Modeling Task Allocation with Time using Auction Mechanisms

A thesis presented

by

Swara S. Kopparty

To

Applied Mathematics

in partial fulfillment of the honors requirements

for the degree of

Bachelor of Arts

Harvard College

Cambridge, Massachusetts

March 30, 2012

Abstract

The efficient allocation of tasks to groups of agents has been studied extensively in various scenarios. In this thesis we look at the task allocation problem with the added parameter of multiple times at which the job may be completed. We consider both the settings where there is a single task to be done and also where there are multiple subtasks of a composite task to be completed. We model the two settings as an auction where workers submit bids for completing individual tasks or bundles of tasks at different times. In the combinatorial auction setting, we entertain the idea that workers may consider groups of subtasks to be complementary, and we adjust bids on bundles of tasks using a measure of complementarity. We propose a mechanism for determining the optimal allocation in this setting. Finally, we simulate our mechanism and show that it can effectively take advantage of the complementarity of subtasks.

Contents

Acknowledgements

I am deeply indebted to my thesis adviser, Professor Yiling Chen, for her immense guidance, constant encouragement, and numerous helpful suggestions. I would like to thank Professor David Parkes, for being instrumental in my introduction to the area of Economics and Computer Science. Many thanks to Haoqi Zhang and David for offering initial discussions. I want to thank my brother, Swastik Kopparty, for being a constant source of inspiration. Lastly, I want to thank my family for providing me with the utmost motivation and moral support throughout the development of this thesis.

1 Introduction

The question of efficient allocation of resources is of great practical significance with applications in a variety of consumer-producer settings. The problem of task allocation has been studied extensively in many situations. However, the setting where the completion time of the job is a tunable parameter has not been studied in detail. In this thesis, we examine the concept of optimal task allocation with an additional parameter of time. We consider the case of the manager as a consumer who has access to a collection of workers. The manager needs a certain job to be completed within a maximal amount of time, and the job has the possibility of being completed at earlier times, at the cost of making the workers work harder. Why is this an interesting twist? Time can be thought of as a measure of efficiency. It seems very natural that when a manager is looking to complete a job, a common necessity is for the job to get done as quickly as possible. However, we can also consider time to be a concrete representation of a measure of the quality of the task. So, while we consider time to be a measure of efficiency throughout this thesis, the introduction of the additional parameter of time could be substituted with any measure of the quality with which the task gets done. The mechanisms we explore in this paper, then, can be used to solve situations with different measures of quality.

There are several questions that must be considered. How should the task be allocated so that it is completed as quickly as possible? How should the task be allocated so that the manager has to pay as little as possible? How can we ensure that the workers work sincerely? We consider a basic model where there is a single manager, and he must choose amongst a group of workers and assign the task in a way that balances all the above factors. In order to concisely capture this problem, we say that workers report bids for completing the task at every possible time. These bids are meant to represent the workers' self-evaluations of their skills and adeptness at completing the task at hand within each period of time. Let us consider an example: say there is a manager, and the task is to develop some software before a certain deadline. Suppose the deadline is time $T = 5$, but the manager would also prefer that the task be done before the deadline, namely at time $T = 1, 2, 3$ or 4 . Let's say there are four agents working under him, called $\mathsf{agent}_1, \mathsf{agent}_2, \mathsf{agent}_3, \mathsf{agent}_4$, and they each have varying levels of skill. We assume that the manager can delegate the task of developing the software to only a single worker, and he will pay the worker according to the time at which the worker completes the job. Again, the questions come up: how should the manager choose whom to assign the task to? We want to ensure that workers truthfully report their valuations. Furthermore, we want to ensure that the task is done as early as possible. This setting can be abstracted as a scenario where the manager is a consumer, and the agents are selling their skills. Thus, we choose to model this setting as an auction, where the workers submit a price that they demand for completing the task at time T, for all possible values of T within the maximal time. If the workers were all competing for the job at a single time, then this would be a traditional single-item auction, where the item being auctioned is the task. However, in this paper, we are considering the added twist that there is a range of times at which the task may be completed. As such this is not a traditional auction setting, and we explore the intricacies and complexities involved with attaching the time parameter into the task allocation problem.

The basic model can evolve in two directions; first, by increasing the number of tasks. The simplest case is where

1. There is a manager who has a single indivisible task to be completed at a specifed time, there are several agents that submit their bids for the task.

2. We can increase the complexity by allowing for the task to be divided into subtasks and still consider that there is only a single time at which groups of subtasks must be completed in conjunction.

If we model these two situations using auction mechanisms, the first would be modeled as an auction with a single item and multiple bidders. The second would be modeled as a combinatorial auction with mutiple items and multiple bidders. These two types of settings have been extensively studied. The second direction in which the model can evolve is to introduce time, and we consider the above two variations with time as a parameter.

1. There is a single indivisible task to be completed at a single time chosen from a specified range of discrete times, and there are several agents that submit their bids for the task.

2. The task can be divided into subtasks and groups of subtasks may be completed in conjunction at different times from within a specified range of discrete times.

We are concerned with studying these two cases in more detail. In this work, we are also interested in modeling the idea of *complementary tasks*. In the setting where there are multiple subtasks that must be allocated to agents at various times, we can consider possible modifications or relaxations of our model that may further increase the efficiency with which the major task is completed. Hence, we also take into account that agents are multi-skilled and that subtasks of a composite task are often related to one another. As such, it is natural to assume that agents may have valuations on completing groups of subtasks at various times. We model this by allowing agents to submit bids on all combinations of subtasks, which we refer to as *bundles.*

In this scenario, we design an auction mechanism based on a social-utility maximizing model which results in optimal bundle allocation. We also design an algorithm to compute the outcome of this mechanism which is faster than the naive brute force algorithm. The resulting mechanism turns out to be an interesting variant of standard auctions. To illustrate the kind of auctions that arise, consider the case where there is just a single task J which can be completed either at time 1, 2 or 3. This can be thought of as 3 different tasks which are up for auction, J_1, J_2, and J_3 (where J_i denotes the task of completing J in time i); but *only one* of of the J_i can be sold.

To evaluate the effectiveness of the mechanism, we assign agents a numerical measure of complementarity, and we use this measure to model the bids for all bundles at all times. The agents' bids for all bundles will depend on their complementarity measure and their valuations for the individual tasks within the bundle. We model the two cases above (with time as a parameter), and modify traditional auctions solving mechanism to determine the winners and payments for the optimal allocation of tasks. Through simulations, we study how this mechanism can take advantage of complementary bids.

We can imagine real-life settings where time (or quality) is a desired parameter. For example, in a setting like Amazon Mechanical Turk [4], we could have a requester post about a time-sensitive job (that can be broken up into subtasks) where the "turkers" could use self-evaluation of their skills in regards to the job to make bids on various bundles of subtasks with various finishing times. Then, the requester can use our mechanism to determine the turkers to whom the subtasks are assigned, what their expected completion times are, and how much they get paid. This thesis ultimately demonstrates how our mechanism for allotting tasks can exploit complementarity of subtasks. The major modifications to the traditional auction settings are the introduction of time as a parameter and modeling complementarity of subtasks in addition to time. We study the interesting effects of allowing for these modifications in the following sections.

2 Related Work

The fields of task allocation using auction mechanisms has been studied in various settings. In particular, multi-auction methods were used for efficient task allocation [1]. Greedy algorithms typically have been used in task allocation using auction methods, including the stochastic clustering approach, where simulated annealing is utilized to determine optimal exchanges between clusters of agents. [11] Winner determination in combinatorial auctions is a hard problem to solve in polynomial time. Winner determination in combinatorial auction settings has been extensively studied [10]. Search algorithms utilizing dynamic progamming were studied in [9]. Other techniques include variations on the greedy algorithm, simulated annealing, and genetic algorithms [3]. A modification on the combinatorial auction setting is the combinatorial exchange setting, where iterative methods have been used for winner determination [6], [5]. In the combinatorial exchange setting, there are multiple items not for sale in an auction, but rather agents get utility from trading items for various prices. [6] describe a tree-based-bidding-language (TBBL Tree Based Bidding Language) and studied mechanisms for winner determination using iterative techniques to achieve socially optimal combinatorial exchanges.

