties. Agents could thus adjust their resource usage plans if any disruption happens (e.g., due to cancelled or unexpected new jobs). With secondary markets, agents could trade among themselves after the resources are allocated if necessary. This design again allows agents to deal with unexpected disruptions.

Our approach differs from these earlier approaches in that we do not plan to introduce new rules to existing mechanisms, instead, we would explore the use of time-period aggregation in reducing impacts resulting from uncertainties. This idea is fairly simple, and the intuition behind it is based on risk pooling. In our case, we distribute resources in *time frames* that are aggregated from multiple consecutive time periods so that variability resulting from demand or supply disruptions could be mitigated. To be more specific, we are interested in finding a common aggregation factor (number of time periods to be included in a time frame) so that performance variability (measured by coefficient of variation) can be significantly reduced while trading off measures like absolute performance and allocation efficiency.

The rest of the paper is organized as follows. Section 2 reviews related literature on market-based approaches. Section 3 introduces motivational example in container port operation, describes the combinatorial auction framework, and formally defines the time-period aggregation approach. A simple two-period inventory model is introduced in Section 4 to explain the intuition behind the time-period aggregation approach. Section 5 generally describes the bid generation and price adjustment processes under the assumption that time periods can be aggregated. Sections 6 and 7 present numerical results from the container port operation and the general job shop scheduling problem. Finally, Section 8 concludes the paper.

2 Review of Past Work

Past work on market-based approaches in resource allocation covers a wide range of applications. Sutherland [1968] was one of the first to propose using markets in allocating computing resources. In cases where we can assume convex preferences and competitive economy, researchers have found promising results computationally [Cheng and Wellman, 1998, Ygge and Akkermans, 1999]. For scheduling problems, these assumptions do not hold in general, as it is quite common for scheduling problems to have discrete resources (time slots) and complementary preferences, and various special designs are proposed [Crawford and Veloso, 2006, Kutanoglu and Wu, 1999, Vidal, 2003, Wellman et al., 2001].

Despite the success of using markets in various setups, a number of inefficiencies might occur when markets are used in allocating resources. One of the major sources of inefficiencies is the complementarity among resources. That is, when multiple independently allocated resources are required in order to accomplish a single task, an agent might end up with only part of the required resources, and efficiency is lost as a result. This issue can be directly addressed by running a combinatorial auction [Sandholm, 2002]. Although we use a particular type of combinatorial auction in our research, the design of combinatorial auction is not our main focus and thus we will not go too deep into the vast amount of literature on it. Cramton et al. [2006] provide a thorough treatment on the subject.

Another major source of inefficiency is caused by uncertainties. Just as complementarities might cause an agent to unexpectedly miss some critical resources required for accomplishing its tasks, an agent's objective function might be altered unexpectedly either due to changes to its current tasks or dynamic arrivals of new tasks. These changes would result in different resource requirements and an agent's currently committed resources might not satisfy the latest requirement. Some authors suggest the use of finance-inspired mechanisms like options and futures in dealing with these uncertainties [Sutherland, 1968]. In our work we focus on how to use time-period aggregation in dealing with these uncertainties. The idea of time-period aggregation is first explored by Cheng et al. [2008]. This paper is a significant extension in generalizing the idea.

3 Problem Statement and Solution Methodology

The problems we are interested in studying are stochastic, decentralized, and require complementary resources. In particular, we would investigate scheduling problems with these properties. Applications with these properties could be easily found in many important domains.

We would like to emphasize that it is not our intent to justify whether markets are the allocation mechanism to use given some specific scenario; instead, we are interested in improving the performance of a particular market mechanism in responding to uncertainties if it is indeed adopted. More importantly, we would like to achieve this performance improvement without modifying the structure of the market mechanism.

When the characteristics of the problem are examined, one would notice that resource complementarities and uncertainties are the major issues we need to take care of. Both issues could be resolved by introducing some structural changes to the mechanism. For example, aforementioned aftermarkets and decommitment protocols could both solve these issues to some degree. However, just as argued by Sandholm [2002], all these methods are not as direct as addressing these issues directly within the market mechanism. On resource complementarities, combinatorial auctions are the most well-studied mechanism that by definition guarantees only desired bundles are allocated. Of course, this comes with a price: added computational complexities are what one has to pay for the services of combinatorial auctions. In our study, we choose a special combinatorial auction setup in order to balance the benefits and the computational burden. The detail of this will be deferred to subsections right after this.

The main contribution of our paper is the proposal of using time-period aggregation as a way in handling uncertainties. Similar to the resource complementarities, although uncertainties could be handled by various modifications to the mechanisms (like aftermarkets or decommitment protocols), it is not as direct as addressing uncertainties directly within the mechanism. We will defer the discussion on the time-period aggregation methodology to later sections.

3.1 Problem Statement

For illustration purpose, the research is conducted with a particular scheduling problem inspired by the operations of a mega container terminal in mind. In a typical container terminal (see Figure 1 for a schematic view on container port operations), there are *quay-side cranes* (QC) that load/unload containers to/from the ship, *prime movers* (PM) that carry containers around, and *yard cranes* (YC) that move containers between yard (storage space) and prime movers. Each QC operator will be given a job schedule (which contains both loading and unloading jobs), and the operations of QC can be considered as semi-independent: each operator is endowed with a small amount of guaranteed PM and YC resources but has to compete for the additional usage rights to PM and YC so that its jobs can be finished as soon as possible. To reflect the fact that these operators actually plan and act on their own behalves, the resulting resource allocation problem is modeled in a decentralized manner, in which each operator is represented by an agent. From the operator's point of view, it is very important to ensure that operations at QC, PM, and YC are synchronized, i.e., there is no gap between operations. In practice, the processing times at both QC and YC are quite stable and thus can be viewed as almost deterministic; however, due to the congestion on the ground, the transport time by PMs can be highly variable. The need for operation synchronization generates complementary resource needs, while the highly variable PM transport time makes the problem stochastic.

To simplify the resource allocation process, we assume that the planning horizon is partitioned into discrete time periods with uniform length and the number of resources (PM and YC) is kept constant throughout the planning horizon. Agents will be competing for the *rights* to use resources within specified time periods. To address this decentralized problem with complementary resource requirements and stochastic job processing time (at PM only), we propose a special breed of combinatorial auction that is based on the general equilibrium framework.

3.2 A Combinatorial Auction based on the General Equilibrium Framework

Following the definition by McAfee and McMillan [1987], an auction is: "*... a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants.*" In our study, resources are owned by a central owner and agents (QC operators) have to bid for the rights to use resources in the desired time periods. Due to the needs of having to synchronize operations, an agent has to secure a bundle of (resource, time period) tuples that satisfies its plan for job completions. A combinatorial auction is a special class of auction that accepts bids containing bundles. Unfortunately, combinatorial auctions come with hefty computational requirements, thus making its adoption hard. The reason for this is that in a combinatorial auction, the bid matching

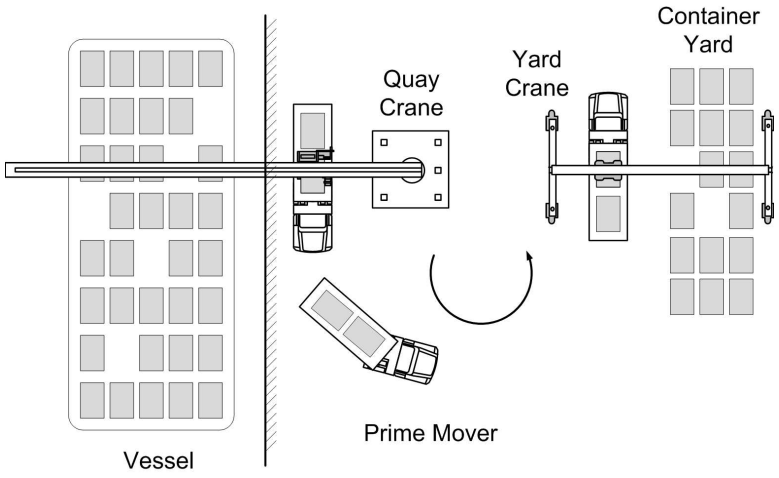

Figure 1: An overview of a typical container terminal operation.

problem, or the *winner determination problem*, is usually modeled as an integer linear program and it is known to be $\mathcal{NP}$-hard. Although a number of efficient implementations are proposed (e.g., see Sandholm [2002]), combinatorial auctions are still not very computationally affordable.

To avoid solving winner determination problems, we assume that unless *clearing* is already feasible (implying that the aggregate demands for all (resource, time period) tuples could be met by supply), the auctioneer would not clear the auction and finalize the allocation. Instead, the auctioneer would adjust the current price for each tuple and announce the adjusted prices back to all agents. If the auctioneer adjusts prices properly, after several iterations agents should come up with bids that are feasible in aggregate. To further simplify the clearing process, we would consider agents as price takers, meaning that they would view market prices as exogenous and not something that would change because of their own actions.

When a price vector induces agents to generate aggregate demands matching supplies for all tuples, a *general equilibrium* is reached. General equilibrium is first proposed by Walras [1874] and has since become a powerful concept in explaining a wide variety of economic phenomena. In recent years, there are also a number of attempts on applying the general equilibrium approach in solving distributed resource allocation problem. For example, the connection between Lagrangian relaxation and combinatorial auctions that are powered by the general equilibrium framework is established by Kutanoglu and Wu [1999]; the decentralized scheduling problem (which violates many theoretical assumptions) is studied by Wellman et al. [2001]; a comparative study on the centralized control and the general-equilibrium-based control is discussed in Ygge and Akkermans [1999].

Generally speaking, to properly implement a combinatorial auction that is based on the general equilibrium framework, there are two major issues that need to be addressed. The first is on how agents generate bids (the formulation of local problems) and the second is on how auctioneer adjusts prices (the coordination between all local problems). These details will be discussed in Section 5.

3.3 Time-Period Aggregation

Combinatorial auctions can effectively handle complementarity in resource requirement. However, to address uncertainties that are associated with job processing times, we need additional devices. In our study, we propose to keep the auction rules unchanged and only tinker with the length of *time frames*. Although time periods could be aggregated arbitrarily, in our study we limit ourselves to only the aggregation of consecutive time periods. The advantage of this type of aggregation is that it does not affect how agents compute their optimal bids, since the aggregation only changes the set of feasible bids.

To present our idea formally, let N be the total number of time periods, and without loss of generality, assume that there is only one resource type r with time-invariant supply capacity M. Define $\mathbf{U}(1)$ to be the set of all feasible bids when the time frame length is exactly one time period. In other words, $\mathbf{U}(1) = (x_1^r, x_2^r, \ldots, x_N^r)$, where x_n^r is the requested quantity for resource r in time period n and it is subject to the capacity constraint, $0 \leq x_n^r \leq M, n =$

$1, 2, \ldots, N$.

When the time frame length is greater than one, say it's y time periods (y must be an integer), it implies that the requested resource quantities for time periods belonging to the same time frame must be identical. This must be true since agents have to bid in time frames. By adding this constraint, we can derive $\mathbf{U}(y)$ as: $\mathbf{U}(y) = (x_1^r, x_2^r, \ldots, x_N^r)$, where x_n^r is subject to both the capacity constraint, $0 \leq x_n^r \leq M, n = 1, 2, \ldots, N$, and the consistency constraint, $x_{(k-1)y+1}^r = x_{(k-1)y+2}^r = \ldots = x_{ky}^r$, $k = 1, 2, \ldots, \lceil \frac{N}{y} \rceil$. Note that if N is not divisible by y, the number of time periods will be less than y for the last time frame.

Our conjecture is that when the job processing time is uncertain, making bidders bid for resources in longer time frames could improve the robustness of the resource allocation plans obtained using exactly the same procedure. The intuition behind this conjecture is that longer time frame length leads to more redundancy in the final allocation (by bidding in time frames, an agent is forced to take additional resources in some time periods), and this redundancy allows agents to flexibly adjust its allocation plan if necessary. In Section 4, we provide a close-form validation for our idea by using a stylized two-stage inventory model. A more realistic and complicated scenario is computationally analyzed in Section 5.

4 Time-Period Aggregation in a Simple Inventory Model

4.1 Important Assumptions

To confirm our conjecture in an intuitive manner, we study the effectiveness of time-period aggregation in a simple, two-stage inventory model. In this model, the planning horizon is fixed to two time periods for all agents, and for agent i, it is given a forecasted demand D_n^i for each time period n (to simplify the notations, exponent i is dropped for the rest of the section). To satisfy one unit of demand in time period n, an agent has to acquire one unit of resource in the same time period. Unsatisfied demands will be backlogged at a cost of B per time period.