3 Allocation of a Single Task with Multiple Times

A task allocation problem, in this context, is the following. There is a single manager whose aim is to use the workers under him to complete a single task as efficiently as possible. This situation can be generalized in many ways, and we consider possible modifications in the coming sections. We first consider the simple case where there is a single task, N workers, and T possible times for which the workers may submit bids. The manager must pick a single worker to complete the task, and he must pick a single time within which the worker must do so. We assume that the workers have some skill levels which they use as a measure to generate bids on their own. We require each worker to submit a bid for each possible time. The manager has some budget and utility that he wants to maximize. Since the manager would like the job to be completed as early as possible (with less time taken to complete the job being the indicator of increased efficiency), the manager's utility is a decreasing function of time. The workers are bidding for the job at multiple times. Of course, it is at least as easy for an agent to complete the job with a larger amount of time, and so their bids for the task at higher times will be lower. Thus, workers' bidding functions decrease with increasing times. The workers derive negative utility from doing the task and derive positive utility from getting paid for it by the manager, and hopefully the mechanism will make their net utility nonnegative.

Several intuitive questions are the following: how should the manager pick the worker that maximizes his utility? How can we be certain that the workers are reporting true valuations of their skills and correspondingly, true bids for each time? Since we are modeling this as an auction, but there are several bids for each possible time this is an atypical auction setting. Some of the issues that we consider are how to adapt existing auction mechanisms to find optimal task allocation assignments in this case. We will use modified auction mechanisms in order to determine both the "winner" and the winner payments.

4 Model for Auction of Single Task with Multiple Times

In this section we describe the problem formally and establish the notation that we will use throughout the remainder of the paper. Generally speaking, given the situation where a manager is faced with the issue of hiring workers to complete a single task as efficiently as possible, we would like to determine the best allocation of the task to a worker, at an optimal time. The optimal allocation will satisfy the condition that the social utility across the agents (including the manager) is maximized. Thus, we assume that there are N workers that the manager may choose amongst, and T possible times for which the workers may submit bids. We require each worker to submit a bid for each of the times. We now formalize the output of our mechanism.

1. Each agent i submits a bid p_{it} for time t (this should be thought as meaning that player i will lose p_{it} utility if he had to do the task in time t).
2. The manager has utility m_t for time t.
3. We say that an *alternative* a is an (agent, time) pair including the possibility that the task is assigned to no one (in this case, the alternative is (Nonec,time).
4. The set of alternatives A consists of

$$\{(agent, time) \mid agent \in \mathsf{Agents} \cup \mathsf{None}, time \in \mathsf{Times}\}$$

5. The agents and manager have valuations over the alternatives

$$v_i(a) : A \to R^+$$

that look like

$$v_i(a = (j,t)) = \begin{cases} -p_{it} & \text{for j = i} \\ 0 & \text{for j } = \mathsf{None} \\ 0 & \text{for j } \neq i \\ m_t & \text{for i = mgr} \end{cases}$$

4.1 Winner Determination

We consider the problem of winner determination with a single task and multiple times. We define the winning alternative to be the alternative that maximizes social utility. This is a general setting for an auction, and as such, we use the Vickrey-Clarke-Groves mechanism to find the optimal outcome.

The social utility is the sum of the agents' valuations:

$$\sum_{i \in \mathsf{Agents}, mgr} v_i(a)$$

and we choose the alternative $a \in A$ such that $\sum_{i \in \mathsf{Agents}, mgr} v_i(a)$is maximum.

This sum can be expanded as follows:

$$\sum_{i \in \mathsf{Agents}, mgr} v_i(a = (i', t')) = 0 + 0 + \ldots + -p_{i't'} + \ldots + m_{t'} = m_{t'} - p_{i't'}$$

We let the optimal alternative be the pair (i^*, t^*), chosen such that $m(t^*) - p_{i^*}(t^*)$ is maximum.

4.2 Vickrey Clarke Groves Mechanism

The Vickrey-Clarke-Groves (VCG) mechanism can be applied in a general auction setting, with sealed bids and multiple items being auctioned. The system ensures that all players is incentivized to report their true valuations over the alternatives, i.e., the payment scheme is incentive compatible [8]. VCG provides both a rule for determining the winner and a rule for describing the payments incurred for every agent involved in the auction. The winner is chosen to be the alternative $a \in A$ that maximizes $\sum_i v_i(a)$. The intuition is that the winning alternative is the one that maximizes social utility, which in this case is the sum of all agents' valuations of an alternative.

4.3 VCG Payments

In addition to specifying the winner for an auction in a general setting, VCG also specifies the payments for every agent involved in the auction [8]. The payment expected from any agent is the "externality cost" incurred by that agent. This is found by determining the total social utility of all other agents given the current winning alternative, and then subtracting this from the total social utility derived from a new winning alternative when the agent is removed from the set of all agents. This payment mechanism is incentive compatible, and can be partly attributed to the idea that the payment for any agent does not depend on his valuation. The VCG mechanism defines payment functions pay_i for every $i \in n$. The payment function is defined as follows:

$$pay_i = h_i(v_{-i}) - \sum_{j \neq i} v_j(a)$$

where

$$v_{-i} = (v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$$

$$h_i(v_{-i}) = \max_{b \in A} \sum_{j \neq i} v_j(b)$$

4.4 Payment for Winner

We will now calculate what every agent involved in this scheme must pay. The VCG mechanism dictates that the payment for player i is

$$pay_i = h_i(v_{-i}) - \sum_{j \neq i} v_j(a)$$

where

$$h_i(v_{-i}) = \max_{b \in A} \sum_{j \neq i} v_j(b)$$

and a is the winning alternative. We first calculate the payment for worker i^*, the winning worker. According to the VCG payment scheme,

$$pay_{i^*} = h_{i^*}(v_{-i^*}) - \sum_{j \neq i^*} v_j(a)$$

where

$$h_{i^*}(v_{-i^*}) = \max_{b \in A} \sum_{j \neq i^*} v_j(b)$$

So, in this case, we have

$$h_{i^*}(v_{-i^*}) = \max_{(i',t') \in A} v_1(i',t') + v_2(i',t') + \ldots + v_{i^*-1}(i',t') + v_{i^*+1}(i',t') + \ldots + v_{mgr}(i',t')$$

Since we know that worker i has nonzero value for an alternative (a,k) only if $i = a$ (only if he is the chosen worker), we see that these terms look like $-p_{ak'} + m_k = m_k - p_{ak}$ over all $a \neq i^*$ and all times k. So we have that

$$h_{i^*}(v_{-i^*}) = m_{t'} - p_{i't'}$$

where alternative (i',t') yields the maximum social utility when agent i^* is removed from the set of possible workers. Now we calculate

$$\sum_{j \neq i} v_j(a)$$

which in this case is

$$\sum_{j \neq i^*} v_j(i^*,t^*) = v_1(i^*,t^*) + v_2(i^*,t^*) + \ldots + v_{i^*-1}(i^*,t^*) + v_{i^*+1}(i^*,t^*) + \ldots + v_{mgr}(i^*,t^*)$$

Since no other agents except i^* and the manager regard this alternative as having nonzero value, and since i^* is omitted from the total valuation of this alternative, this is simply equal to m_{t^*}. Thus, the winner has to pay

$$m_{t'} - p_{i't'} - m_{t^*}$$

In other words, the winner will receive a payment of

$$m_{t^*} + p_{i't'} - m_{t'}$$

4.5 Payment for Non-Winner Workers

Now we calculate the payments for the remaining workers, using the VCG payment scheme. All other agents l will have to pay

$$pay_l = h_l(v_{-l}) - \sum_{j \neq l} v_j(i^*, t^*)$$

. We know that

$$\sum_{j \neq l} v_j(i^*, t^*) = v_1(i^*, t^*) + \ldots v_i(i^*, t^*) + \ldots + v_m gr(i^*, t^*) = m_{t^*} - p_{i^* t^*}$$

because this expression is just the sum of all agents' valuations of the winning alternative, omitting the current worker l - since the winner i^* is still included in the sum of the valuations, his utility from winning is counted in the total. Next, we know that

$$h_l(v_{-l}) = \max_{b \in A} \sum_{j \neq l} v_j(b)$$

The winning alternative (i^*, t^*) is still included in the set of alternatives because worker i is now omitted when constructing the new set of alternatives, and i^* remains. So we pick the $b \in A$ that maximizes the sum of utilities without agent l, which will be the alternative (i^*, t^*). Thus

$$h_l(v_{-l}) = p_{i^* t^*} - m_{t^*}$$

So, we have that the payment for any agent l that is not a winner is

$$p_{i^* t^*} - m_{t^*} - (m_{t^*} - p_{i^* t^*}) = 0$$

All other agents will not have to give any payment.