Resources in both time periods are dispatched from a central office according to agents' requests. For resources in time period n, the central office will set the unit price p_n that is payable by all agents. Note that if the time frame is defined to contain both time periods, the prices in both time periods must be identical (to reflect the fact that agents are bidding on a single time frame instead of two independent time periods). Resources cannot be stored and unused resources for a particular time period are lost. As all agents are cost minimizers, they might want to postpone the servicing of demands if the price difference is significant enough. Unfulfilled demand incurs a penalty that is far greater than the resource prices to ensure that all agents will try their best in fulfilling their respective demands.

The deterministic version of the problem is trivial to solve; unfortunately, the "real" demand each agent receives will remain uncertain until after the resource allocation is finalized. To illustrate the impact of time-period aggregation, agents will adopt an identical deterministic procedure in generating their bids. While this two-stage inventory model is simple, it provides some valuable insights for us to infer what might happen when the problem becomes more complicated.

We make the following assumptions in our analysis:

1. D_n is determined arbitrarily. The real demand, $\hat{D}_n$, following certain distribution, is realized only after agents are committed to the resource allocation plan.

2. For simplicity, assume that it will never be beneficial to purposely postpone the demand satisfaction. In other words, $p_2 + B > p_1$, and the optimal policy when the time frame length is one will be to simply order D_n for time period n.

3. There is no hard limit on the resource supply. This assumption is not really necessary to obtain our conclusion; the analysis only requires that the resource allocation monotonically increases with rising resource demand (this is already guaranteed since the pricing of the resource is uniform). Nonetheless, this assumption would significantly simplify our analysis.

4.2 The Analysis

Since unfulfilled demand dominates our decision making criterion, the main focus of our analysis is the expected volume of unfulfilled demand under different time frame lengths. The major analytical result we would like to demon-

strate is the negative correlation between time frame length and the expected volume of unfulfilled demand, which is highlighted in the following Lemma:

Lemma 4.1 *The expected volume of unfulfilled demand will be non-increasing as the length of the time frame increases.*

Proof. Let l be the length of the time frame and (x_1^l, x_2^l) be the corresponding amount of requested resources in time periods 1 and 2. As described earlier, for $l = 2$, x_1^2 must equal x_2^2 since both time periods now belong to the same time frame. Since the expected volume of unfulfilled demand depends only on (x_1^l, x_2^l), we can denote its value as $E(u|x_1^l, x_2^l)$. Obviously, the value of $E(u|x_1^l, x_2^l)$ will be non-increasing in either x_1^l or x_2^l.

From Assumption 2, the optimal policy for $l = 1$ is to match the demand in each time period, i.e., $x_1^1 = D_1$ and $x_2^1 = D_2$. For $l = 2$, the optimal policy will depend on the relationship between D_1 and D_2, and can be summarized as follows:

$\underline{\mathbf{D_1} = \mathbf{D_2} = \mathbf{d}}$. The optimal policy is to order d units for the time frame (i.e. $x_1^2 = x_2^2 = d$). In this case, the optimal policies for both $l = 1$ and $l = 2$ will be identical ($x_1^l = x_2^l = d$), thus the expected volume of unfulfilled demand will be the same as well.

$\underline{\mathbf{D_1} < \mathbf{D_2}}$. The optimal policy is to order D_2 units. Compared to the optimal policy for $l = 1$, the agent is asking for more resources in the first time period. As a result, $E(u|x_1^2, x_2^2) \leq E(u|x_1^1, x_2^1)$.

$\underline{\mathbf{D_1} > \mathbf{D_2}}$. There are two sub-cases here. When $(p_1 + p_2) < B$, the optimal policy is to order D_1 units; otherwise order $\lceil (D_1 + D_2)/2 \rceil$ units (since backlogging is allowed). Note that the agent is indifferent if $(p_1 + p_2) = B$. This can be derived by considering the minima for agent's (deterministic) cost function:

$$(p_1 x_1^2 + p_2 x_2^2) + B(D_1 - x_1^2), \tag{1}$$

such that $x_1^2 + x_2^2 \geq D_1 + D_2$ and $x_1^2 = x_2^2$. These two cases are further discussed as follows. (For notational simplicity, for the rest of the proof we will simply assume that $D_1 + D_2$ is even and drop $\lceil \cdot \rceil$.)

- Order D_1 units. Compared to the $l = 1$ case, $x_1^2 = x_1^1 = D_1$, but $x_2^2 = D_1 > x_2^1 = D_2$. Therefore, $E(u|x_1^2, x_2^2) \leq E(u|x_1^1, x_2^1)$.

- Order $(D_1 + D_2)/2$ units. For both $l = 1$ and $l = 2$ cases, the total amount of requested resource is the same ($x_1^l + x_2^l = D_1 + D_2$ for $l = 1, 2$), however, for $l = 2$, the policy orders less in time period 1 and more in time period 2. To be more specific, the quantity of $(D_1 - D_2)/2$ is shifted from time period 1 to 2 for $l = 2$. This shift will reduce expected unfulfilled demand since additional resource in time period 2 will help to reduce unfulfilled demand whenever $\hat{D}_2 > D_2$ while unmet demand in time period 1 can always be backlogged to time period 2.

In all the above instances, we have $E(u|x_1^2, x_2^2) \leq E(u|x_1^1, x_2^1)$, thus the lemma is proved. ■

5 Solution Approach

Following the introduction given in Section 3.2, we will now formally state the combinatorial auction model with time-period aggregation.

We use a multi-iteration combinatorial auction in which each agent is allowed to submit only one *combinatorial quantity bid* (agents are assumed to be price takers and they submit their desired quantities assuming that they will be charged at the announced prices). We assume that resources are centrally owned by the auctioneer, and the exclusive rights to use the resources in each and every time frame are allocated to agents through the bidding process. The planning horizon is discretized into time periods with uniform length, however, the resource usage rights are allocated in time frames, which have lengths that are integer multiples of time periods. The length of time frame, unit price for each (resource, time frame) tuple (for the rest of the paper, we will just use "tuple" for brevity), and the total available resource supply are known to all agents.