4.6 Manager Payment

In this section we calculate the VCG payment for the manager.

$$pay_{mgr} = h_{mgr}(v_{-mgr}) - \sum_{j \neq mgr} v_j(i^*, t^*)$$

where

$$h_{mgr}(v_{-mgr}) = \max_{b \in A} \sum_{j \neq mgr} v_j(b)$$

So we have

$$\sum_{j \neq mgr} v_j(i^*, t^*) = v_1(i^*, t^*) + \ldots + v_{i^*}(i^*, t^*) + \ldots v_N(i^*, t^*)$$

recalling that there are N possible workers in total. This is equivalent to

$$\sum_{j \neq mgr} v_j(i^*, t^*) = 0 + \ldots + (-p_{i^*t^*}) + \ldots + 0 = -p_{i^*t*}$$

Next we calculate

$$h_{mgr}(v_{-mgr}) = \max_{b \in A} v_1(b) + v_2(b) + \ldots + v_{i^*}(b) + \ldots + v_N(b)$$

Since we want the alternative that maximizes this sum, and no one being assigned the task is a viable alternative, we find that it is the alternative that yields the maximal utility of 0. If any other alternative was chosen, the sum of utilities would be negative since the manager's utility is not considered in this setting. So we find that the manager's payment is $-p_{i^*t^*}$. This is an important point to be discussed about the VCG mechanism; it often can lead to an imbalance in payments [2]; here we see that the sum of payments for all involved agents (manager and workers) is nonzero, but the assigned worker must recieve payment. Thus in this setting, the manager serves as both the auctioneer and as an involved agent in the calculation of social utilities. The introduction of the additional parameter of time requires that the manager's utilities be taken into account when determining the optimal alternative, however when decided payments the manager must make up for the imbalance and step back into the role of auctioneer.

5 Incentive Compatibility

We now show that these payments are incentive compatible. The VCG mechanism is known to ensure incentive compatibility, but we show that the payments derived above are incentive compatible for all agents and also show that VCG is applicable in this scenario.

With the payments derived above, we now check: does any agent involved in this scheme have incentive to deviate?

We first note that the original winner i^* has no incentive to deviate; if he deviates so that he does not win the task, then he will receive utility 0 which is less than his current profit. If he deviates in a manner such that he is still the winner, his payment will not change because it does not depend on his reported valuations. Overall, it is futile for i^* to deviate from revealing his true valuations.

Take agent i. Say he reports some bid ${p'}_{it}$ instead of his true bid p_{it} for some time t.

If $m_t - {p'}_{it}$ is not the new maximum value for the social utility, then it is clear that i no incentive to deviate because he is still not a winner.

If it is the case that $m_t - {p'}_{it}$ is the new maximum, that is,

$$m_t - {p'}_{it} \geq m_{t^*} - p_{i^*t^*},$$

then agent i will be the new winner and consequently will get paid

$$m_t + p_{i^*t^*} - m_{t^*}$$

and will derive utility equal to his true valuation of doing the job at time t, which is $-p_{it}$. Now we check, is the agent's net profit positive? That is, is it that case that

$$m_t + p_{i^*t^*} - m_{t^*} + (-p_{it}) > 0?$$

Since we know that in reality

$$m_{t^*} - p_{i^*t^*} \geq m_t - p_{it}$$

we can say that

$$m_{t^*} - p_{i^*t^*} - m_t + p_{it} \geq 0$$

and consequently that

$$m_t + p_{i^*t^*} - m_{t^*} - p_{it} \leq 0$$

Thus, agent i will not derive positive profit from deviating from his true valuation, and the payments ensure incentive compatibility.

6 Divisible Task with Multiple Times

6.1 Modeling Complementarity

So far, we have considered the case where there is a manager whose aim is to complete a single task, and he has the option of assigning the single task to a single agent to complete within a predetermined period of time. However, this is often not very realistic and does not seem to allow for efficiency. While it is possible that a task may be such that it can only be completed in entirety to an agent, we shall now consider tasks that may be easily divided into subtasks. Similarly, it is possible that budgetary or other restrictions require that the manager assign a task to a single agent, but it is more realistic that the manager has been given workers that can all work on different tasks in parallel. Furthermore, it is reasonable to assume that workers can handle working on multiple groups of tasks. We again assume that workers must submit bids for a discrete set of times, but now we allow that a worker can submit bids for collections of subtasks. We call such a collection a *bundle* of subtasks. Thus, with the introduction of bundles, and the allowance for workers to submit bids on bundles, we can now consider the problem in a more general setting.

To summarize, the setting is as follows:

1. The task at hand can be easily split into a group of subtasks.

2. The workers must submit bids on all bundles of subtasks.

3. The workers submit such bids for all possible times.

Using this preliminary setup, we can now introduce the notion of *complementary subtasks*. One of the main reasons for the division of the original task into subtasks is to make the process of completing the task more efficient. A very natural assumption about the subtasks of a larger task is that the output or results of one subtask may aid in faster completion of another. When a worker bids on a bundle of subtasks, it means that he believes that bundle of subtasks to be complementary in some way.

6.2 Formal Model of Problem

In this section we describe the problem formally and establish the notation that we will use throughout the remainder of the thesis. Generally speaking, given the situation where a manager is faced with the issue of hiring workers to complete a task, composed of several subtasks, as efficiently as possible, we would like to determine the best allocation of bundles of subtasks to the workers, with a time assigned for each worker. The optimal allocation will satisfy the condition that the social utility across all agents (including the manager) is maximized. Thus, we assume that there are N workers that the manager may choose amongst, M subtasks to be bid upon, and T possible times for which the workers may submit bids. Each worker will submit a bid for every (bundle,time) pair. We now formalize the output of our mechanism. We say that an *alternative* is a function a from the set of all subtasks to the set of all (agent,time) pairs, including the possibility that a subtask may be assigned to no one (represented by None). Formally, this is:

$$a : \mathsf{Tasks} \to (\mathsf{Agents} \times \mathsf{Times}) \cup \mathsf{None}$$

where

$$\mathsf{Tasks} = \{\mathsf{task}_1, ..., \mathsf{task}_M\}$$

$$\mathsf{Agents} = \{\mathsf{agent}_1, ..., \mathsf{agent}_N\}$$

$$\mathsf{Times} = \{1, ..., T\}$$

We denote the set of alternatives by A.

In this setting, an optimal allocation will be a disjoint splitting over all subtasks into bundles, with each bundle being assigned to the agent with the lowest bid for that bundle. We will discuss the optimal allocation more formally in coming sections.

6.3 Divisible Task Setting as an Extension of Single Task Setting

The divisible task setting requires agents to submit bids on bundles of subtasks and times. This is basically as though there is now a more expansive set of tasks for each time - we can consider every bundle to be a different "task". Taking complementarity into account, the agents now will re-evaluate their bids for each bundle of tasks based on their notion of how complementary the tasks are. Acknowledging this lets us reuse the VCG mechanism for both determining the optimal outcome as well as determining payments for all agents in the optimal allocation.

6.4 Valuation Functions

We assume the manager will get utility only if all subtasks are completed, i.e. that the single major task is done to completion, and furthermore that the manager seeks to minimize the maximum time for all bundles of subtasks to be completed. Thus the manager has utility $m(t)$ solely as a function of time, and his utility is nonzero only when the major task is finished. (We will refer to the subtasks of the major task interchangeably as both tasks and subtasks. When discussing the major task to be completed, we will refer to it as the "major task".)