Compared to the centralized planning paradigm, our proposal is two-tier in nature, as depicted in Figure 2. At the lower tier, every agent receives updates on unit prices for all tuples at the beginning of each round and generates a single bid containing quantities requested for all tuples. Agents submit their bids back to the auctioneer and if the

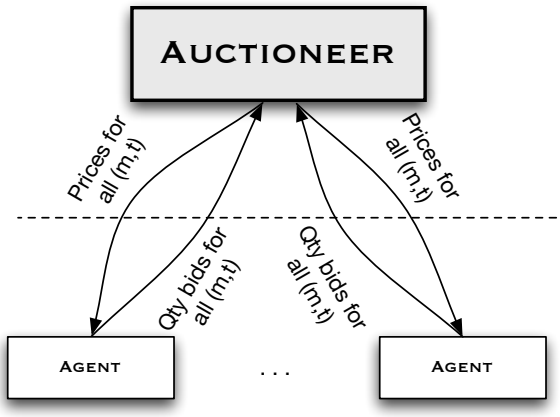

Figure 2: A two-tier structure for solving decentralized resource allocation problem.

stopping criterion (to be defined later) is not met, prices for all tuples will be adjusted by the auctioneer according to the net demands at the upper tier (difference between aggregate demand and supply). In general, prices are lowered for tuples with negative net demands and prices are increased for tuples with positive net demands. After price adjustment, all agents are notified of the new prices for another round of bidding. A feasible solution is discovered when the demand for each tuple does not exceed the total resource supply. Empirically speaking, this iterative process would continue until the specified stopping criteria are met. The stopping criteria can be flexibly set by the user and in most cases, it can be either time-based (stop after computational time expires), solution-based (stop after number of feasible solutions exceeds certain threshold), or convergence-based (stop after the difference between two consecutive price vectors falls below certain threshold). Depending on the scenario we are studying, these criteria can be combined arbitrarily to create a composite stopping criterion.

The lower-tier problem – *bid generation*, and the upper-tier problem – *price adjustment*, are described in detail in the following two subsections.

5.1 Bid Generation

As described in Section 3, bidders in the system are assumed to be price takers. Thus, their bid generation problems, given tuple prices, can be modeled and solved as independent optimization problems. Following the problem statement in Section 3.1, we now formally provide an abstract model for the bid generation problem.

For every agent, a job list containing both unload and load jobs is assigned, with all unload jobs required to be processed before load jobs. An unload job requires a sequence of resources R_1, R_2, and R_3 (representing QC, PM, and YC respectively). A load job, on the other hand, requires a sequence of resources R_3, R_2, and R_1. No job can queue for resources in between processes (in other words, we must ensure that next process can start immediately after the current process ends), and there is a strict precedence constraint for all processes at resource type R_1 (i.e., job $n+1$ cannot start until job n is finished). Jobs might have different (and stochastic) processing times at each resource. From the description of this bidder model, we can see that this problem is both complementary and (potentially) stochastic.

Every agent i has an arrival time a_i, a due time d_i, a makespan cost rate per time period ϕ_i and a tardiness cost rate per time period γ_i. Let the time agent i finishes its job list be l_i, then the makespan of agent i is defined as $M_i = l_i - a_i$. Let the unit price and the requested quantity of R_k in time frame t be p_{kt} and D_{kt} respectively. The cost function of agent i is then:

$$\sum_{k,t} D_{kt} p_{kt} + M_i \phi_i + \gamma_i \max(0, M_i - d_i). \tag{2}$$

Equation (2) can be minimized by solving an integer programming model, however, for large-scale problems with hundreds of jobs (as studied in this paper), this approach is computationally prohibitive, as observed by Lau et al. [2007].

Alternatively, we apply a local search technique called *relax and repair*, which consists of two steps. The first step is the relax phase in which the multi-period problem is relaxed to a single period problem and a common resource

level that minimizes total cost is identified. In other words, for each resource type k, we let $D_{kt} = D_k^*$ for all t, such that (2) is minimized. This search procedure is simply hill-climbing in the landscape of all the combinations of D_k's. Note that a by-product of this hill-climbing search is a makespan matrix that contains the makespan value for every resource combination, regardless of resource prices. Each entry in the matrix only needs one computation, and can be looked up subsequently when prices change between iterations. The second phase is the repair phase in which the solution from the relax phase is improved by local search. The local search algorithm looks at one tuple at a time, and performs steepest descent search on the quantity of the tuple until a local optimum is reached. This is repeated for all tuples. Relax and repair is shown to be very efficient and effective [Lau et al., 2007], and the computational requirement grows only linearly with the number of resources and time frames.

5.2 Price Adjustment

For resource type R_k in time frame t: let C_{kt} be the available resource supply, p_{kt}^r be the unit resource price in the r^{th} iteration, D_{ikt}^r be the demand from agent i in the r^{th} iteration, D_{kt}^r be the aggregated demand over all agents (i.e., $D_{kt}^r = \sum_i D_{ikt}^r$). Furthermore, let ρ_k be the reserve price for R_k. With these definitions, price adjustment process can then be described as:

$$\begin{aligned} p_{kt}^{r+1} &= \max\left\{\rho_k, p_{kt}^r + s^r\left(D_{kt}^r - C_{kt}\right)\right\}, \\ s^r &= \alpha \frac{\sum_{k,t} C_{kt} p_{kt}^r / \sum_{k,t} C_{kt}}{\sum_k \sqrt{\sum_t \left(D_{kt}^r - C_{kt}\right)^2 / T}}, \end{aligned} \tag{3}$$

where α is a constant over iterations and is in $[0, 2]$. Interpreted intuitively, the price is adjusted adaptively according to the difference between the capacity and the aggregated demand. Our price adjustment formula is adopted from Fisher [1985], and it is proved to be practically effective by Kutanoglu and Wu [1999]. Compared to the version originally proposed by Fisher [1985], the difference between upper and lower bounds is removed from the numerator and we also insert the average price of all resources over all time frames. This modification has helped the convergence empirically, since larger step size at higher price allows the auctioneer to balance the excess demand/supply faster, while smaller step size at lower price enables the auctioneer to take finer steps in approaching the equilibrium instead. The parameter α is an user-specified parameter to further control the price adjustment process. A more detailed discussion on the selection of α can be found in Lau et al. [2007].

Each auction is assumed to last for K rounds. All feasible allocations are stored, and the one with smallest expected makespan (which is the sum of expected makespan reported by individual agents) will be selected as the final allocation.

5.3 Reducing Price Oscillation

During our initial experiments we discovered that under certain time frame lengths, demands in adjacent time frames may be negatively correlated, as demand moves from the more expensive time frame to the cheaper neighbors. This behavior could easily result in a cycle as demand is shifted back and forth between two time frames. To reduce the possibility that demands oscillate between adjacent time frames, we dampen the price adjustment for selected time frames if such negative correlations are detected.