Each player will get different utility depending on the bundle of subtasks he is assigned, and also depending on the time for which the bundle is assigned. Players will have zero utility if they are not assigned any bundle of tasks in the winning alternative. So, when an agent bids on a single subtask and is assigned that subtask in the optimal allocation, then he has valuation equal to his bid for the single subtask. If an agent does not get assigned any subtask, his valuation on the optimal allocation is 0. If an agent is assigned a bundle of subtasks, then his valuation on this allocation is his bid for the bundle of subtasks. Recall that we are assuming that agents submit true valuations because we use the VCG principle of maximizing social utility to solve for the winning allocation, and VCG ensures incentive compatibility.

We encapsulate these ideas in a valuation function over the space of alternatives. For an alternative a, the valuation function will take the form $v_i(a)$ as described below. For a given alternative a, we use $a_{agent}(b)$ to denote the agent assigned to task b in the allocation, and we use $a_{time}(b)$ to denote the time allotted for the the completion of task b (this is the time allotted to $a_{agent}(b)$ to complete b). Effectively, a_{agent} contains the set of all agents assigned tasks in the alternative, and a_{time} contains the set of all times chosen to finish the tasks. We let B_i represent all (task,time) pairs (b, t) assigned to agent i and say that $-p_i(B_i)$ denotes agent i's utility for being assigned the bundle B_i. That is, the agent has utility equal to negative of his bid for his assigned (bundle, time) pair. In summary, we have:

$$v_i(a) = \begin{cases} v_{mgr}(a) & \text{for i = mgr} \\ v_{p_i}(a) & \text{for i} \in \textsf{Agents} \end{cases}$$

$$\text{Where} \qquad v_{mgr}(a) = \begin{cases} 0 & \exists\ \mathsf{task}_j, a_{agent}(\mathsf{task}_j) = \mathsf{None} \\ m(t) & t = \max_j a_{time}(\mathsf{task}_j) \end{cases}$$

and

$$v_{p_i}(a) = \begin{cases} 0 & i \neq a_{agent}(\mathsf{task}_j)\ \forall\ \mathsf{task}_j \in \mathsf{Tasks} \\ -p_i(B_i) & B_i = \{(b,t) \mid a_{agent}(b) = i \text{ and } a_{time}(b) = t\} \end{cases}$$

6.5 Winner Determination

We again follow the VCG method for determining the winner. We want to choose the alternative a that maximizes the social utility of all agents, where social utility is defined as

$$\sum_{i \in Agents, mgr} v_i(a)$$

We call a the *optimal allocation* for this problem. We note that optimal allocation will be some assignment of bundles of subtasks to the agents, including the possibility within a some bundle of subtasks is assigned to no agent if the bids for those subtasks were not high enough. We can treat this as a combinatorial auction because each agent submits bids for all bundles at different times, and it is as though all (bundle,time) pairs correspond to bids for separate "tasks". This is effectively a combinatorial auction, with an added twist that the optimal allocation of tasks to workers must be optimal with respect to the time parameter as well. How would one determine the winner in this setting, using the VCG premise of maximizing the total social utility? A natural choice would be to cycle over all the possible alternatives and determine the sum of the social utility of each agent, including the manager, of each alternative. Then, the optimal allocation is chosen to be the alternative that maximizes total social utility.

The total number of bundles that workers may bid on is 2^M, because it is simply the total number of subsets over the set of subtasks. Thus, the total number of bids that workers can submt s $2^M T$ because for each bundle, a worker must submit bids for all possible times.

For each task, every (agent,time) pair is a viable allocation, and the allocation of 'none' is also an option. As such, the total number of possible alternatives is

$$(NT+1)^M$$

So, the winner determination problem can be solved in time

$$O(NT)^M$$

This is the running time of the naive brute force winner determination algorithm.

We use a dynamic programming algorithm to determine the winner more efficiently.

6.6 Dynamic Programming for Winner Determination in Combinatorial Auctions

We now recall the dynamic programming approach to solving winner determination in standard combinatorial auctions (without time) [9]. We will see in a later section how the winner determination problem for our time-enhanced task allocation problem can be solved by invoking this dynamic programming algorithm several times.

The dynamic programming approach to solving this problem involved starting from a "small" subset S out of the set of all subtasks M, taking all possible splittings of the small set into two bundles and comparing the sum of the bids over all agents for each possible splitting. The current "optimal" splitting of S into bundles is noted, and the minimal bids and the agents corresponding to those minimal bids are also noted. The minimal bid of the optimal splitting is compared to the minimal bid for the bundle $\{S\}$, and the new larger set S' is considered in the same way.

The dynamic programming algorithm iterates over possible subsets of size less than or equal to $|\mathsf{Tasks}| = M$. Beginning with subsets of size 1, it determines what the minimum bids and bidders for each individual task are. Then, it checks all possible splittings of the subsets of size 2. Given the minimal splittings of these subsets, it checks the subsets of size 3, and so on. Thus, for a subset of size k, it must check all 2^k splittings to determine the minimal bids and bidders for each split; the algorithm must determine this for all possible subsets of Tasks that are of size k. There are $\binom{M}{k}$

subsets of Tasks of size k. So the total number of operations the algorithm performs is

$$\sum_{k=0}^{M} \binom{M}{k} \cdot 2^k = 3^M.$$

Thus the dynamic programming algorithm for solving combinatorial auctions runs in time 3^M.

We chose the dynamic programming algorithm for winner determination in this case because, while still exponential in the number of tasks, it provides much better running time than searching through the space of all alternatives for the alternative that yields the highest social utility. There have been other approaches for optimal winner determination in the CA setting, for example, a search algorithm that utilizes expressive bidding languages and preprocesses the search space [9] and a greedy heuristic algorithm that approximates the optimal auction outcome. However, we favor an exact optimal allocation over an approximately optimal allocation, and we also favor minimal preprocessing of the search space since we require that agents submit bids for every possible (bundle,time) pair. As such, using dynamic programming for winner determination was chosen in this setting. The pseudocode is given below.

INPUT: $\bar{b}(S)$ for all $S \subseteq \mathsf{Tasks}$. If no $\bar{b}(S)$ is specified in the input (no bids were received for that S) then $\bar{b}(S) = 0$.

OUTPUT: An optimal exhaustive partition W_{opt}.

```
1. For all $x \in \mathsf{Tasks}$, set $\pi(\{x\}) := \bar{b}(\{x\}), C(\{x\}) := \{x\}$

2. For $i = 2$ to $m$, do:
   For all $S \subseteq M$ such that $|S| = i$, do:

    (a) $\pi(S) := \min\{\pi(S \backslash S') + \pi(S') : S' \subseteq S$ and $1 \leq |S'| \leq \frac{|S|}{2}\}$

    (b) If $\pi(S) \leq \bar{b}(S)$, then $C(S) := S^*$ where $S^*$ minimizes the right hand side of (a)

    (c) If $\pi(S) > \bar{b}(S)$, then $\pi(S) := \bar{b}(S)$ and $C(S) := S$

3. $W_{opt} := \{\mathsf{Tasks}\}$

4. For every $S \in W_{opt}$ do:
   If $C(S) \neq S$, then

    (a) $W_{opt} := (W_{opt} \backslash S) \cup \{C(S), S \backslash C(S)\}$

    (b) Goto 4 and start with the new $W_{opt}$
```

Here, $\bar{b}(S)$ is the set of minimal bids for every possible bundle. Steps (1) and (2) encapsulate finding the optimal bids over the optimal allocation, and steps 3 and 4 involve recovering which partition of Tasks into bundles gave these optimal bids. We define π to be the smallest sum of bids over all partitions of S. Once we make this definition, the dynamic program starts computing $\pi(S)$ for all S of size 1, then for all S of size 2, and so on. The main idea is that given the values of $\pi(S')$ for all S' of size $< k$, we can find $\pi(S)$ by checking for the minimal sum over all $\pi(S'') + \pi(S' - S'')$, for all $S'' \subset S'$. We maintain $C(S)$, which is the optimal partition of S into bundles that corresponds to $\pi(S)$.