Our idea of dampening price adjustment is to first detect the existence of potential price oscillation and then identify time frames that are related to this oscillation. To achieve this, we keep track of the *sign switching* events, i.e., when the net demand changes from negative to positive or vice versa. Whenever this is detected, we will check the sign switching status in the adjacent time frames and if they are negatively correlated to the current time frame (i.e., the changes are in the opposite direction), a dampening factor is applied. This dampened price adjustment process can break the price oscillation since agents can no longer simply move from one time frame to another. This is an easy addition to the original price adjustment process since we only need to keep track of the aggregate demand from the last iteration. Our proposal is limited in that it only looks at adjacent time frames and handles every resource type independently, however, its simplicity makes it an ideal first-pass filter for countering price oscillations.

Formally speaking, a *sign switching* happens when the net demands (total resource demand - resource supply) for R_k in iterations r and $r-1$ change signs. For each time frame t^* that has a sign switching for R_k, the adjacent time frames t^*-1 and t^*+1 (if they exist) are checked to see whether these are *influencing time frames* to t^* with respect to R_k. An adjacent time frame t' is an R_k-influencing time frame of t^* if it satisfies the following conditions: (1) it has not

been previously flagged as an R_k-influencing time frame for another time frame, and (2) its net demand change for R_k is in the opposite direction of the net demand change of t^*. For resource R_k, let the set containing t^* and its influencing time frames be $\delta(t^*, k)$. $\delta(t^*, k)$ has at least one element t^* and can have at most three elements $(t^*, t^* - 1, t^* + 1)$. To reduce demand oscillation among time frames in $\delta(t^*, k)$, an additional damping factor $\beta(t^*, k) > 1$ is applied to the price of R_k for all time frames in $\delta(t^*, k)$. The damping factor is defined as:

$$\begin{aligned} \beta(t^*, k) &= \left| \frac{D^{r-1}_{kt^*} - C_{kt^*}}{(D^{r}_{kt^*} - C_{kt^*}) - (D^{r-1}_{kt^*} - C_{kt^*})} \right| \\ &= \left| \frac{D^{r-1}_{kt^*} - C_{kt^*}}{D^{r}_{kt^*} - D^{r-1}_{kt^*}} \right|. \end{aligned} \tag{4}$$

Since sign switching happens at t^*, either $D^r_{kt^*} < C_{kt^*} < D^{r-1}_{kt^*}$ or $D^r_{kt^*} > C_{kt^*} > D^{r-1}_{kt^*}$ must hold. For both cases, we can see that $\beta(t^*, k) < 1$. The intuitive explanation for Equation (4) is that price adjustment should be dampened more if the price oscillates more violently.

The step size α for elements in $\delta(t^*, k)$ (t^* refers to any time frame that has sign switching) is set to $1.5\beta(t^*, k)$. For time frames and resource types that do not belong to any set $\delta(t^*, k)$, the step size is set to 1.5.

5.4 Performance Evaluation

Up to this point it is assumed that agents generate bids in a deterministic environment. However, when a deterministically generated resource allocation plan is used in an uncertain environment, the performance of such plan is expected to deteriorate significantly. To help individual agents cope with such concerns, the mechanism designer could adjust the length of time frames and hope that even uninformed agents could have satisfactory performances in uncertain environments.

To measure the effectiveness of such time-period aggregation methodology, several important types of performance indices should be tracked. First performance index we track is the variability of the performance; that is, how variable the performance of the allocation plan would be if it is used in uncertain environments. Second performance index we track is the average performance of the allocation plan; more specifically, the resource allocation plan is used in sufficient numbers of realizations of the uncertain environment, with resulting performance averaged.

The above two performance indices are only measurable if the resource allocation plan could stay feasible regardless of the realized uncertainty. In cases where feasibility of the plan cannot be guaranteed, we will need to come up with appropriate proxies for measuring variabilities and averages. Such proxies will be dependent on the problem context and we will introduce them where appropriate.

6 Test Case: Container Port Operation

The first test case we study is the motivating example of the container port operation that is mentioned in Section 3.1. In the next section, we will investigate the more general job shop scheduling problem.

6.1 Experiment Setup

In all instances we studied, there are four agents, each representing a ship and the QC operator serving it. The implication of this is that every agent will be endowed with exactly one unit of R_1 (QC), and no additional unit would be available. For R_2 (PM) and R_3 (YC), each agent is initially assigned two units and one unit respectively[1], and there are ten units of R_2 and eight units of R_3 available for bidding in the market. The initial prices for R_2 and R_3 are \$50/hour and \$30/hour respectively.

All bidders have the identical relax-and-repair bid generation strategy as described in Section 5.1, and all have makespan cost rate of \$100/hour and tardiness cost rate of \$500/hour. All bidders have ten unload jobs followed by ten load jobs.

To provide necessary granularity, we assume that the length of each time period is one minute. The processing times at R_1 and R_3 are assumed to be deterministic, and are fixed at three and six minutes respectively (QC and

[1] Since each agent has initial endowment on R_2 and R_3, even it gets no additional units from the market, its job list can still be finished. However, the resulting makespan might be prohibitively long.

YC almost never fail and their processing times are fairly stable). The processing time at R_2 is stochastic and only realized after the conclusion of the auction process (the processing times of PM are quite volatile since factors like road congestion, movement speed, and route selection all have direct impacts). However, its mean processing time is known to the bidders as μ_j for job j. It is assumed that the processing time at R_2 for job j follows a discrete uniform distribution that has twenty possible values, $\{0.525\mu_j, 0.575\mu_j, \ldots, 1.475\mu_j\}$, i.e., $(0.525 + 0.05i)\mu_j,\ i = 0, \ldots, 19$. The standard deviation of this distribution is thus:

$$\sqrt{\frac{\sum_{i=0}^{19}((0.525 + 0.05i)\mu_j - \mu_j)^2}{20}} = 0.288\mu_j.$$

Our additional experiments showed that changing the distribution range (and hence the variability) does not qualitatively change the conclusions obtained.

Eleven problem sets have been designed based on real-world data, each with different R_2 mean processing times. To better approximate the real-world operational conditions, we assume four different agent arrival and due time patterns for each problem set. The first agent is assumed to always arrive at the beginning of the first time frame and has a due time that is $f/2$ hours after arrival, where f is the number of unload jobs. There are six different time frame lengths considered throughout the experiment: 15, 30, 45, 60, 90 and 120. We stopped at 120 since further increases in the time frame length might reduce the problem to a single-time-frame problem.

6.2 Results and Discussions

As discussed in Section 5.4, the effectiveness of the time-period aggregation approach could be assessed by two measures: 1) the robustness of the resource allocation plan (how variable is the result), and 2) the performance of the allocation plan under uncertainty (exact measures are to be defined later).

In the following analysis, we repeatedly refer to the performance within a deterministic and a stochastic environment, and their definitions are formally provided here. For the performance in a deterministic environment, we refer to the performance achieved by the allocation plan generated with deterministic agents (agents' realizations of the resource requirements match their forecasted requirements, and bids are generated by solving deterministic problems). For the performance in a stochastic environment, we refer to the expected performance of the same allocation plan when the processing time of R_2 is randomized. The expected performance is estimated by running 5,000 Monte Carlo simulations on the R_2 processing times.

6.2.1 Impact of Time-Period Aggregation on Robustness

The robustness of the resource allocation plan is measured by the *coefficient of variation* (CV) of makespans and agent costs. CV is a normalized measure for the variability and is computed by dividing the standard deviation by the mean. Since CV is dimensionless and scale invariant, it allows us to compare variability across scenarios consistently.

The CV of both makespans and agent costs are plotted in Figure 3, and the resource allocation plans are indeed more robust when the time frame lengths are longer. This is consistent with our conjecture that longer time frames could stabilize the unexpected fluctuations in the processing times. However, as the figure suggests, the incremental benefit wanes as the time frames grow longer.

Another indication that longer time frames stabilize unexpected fluctuations is the difference between average makespans under deterministic and stochastic assumptions (please refer to the beginning of this section for their respective definitions). As shown in Figure 4, the gap between the deterministic case and the stochastic case narrows as time frame length increases.

6.2.2 Impact of Time-Period Aggregation on Performance

We measure the performance of the resource allocation plan by (1) the average makespan of all agents and also (2) the average agent cost (which includes tardiness cost and resource cost besides makespan cost). Makespans obtained under deterministic and stochastic cases can already been seen in Figure 4. Agent costs under deterministic and stochastic cases are plotted in Figure 5 and exhibit similar patterns.

As Figures 4 and 5 demonstrate, when the time frame length increases, the performance of deterministic allocation plan deteriorates. This is because the deterministic model grows more constrained as time frame length increases,

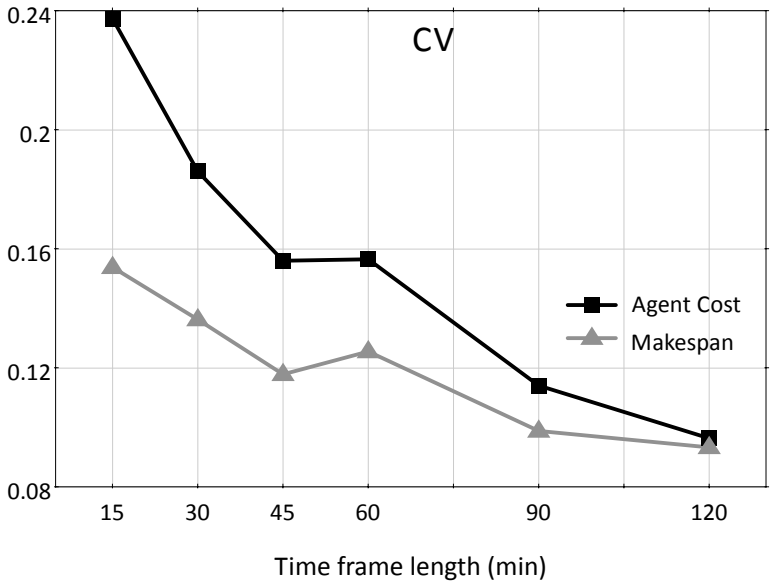

Figure 3: CV for makespans and agent costs.

resulting in worsening options for bidders[2]. However, the improving trend of the performance in the stochastic case suggests that for deterministically generated plans, longer time frames could indeed be helpful in stabilizing the fluctuations caused by unexpected disturbances. Although we are only showing the performance averages, this observation is generally true in all scenarios we tested (each scenario differs in agent's arrival times and deadlines).

6.3 Summary

The results discussed in this section empirically confirm the effectiveness of the time-period aggregation for job scheduling at a container port. Besides the benefits of increased robustness and better performances within a stochastic environment, the aggregation of time periods also allows individual agents to reduce the sizes of their respective optimization problems (aggregation reduces number of choices). This feature is important since even agents with limited computational power could benefit from the approach.

7 Test Case: Job Shop Scheduling Problem

The container port scheduling problem studied in Section 6 is a specialized job shop scheduling problem (JSSP). To demonstrate that our approach is generally applicable, we also look at a class of generic JSSP from the literature in this section.

Traditional JSSP is solved assuming centralized control; however, as elaborated earlier, one important focus in our study is to solve problem with decentralized control, and more specifically, the resources from a central pool are distributed via market mechanisms. To decentralize the classical JSSP, we assume that there are multiple agents and each of them is endowed with a list of jobs. All agents need to acquire resources via auctions in order to finish their respective jobs. Of course, it is this resource allocation process that exposes agents to the risk of not having enough resources when they face uncertainties. Thus, we will apply the same time-period aggregation approach and measure its effectiveness in achieving robust resource allocation.

The major challenge of JSSP is the fact that the resource tuple (which consists of machine and time period) is unique and has a capacity of one. This fact, together with the constraints that operations must be executed in certain order, implies that the resource allocation might become infeasible if the durations of operations or job arrivals are

[2]It might be tempting to argue that the performance deterioration would be monotonic in the time frame length. However, for a pair of lengths l_1 and l_2 where $l_1 < l_2$ and their GCD = 1, no subset relationship could be established and thus we could not conclude which case would result in a better allocation.

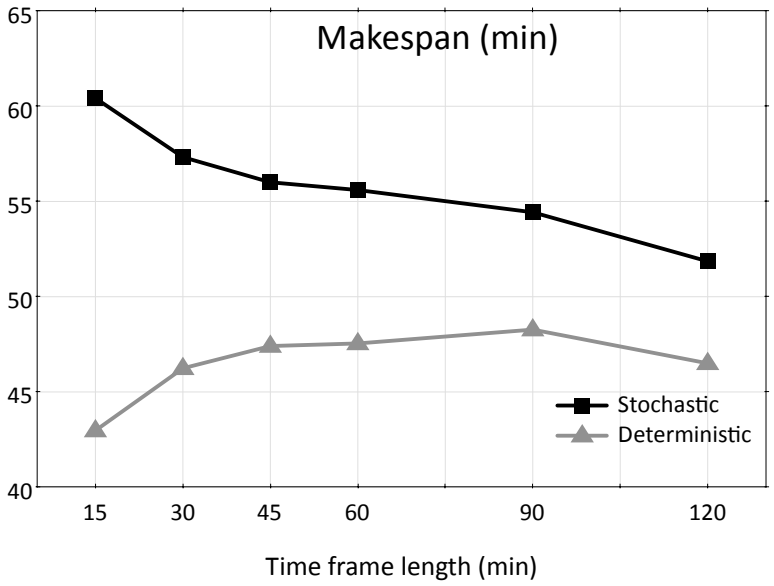

Figure 4: Average makespans in deterministic and stochastic cases.

subject to uncertainties. As noted in Section 5.4, this will affect how we measure the effectiveness of our approach. This issue is discussed in Section 7.2.