6.7 Winner Determination for Divisible Task Allocation with Time

So far, we have considered winner determination in the case where agents may submit bids on bundles for a single time. Now, we discuss how to use the solution for a given time to solve the winner determination problem across multiple times. We first show that any optimal allocation has a single optimal time. That is, an optimal alternative as determined by the winner determination algorithm, which is a function from Tasks to $\mathsf{Agents} \times \mathsf{Times}$, is at least as optimal as the alternative defined as the function from Tasks to $\mathsf{Agents} \times \{t^*\}$, where t^* is the optimal time.

Theorem 1. *An optimal allocation of tasks to agents in the scenario where time is a parameter has the property that there is a single optimal time t^*, namely the maximum time over all (agent, time) pairs in the optimal allocation.*

Proof. Take any optimal allocation $a' : \mathsf{Tasks} \to \mathsf{Agents} \times \mathsf{Times}$. Since a' is optimal we know that it maximizes social utility across all agents and the manager. That is, a' is an assignment of bundles of tasks to agents, with an allotted time for each bundle, such that the utility that all agents get from this allocation is maximum. We define a^* to be the same allocation of bundles of tasks to agents, except that agents are all now allotted the maximum time across all (bundle,time) pairs in a' to complete their tasks.

We want to show that $a^* : \mathsf{Tasks} \to {a^*}_{agents} \times \{\max({a^*}_{time})\}$ yields at least as much social utility as the social utility derived from allocation a'. Thus, we want to show that

$$\sum_{i \in \mathsf{Agents}, mgr} v_i(a^*) \geq \sum_{i \in \mathsf{Agents}, mgr} v_i(a')$$

Expanding the terms in the right hand side, we have

$$\sum_{i \in Agents, mgr} v_i(a') = v_1(a') + v_2(a') + \ldots + v_N(a') + v_{mgr}(a')$$

For every worker $i \in a'_{agent}$, we know that his valuation on a bundle B_i consisting of a set of $(task, time)$ pairs is the negative of his bid for that bundle.

$$v_i(a') = -p_i(B_i)$$

$$\text{where} B_i = \{(b,t) \mid a'_{agent}(b) = i \text{ and } a'_{time}(b) = t\}$$

We let B_i^* denote the bundle B_i but with each task now being paired with time $t*$, where $t*$ is the maximum assigned time over all bundles and agents.

So we have

$$t^* = \max_{j \in \mathsf{Tasks}} a'_{time}(j)$$

and

$$B_i^* = \{(b, t^*) | (b,t) \in B_i\}$$

Since the agents have bids that decrease with time, and corresponding utility that decreases with time, we know that for all i the utility that an agent gets from being assigned a bundle B_i is less than or equal to the utility that the agent gets from being assigned the same bundle at a higher time.

$$v_i(B_i^*) \geq v_i(B_i)$$

Taking the sum of the valuations over all agents, we have

$$\sum_i v_i(B_i^*) \geq \sum_i v_i(B_i)$$

Since

$$v_{mgr}(a^*) = v_{mgr}(a')$$

we conclude that

$$\sum_i v_i(B_i^*) + v_{mgr}(a^*) \geq \sum_i v_i(B_i) + v_{mgr}(a')$$

Restating this, we have our original claim

$$\sum_{i \in \mathsf{Agents}, mgr} v_i(a^*) \geq \sum_{i \in \mathsf{Agents}, mgr} v_i(a')$$

So the social utility derived from the same allocation of tasks to agents, except at a single maximal time t^* instead of the times at which the bundles may have originally been assigned, is greater than the social utility in the previous case.

□

6.8 Algorithm for Optimal Allocation in Divisible Task Setting with Time

In the combinatorial auction model of the problem at hand, the optimal allocation is an assignment of tasks to agents. However, we have established that this setting further differs from a normal combinatorial auctions because the alternatives are now functions from the set of tasks to (agent, time) pairs, and the new factor of time must be somehow incorporated when finding the optimal outcome. In the previous section, we were able to show that any optimal allocation in fact has only a single optimal time. That is, in any optimal allocation, agents are assigned disjoint bundles of subtasks, and they are all allotted a single maximal time within which to complete their assigned bundle. Given this information, it is possible to adapt winner determination in the combinatorial auction setting to winner determination in the combinatorial auction setting with time. We outline the algorithm below.

1. For each time $t \in \mathsf{Times}$ do:
   (a) Take as input the bids from all agents $i \in \mathsf{Agents}$ for each bundle of tasks $S \subseteq \mathsf{Tasks}$ for the time t.
   (b) Determine $\overline{b_t}(S)$ for every $S \subseteq \mathsf{Tasks}$: the minimal bids for bundles at time t
   (c) Run the dynamic optimization algorithm with input $\overline{b_t}(S)$
   (d) Store the output - an optimal alternative consisting of bundles of tasks to agents, and minimal bids for each bundle - in $OptAlloc[t]$
   (e) Compute $v_{mgr}(OptAlloc[t]) - \sum_{i \in \mathsf{Agents}} v_i(OptAlloc[t])$ and store in $SocUtil[t]$

2. Determine $t^* = t \in T$, the time that maximizes $SocUtil[t]$

3. Output $OptAlloc[t^*], SocUtil[t^*]$

The way to optimize this kind of combinatorial auction is to treat all the bids for a given time as bids for a single combinatorial auction. Thus an optimal allocation can be determined for each time, and then the manager's utility can be used over all times to determine which optimal allocation delivers the manager the most utility.

More specifically, an optimal allocation for a given time would mean that the agents derive most social utility from that allocation for that time. Then, since we want to maximize total social utility over all times, we introduce the manager's valuation for each time to determine which allocation yields the highest social utility with the manager included. So, an algorithm for solving combinatorial auctions would be able to solve this problem as well.

6.9 VCG Payments

Now we discuss how to allot the payments for the combinatorial auction case with time. The CA setting takes as input agents' bids for all bundles for every possible time, so it can be considered a generalized version of the setting where agents submit bids for each individual task for every possible time. It is as though each bundle is a separate "task" for which an agent is revealing his valuation. Since we determine the winning allocation in the CA setting with time using the VCG principle of maximizing total social utility of all agents, including the manager, we use the VCG mechanism to determine the payments for all agents involved. Recall that the VCG payment for each agent $i \in \mathsf{Agents}$ is given by

$$\text{pay}_i = h_i(v_{-i}) - \sum_{j \neq i} v_j(a^*)$$

where a^* is the winning alternative and

$$h_i(v_{-i}) = \max_{\alpha \in A} \sum_{j \neq i} v_j(\alpha)$$

where A is the space of all alternatives, is the social utility of all agents when agent i is removed from the set of agents. The payment is calculated as follows: for every agent i, we calculate the sum of every other player's valuation of the winning alternative a^*. This is $\sum_{j \neq i} v_j(a^*)$. Then, we remove i

from Agents, and consequently, remove his bids from the set of all bids on bundles of tasks. Then, we re-solve for the optimal allocation using the algorithm in 6.8; first we determine the optimal allocation for each time, and then we determine the optimal allocation across all times by choosing the allocation and time that yields the manager the highest utility. We denote this new optimal allocation without agent i to be a'. This allocation a' has the property that it maximizes the social utility of all agents. The payment for i is given by the difference in the two:

$$\text{pay}_i = \sum_{j \in \mathsf{Agents} \setminus \{i\}, mgr} v_j(a') - \sum_{j \in \mathsf{Agents}, mgr, j \neq i} v_j(a^*)$$

We have just described how to find the payment for agent i. We now discuss the properties that this payment has, namely being greater than 0 and at least as much as the i's valuation of the alternative a^*. We know that, by virtue of being the optimal allocation, the social utility generated over all $k \in$ Agents for a^* is highest, and is greater than the social utility generated over all agents (including i for a').

That is

$$\sum_{k \in \mathsf{Agents}, mgr} v_k(a^*) \geq \sum_{k \in \mathsf{Agents}, mgr} v_k(a')$$

We further know that

$$\sum_{k \in \mathsf{Agents}, mgr} v_k(a^*) = \sum_{j \in \mathsf{Agents}, mgr, j \neq i} v_j(a^*) + v_i(a^*)$$

So we have

$$\sum_{j \in \mathsf{Agents}, mgr, j \neq i} v_j(a^*) + v_i(a^*) \geq \sum_{k \in \mathsf{Agents}, mgr} v_k(a')$$

Also, since alternative a' is chosen after i is removed from Agents, we know that i's valuation of a' is 0.

Thus we know that

$$\sum_{k \in \mathsf{Agents}, mgr} v_k(a') = \sum_{j \in \mathsf{Agents} \setminus \{i\}, mgr} v_j(a')$$

Putting this together, we have

$$\sum_{j \in \mathsf{Agents}, mgr, j \neq i} v_j(a^*) + v_i(a^*) \geq \sum_{j \in \mathsf{Agents} \setminus \{i\}, mgr} v_j(a')$$

and finally

$$\text{pay}_i = \sum_{j \in \mathsf{Agents} \setminus \{i\}, mgr} v_j(a') - \sum_{j \in \mathsf{Agents}, mgr, j \neq i} v_j(a^*) \leq v_i(a^*)$$

So, we know that player i will have to pay at least $v_i(a^*)$. However, since the valuations are in fact negative, this means that agent i will recieve a positive amount that is at least his valuation of the optimal alternative. What about agents that are not "winners"? If an agent is not allocated any task in the winning alternative, then if he is removed from the set of Agents the winning allocation and time will be the same. Consequently, the sum of all other agents' valuations of the original optimal allocation and the new allocation (which is the same as the original) is the same, and the payment for any agent that is not assigned a task, the payment will be 0.

Now we discuss the payment demanded from the manager. It is well known that the VCG mechanism fails with regards to achieving a balanced set of payments. Here, the manager is included in determining the alternative that maximizes social utility across time, because the time parameter is of utmost importance in terms of his utility. If we use VCG to determine the manager's payments, it is possible that we incur a budget deficit [7]. When handing out the payments, the manager must step back into the role of auctioneer and make up for any imbalance in the payments. Thus the manager's role is to deliver the payments required to all agents that are assigned bundles in the final allocation.

To summarize, from the players' perspective this mechanism yields the same payments as given by VCG and hence is incentive compatible from their point of view. Furthermore, the net social utility (including the manager's) is maximized. However the mechanism is not incentive compatible from the manager's perspective, but this is simply a result of the manager assuming a double role of a utility maximizing agent and the auctioneer. Nevertheless, we see that since players can never get negative utility, and so the manager's lack of incentive compatibility cannot cause unrestrained damage.

7 Simulations

In order to compare the true effectiveness of allowing for complementarity of subtasks, we simulate the model using specific utility functions for agents and managers, and generate random data that we feed into our models. We model valuation functions for all agents and also for the manager. We first describe the shape of the utility functions for the agents and manager in the case where agents are submitting bids on individual tasks for various times. Next, we describe the shape of the utility functions for agents and manager in the case where the tasks are subtasks of a composite job to be completed, and agents can submit bids on complementary bundles of tasks. We use these utility functions to generate bids and utilities, and run our optimization algorithm to determine optimal allocations given these valuations. The best measure of the advantage of complementarity is to compare social utility in cases with and without complementarity.

7.1 Modeling Bids for Individual Tasks

We can think of the manager as the consumer in this setting, and the workers offering their skills as the product. As such, if a worker is considering the bid for completing a given task in a longer period of time, he must account for having a longer amount of time to complete the same job. We therefore want agents' bids to be decreasing as a function of time. Similarly, the manager's utility is dependent on how efficiently the task is completed, so his utility is also decreasing as a function of time.

We model an agent i's bid for a single task j at time t to be

$$p_i(j,t) = a_{ij} + \frac{b_{ij}}{C \cdot t + c_{ij}}$$

where a_{ij}, b_{ij}, c_{ij} are chosen randomly for every (agent, task) pair, and C is some constant. Similarly, we model the manager's valuation function of time t to be

$$m(t) = a_{mgr} + \frac{b_{mgr}}{C \cdot t + c_{mgr}}$$

7.2 Modeling Bids for Bundles

In this section, we describe how agents submit bids on bundles of subtasks that they believe have some degree of complementarity. How should we model complementarity? We consider that each agent has an individual notion of how complementary they think bundles of tasks, and we let this be a number c_i for agent i. We refer to c_i as the *complementarity measure*, and also as the agent's $c - value$.

Then we can say that agent i's valuation of a bundle $B = \{(task_a, t_a), (task_b, t_b), \ldots (task_k, t_k)\}$ over some subset $\{task_a, \ldots, task_k\} \in \mathcal{P}(S)$ of all subtasks is

$$[\sum_{j \in B_{tasks}} p_i(j,t)^{c_i}]^{1/c_i}$$

We refer to this value as the agent's $c - bid$ for bundle B. Why does this capture complementarity? If an agent considers that some group of tasks are complementary to one another, then the agent believes that completing these tasks together enables more efficient completion than if they were to be completed separately. For example, the result of completing one task may be of great use in completing another. Given this, an agent should expect payment for a complementary bundle of subtasks to be less than the sum of his bids for the individual subtasks. Using c_i as the "complementarity measure" enables this, because for any bundle of tasks, taking the c-bid of the bundle yields a bid that is greater than the maximum bid for any individual (task,time) pair, but at the same time is less than the sum of the agent's bids over all separate (task,time) pairs. The c-bid also has the property that if a bid for a (task,time) pair in the bundle is much greater than all other individual bids (say that some task requires much more effort than the others and consequently has a higher bid), the c-bid for the bundle will be closer to this maximum value. If $c_i = 1$ for some agent, then we note that the $c - value$ for any bundle will be exactly the sum of the individual bids for each (task,time) pair in the bundle. When $c_i = 1$

$$[\sum_{j \in B_{tasks}} p_i(j,t)^{c_i}]^{1/c_i} = [\sum_{j \in B_{tasks}} p_i(j,t)^1]^1$$

$$= \sum_{j \in B_{tasks}} p_i(j,t)$$

Thus, $c_i = 1$ for any agent represents the situation where the agent does not believe any tasks are complementary in completion of the major task. Furthermore, as c_i tends to infinity, it is known that the c-bid for a bundle lies between the maximum bid for any individual (task,time) pair in the bundle and the sum of all individual bids, and approaches the maximum bid. This is also a useful in our model, because it captures the idea that if as an agent tends to view the tasks as very complementary, then his bid for any bundle should be lower relative to his valuation of the sum of each task in the bundle individually. Since the bundle together seems to be easier for him to complete, its bid should be more on the order of a bid for an individual task (but still take into account that he is bidding on a group of tasks).

8 Results and Discussion

8.1 Varying c-Values

We first looked at the effect of varying the complementarity measure for the workers in our setting. We assigned each agent a random c-value from the four ranges [1,1.5], [1.5,2], [2,4], and [4,10]. We then ran 40 trial auctions, reassigning new utility functions to the agents and the manager with the form discussed in the previous section. The output was an optimal allocation, an optimal time, the payments to all the agents, the net payment required of the manager, and the social utility of all agents. Thus, for any trial, we first assigned all agents brand new utility functions. Then, we assigned all agents a c-value from within the specified range. We ran our dynamic optimizer in this setting and recorded the output. Within the same trial , we re-solved for the optimal setting after resetting all agents' c-values to 1 (recall that an agent having a c-value of 1 indicates no complementarity). We ran 40 such trials for each range of c-values. We let $N = 10$, $M = 6$, and $T = 12$.

8.1.1 Effect on Social Utility

For each range, we measured the benefit of allowing complementarity as follows: we took the ratio of the social utility in the complementarity case over the social utility in the no-complementarity case (shown in the graphs as COMP Social Utility/ No-COMP Social Utility). This provided a scaled measure of the advantage of introducing complementarity.

The graphs have the complementary benefit ratio on the x-axis, with the trial number on the y-axis. We sorted the ratios in ascending order, and disregarded the data points where there was negative social utility (in this case the social utility is maximized by the task not being allocated). The complementarity benefit is the ratio of social utility There are several trends that can be observed.