7.1 Problem Description

JSSP is a classical optimization problem that aims at assigning resources to jobs at different times so as to avoid resource conflict and minimize makespan. Many variations of JSSP exist, and here we will focus on a decentralized version that is derived from a basic JSSP formulation.

We assume that there are N agents in the problem and K_n jobs are assigned to agent n. Every job $j_{n,k}$ (k^{th} job belonging to agent n) contains M unique types of operations that need to be completed in a pre-determined order. Every operation type needs to be handled by one dedicated machine (this implies that there are exactly M machines in the problem). All machines are owned by an auctioneer and the usage rights of all machines at different time frames are allocated via the same auction mechanism as described in Section 5. Although agents bid for resources in "time frames", they generate plans in time periods.

The structure of this decentralized JSSP follows the general two-tier structure as depicted in Figure 2. At the lower tier, an agent needs to solve a modified version of the basic JSSP that considers resource prices in addition to its makespan. The required amount of resources at every time frame will constitute the quantity bid from this agent and be sent to the auctioneer. At the upper tier, the auctioneer will collect quantity bids from agents and executes very similar price adjustment process.

The bid generation problem and the price adjustment process are described in detail in the following two subsections.

7.1.1 Bid Generation

JSSP is notoriously difficult to solve; in fact, besides being $\mathcal{NP}$-complete, it is also among the most challenging $\mathcal{NP}$-complete problems in practice. As such, almost all recent studies on JSSP have been on approximation algorithms that are effective in various different settings. In our case, we construct the bid generation algorithm by extending one of the most well-known approximation algorithm: the shifting bottleneck (SB) algorithm proposed by Adams et al. [1988]. The primary change that we have implemented is the inclusion of resource prices in agent's objective function.

The SB algorithm tries to iteratively identify the *critical machine* among unscheduled machines[3] and include it in

[3]The criticality of a machine m, $v(m)$, is obtained by solving an one-machine scheduling problem for machine m that minimizes maximum

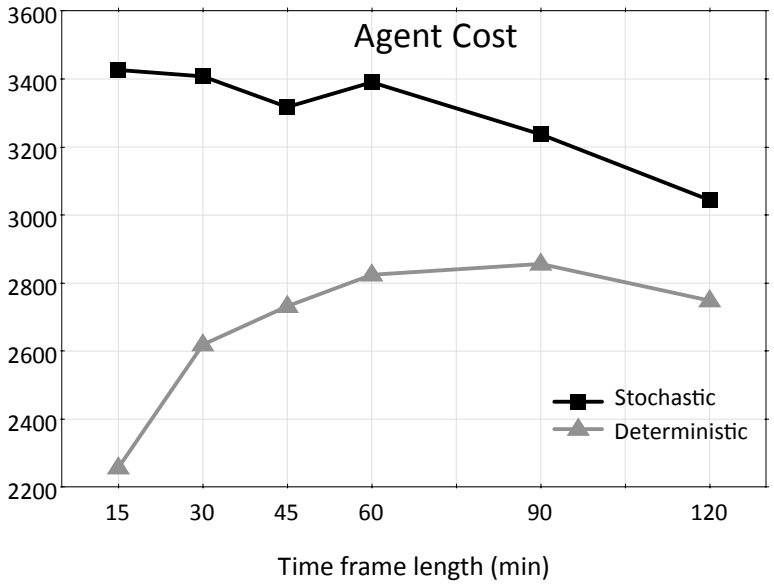

Figure 5: Average agent costs in deterministic and stochastic cases.

the set of sequenced machines. The sequence at all identified critical machines are then re-optimized one by one. The algorithm terminates when all machines are included in the set of sequenced machines.

Resource costs can not be directly considered in the original SB algorithm, however, a greedy local search algorithm can be implemented to search the space of resource allocations. The idea is very simple, start by assuming that the agent holds all (resource, time frame) tuples; for each and every tuple, remove that tuple and re-solve the JSSP by using the same SB algorithm. If a better solution is obtained, it will replace the current standing solution. The loop will continue until no improvement can be made after a full cycle over all tuples.

7.1.2 Price Adjustment

The price adjustment process is almost identical to the one described in Section 5.2. The only differences are that we remove the *sign switch* damping as described in Section 5.3 and set the step size (α) to 1. These changes are proposed since they can effectively handle the case where every resource (machine) only has one unit available for allocation.

7.2 Performance Evaluation under Uncertainty

In our study, we implement a time-based stopping criterion: the process will keep running for 100 iterations then terminates. The best feasible solution found will be utilized to finalize the allocation of tuples.

Operation processing times are uncertain, however, mean operation processing times are known to agents and agents will generate bids based on them. All operation processing times are realized after the resource allocation is finalized. With this information and also the allocation result, agents will need to generate the *real* schedule. Of course, because of these uncertainties, the received allocation might not be able to satisfy all operations. As such, instead of focusing on makespan (makespan cannot be defined if not all operations can be completed), we assume that agents will try to maximize the number of operations they can finish.

In other words, the number of completed operations is viewed as the proxy to the allocation performance. With this proxy introduced, we will similarly measure its variability and average when the resource allocation plan is used in a number of generated realizations.

lateness. The machine with largest $v(m)$ is defined as the critical machine.

7.3 Experimental Setup

Two agents are included in our experiments. Due to the complexity of the bid generation process and the structure of JSSP, including more agents will cause solution time to increase significantly. Nonetheless, even with just two agents, the impact of decentralization and the benefit of time-period aggregation can still be clearly observed.

To evaluate the performance of increasing time frame lengths, 10 randomly generated problem instances, each with 15 jobs and 3 operations per job, are solved. For each of these 10 problem instances, we evaluate the performance of the resource allocation (following the performance metrics described in the previous subsection) for time frame lengths of 30, 60, 90 and 120 time periods.