First, for the range of the c-value lying within [1,1.5], we see that there are more results with the complementarity benefit being closer to 1. The maximum benefit observed is approximately 1.25, and most ratios tend to fall between 1 and 1.1.

For the range [1.5, 2], the maximum benefit is a ratio of 1.43, with more than half the data points passing the ratio 1.1.

For the range [2,4] the maximum ratio observed is 1.4, with more than half the points passing 1.1, and a significant number of points passing 1.2.

Finally, for the range [4,10] the maximum benefit is 1.9, with a significant number of points passing

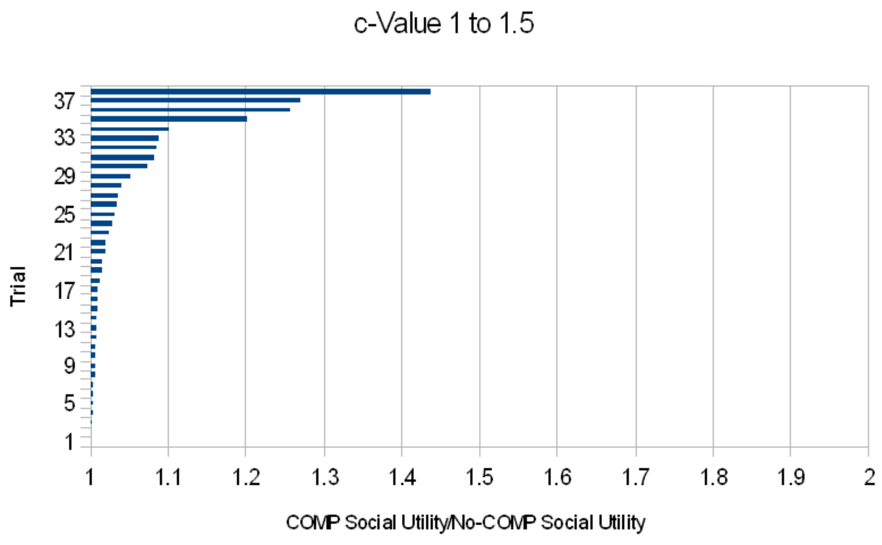

Figure 1: Social Utility Advantage Complementarity between 1 and 1.5

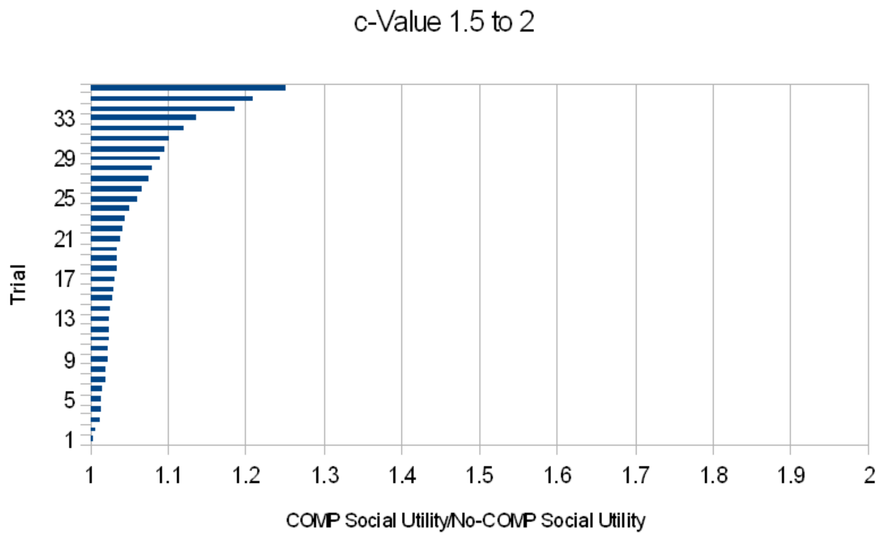

Figure 2: Social Utility Advantage Complementarity between 1.5 and 2

by the ratios 1.2, 1.3, and 1.4.

So, we see that the maximum benefit observed tends to increase as the agent's c-values increase, and that higher social utility is observed more often as complementarity increases. From these results, we can observe a general trend in an increase in overall social utility as agents' measure of complementarity increases. As we expected, the allowance of complementarity causes an increase in social utility overall.

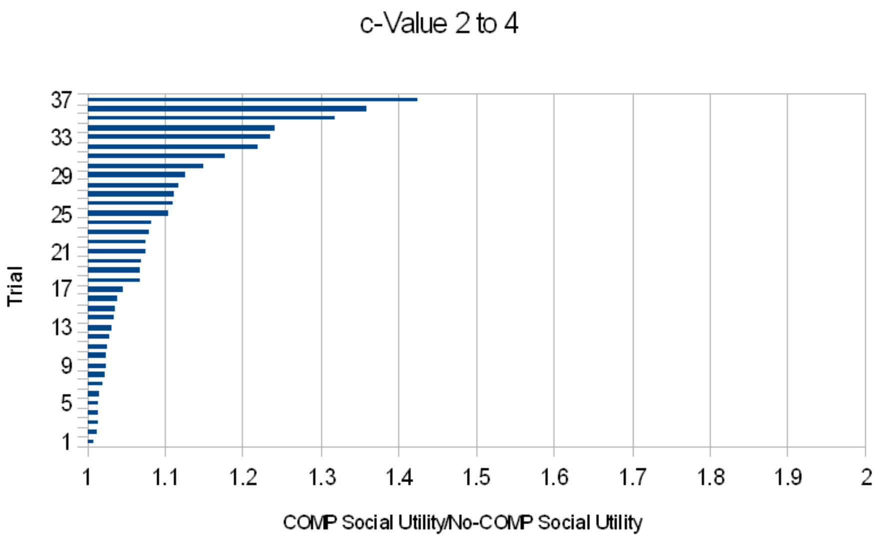

Figure 3: Social Utility Advantage Complementarity between 2 and 4

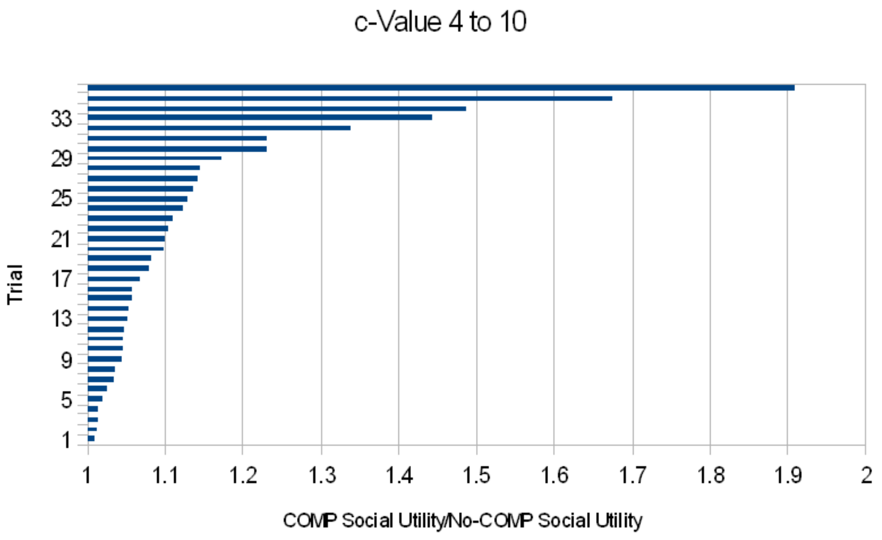

Figure 4: Social Utility Advantage Complementarity between 4 and 10

Higher complementarity values mean that agents tend to think that the effort expended in doing complementary bundles of tasks is much less than the sum of effort expended by doing the tasks in the bundles individually. As a result, tasks will be assigned more in bundles, and workers will tend to think that they can complete the tasks at less cost to themselves. This correlates with higher social utility.

8.1.2 Effect on Manager Payment

We also looked at the trend in the manager's payments as we increased the c-values overall. On the x-axis, we graphed the social utility in the setting with complementarity with a range of (0,80), and on the y-axis, we graphed the social utility in the setting without complementarity with a range of (0,12). In order to compare across the different ranges for the c-values, we look at the trend in the slopes of these points.