For each problem instance, the mean processing times of all operations are randomly drawn from the discrete uniform distribution between 1 and 20 (time periods). When we evaluate the performance of allocation plan under uncertainties, we generate real operation processing times from the discrete uniform distribution of $(\max(1, 0.5m_o), 1.5m_o))$, where m_o stands for the mean processing time of operation o. For each problem instance, 10 realizations are generated for evaluation. The initial prices of all resource tuples are set to \$30 in all instances that we study.

7.4 Results and Discussions

As discussed in Section 7.2, the primary performance metric we plan to measure is the number of finished operations. For all scenarios that we study, two agents in the problem are given 15 jobs that contains 3 operations each; the upper bound on the number of finished operations is thus 90.

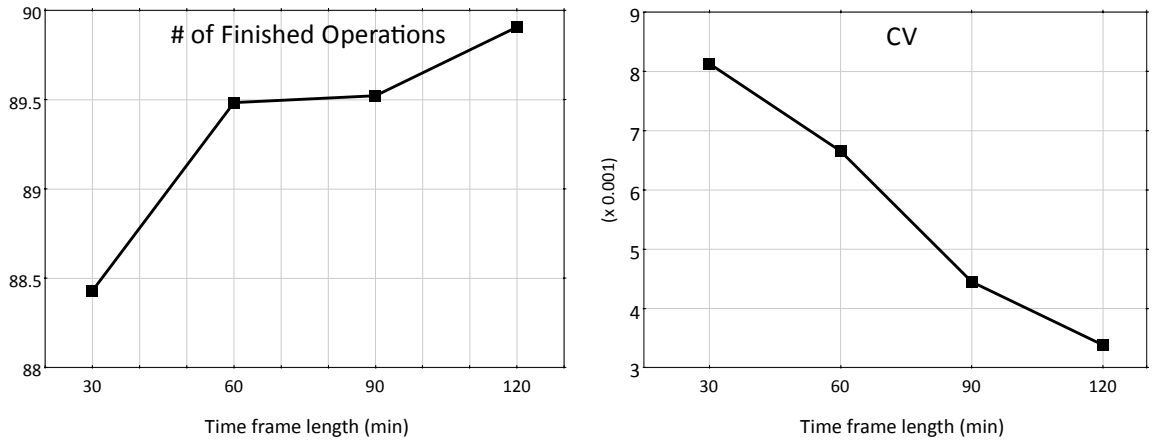

Figure 6: Average and CV on the number of finished operations.

Averages and CV of the number of finished operations under different time frame lengths are plotted in Figure 6. For each time frame length, the average is computed from the 100 shared problem instances (10 scenarios, 10 realizations per scenario are shared by all time frame lengths). As time frame length increases, we can see that the average increases while the CV decreases. Statistically speaking, the improvements achieved by increasing the time frame length from 30 to 60 and 90 to 120 are both significant with p-values equal 0.0002 and 0.0055 respectively. The slight improvement from 60 to 90 is not statistically significant.

Compared to the test case on container port operations, the JSSP is more general and difficult. Nonetheless, our computational study suggests that the aggregation of time periods works equally well even in this more challenging class of problem.

8 Conclusions

An issue with allocating resources via auctions is the robustness of the resulting allocation under uncertainties. The resource allocation plan with finest granularity leaves little room for change and might be particularly sensitive to uncertainties. We evaluate our conjecture that increasing time frame length in which resources are auctioned off can accord stakeholders increased robustness in their solution qualities.

In our numerical case studies, we assume that agents generate their bids deterministically. The resulting resource allocation plans are then used in large numbers of realizations from an uncertain environment. This same process is repeated for each and every time frame length, and we measure the resulting average and variability. For both the container port operations and the classical job shop scheduling problem, we show that as time frame length increases, the associated variability decreases without sacrificing averages in most cases. However, these benefits eventually tail off once the time frame length exceeds certain threshold.

Our study demonstrates a simple yet effective way in dealing with uncertainties. Since our idea is based on risk pooling, it implies that we might be able to achieve similar or even better performance improvement if the aggregation is conducted beyond just consecutive time periods. For example, one could explore the bundling of several related resources together or even the bundling of arbitrary resource tuples.

Another interesting issue not fully explored in this paper is the identification of aggregation factor. We have demonstrated the benefits in increasing time frame length; however, we have not explored how to identify the *optimal* time frame length. A straightforward approach for approximating the optimal time frame length will be to gradually increase the time frame length until the cost (higher agent costs or longer makespan) overtakes the benefit (reduced variability or shorter makespan). However, the definitions of costs or benefits depend on the problem context and will have to be determined on a case-by-case basis.

9 Acknowledgements

This work was supported in part by the Wharton-SMU Research Centre at the Singapore Management University under Grant C220/MSS7C008. We also thank anonymous reviewers for their comments.

References

J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401, 1988.

J. Q. Cheng and M. P. Wellman. The WALRAS algorithm: A convergent distributed implementation of general-equilibrium outcomes. *Computational Economics*, 12:1–24, 1998.

S.-F. Cheng, J. Tajan, and H. C. Lau. Distributing complementary resources across multiple periods with stochastic demand: Hedging via time frame aggregation. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 373–379, 2008.

P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

E. Crawford and M. Veloso. Mechanism design for multi-agent meeting scheduling. *Web Intelligence and Agent Systems*, 4(2):209–220, 2006.

M. L. Fisher. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15(2):10–21, 1985.

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

H. C. Lau, S.-F. Cheng, T. Y. Leong, J. H. Park, and Z. Zhao. Multi-period combinatorial auction mechanism for distributed resource allocation and scheduling. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 407–411, 2007.

R. P. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25(2):699–738, 1987.

T. Sandholm. Algorithm for optimal winner determination in combinatorial auction. *Artificial Intelligence*, 135(1–2): 1–54, 2002.

I. E. Sutherland. A futures market in computer time. *Communications of the ACM*, 11(6):449–451, 1968.

J. M. Vidal. A method for solving distributed service allocation problems. *Web Intelligence and Agent Systems*, 1(2): 139–146, 2003.

L. Walras. *Elements of Pure Economics*. 1874. English translation by William Jeffé, 1954.

M. P. Wellman and P. R. Wurman. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24:115–125, 1998.

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.

F. Ygge and H. Akkermans. Decentralized markets versus central control: A comparative study. *Journal of Artificial Intelligence Research*, 11:301–333, 1999.