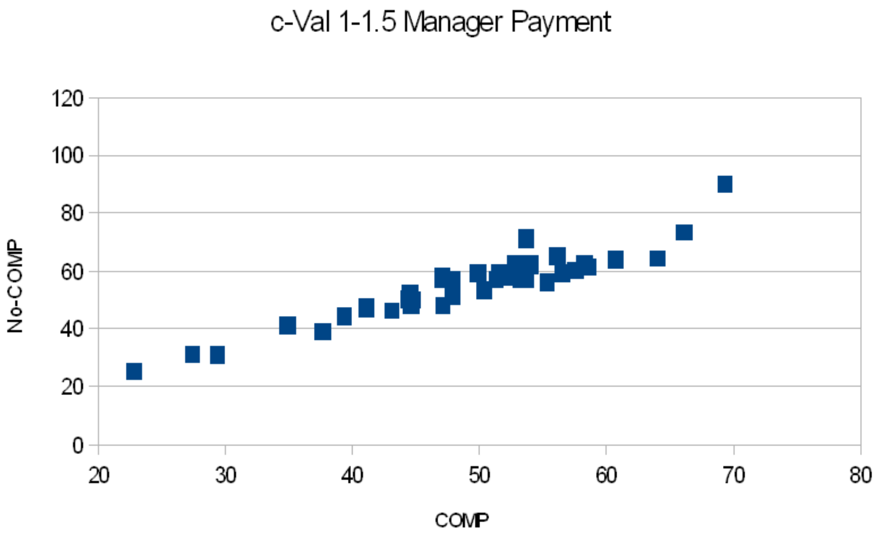

Figure 5: Manager Payments for COMP vs No-COMP with c-Value between 1 and 1.5

We observe the folllowing:

For the range [1,1.5] the slope is much closer to 1, indicating that the manager's payments For the range [4,10] the slope is much higher than 1. From the four graphs, we observe the general trend that the manager has to pay less in the setting where we allow for complementarity, and the difference in the amount the manager has to pay in the Comp. setting vs. No-Comp is increasing as the c-values increase. This is also expected, because as agents think that more bundles are complementary, their bids for bundles will be much less than the sum of the bids for each tasks in that bundle, and the manager's payment to the winning agents will consequently be less. The difference will be more pronounced as complementarity increases because the difference between the bids for a complementary bundle of tasks and the sum of the individual bids for the tasks in the same bundle is increasing as the c-values increase.

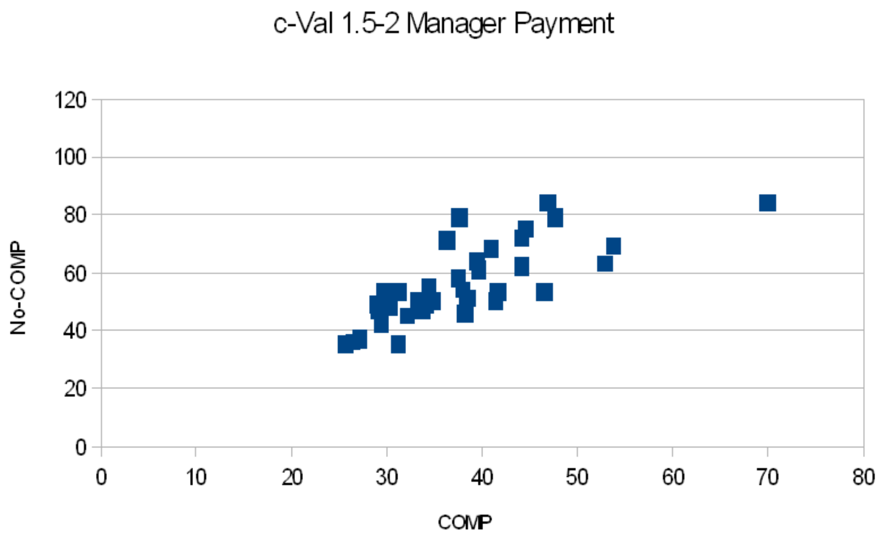

Figure 6: Manager Payments for COMP vs No-COMP with c-Value between 1.5 and 2

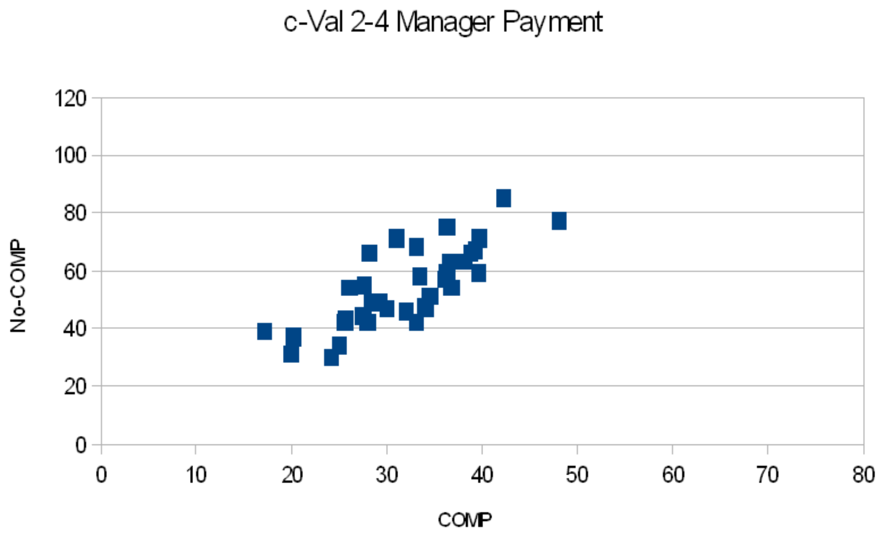

Figure 7: Manager Payments for COMP vs No-COMP with c-Value between 2 and 4

8.2 Varying the Number of Agents

We observed the effect of the benefit of complementarity as N, the number of agents, changes. We varied N in the range [2, 3, 4, 5, 10, 15, 20, 25, 30]. We said the number of subtasks M was 6 and the number of times T was 12. For each value of N, we ran 40 trials, reinitializing utility functions of the manager and the agents at the beginning of each run. For each trial, we first ran the optimization

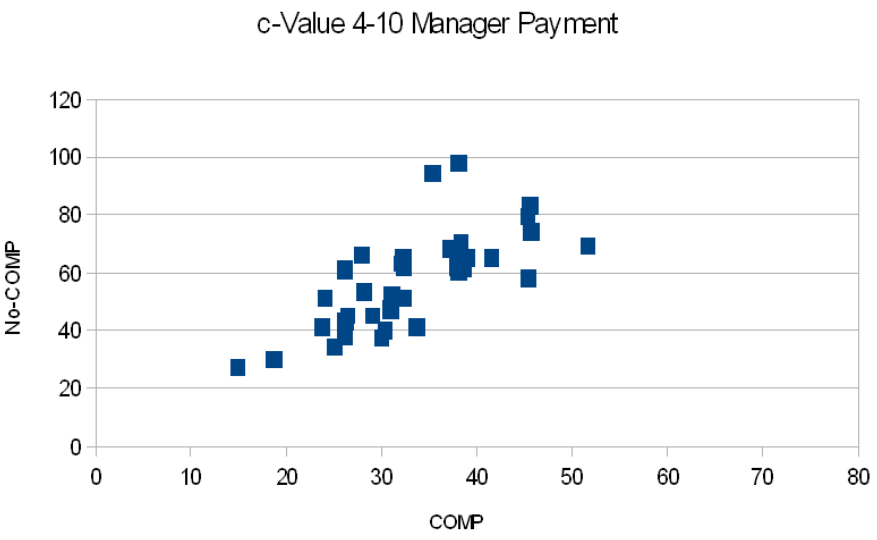

Figure 8: Manager Payments for COMP vs No-COMP with c-Value between 4 and 10

algorithm for c-values in the range [1,3] (COMP), and next ran the optimization with c-values equal to 1 (No-COMP). We then recorded the net social utility of each allocation at the end of the run, both for the COMP and No-COMP settings, and graphed the complementarity advantage ratio (as described earlier) against N.

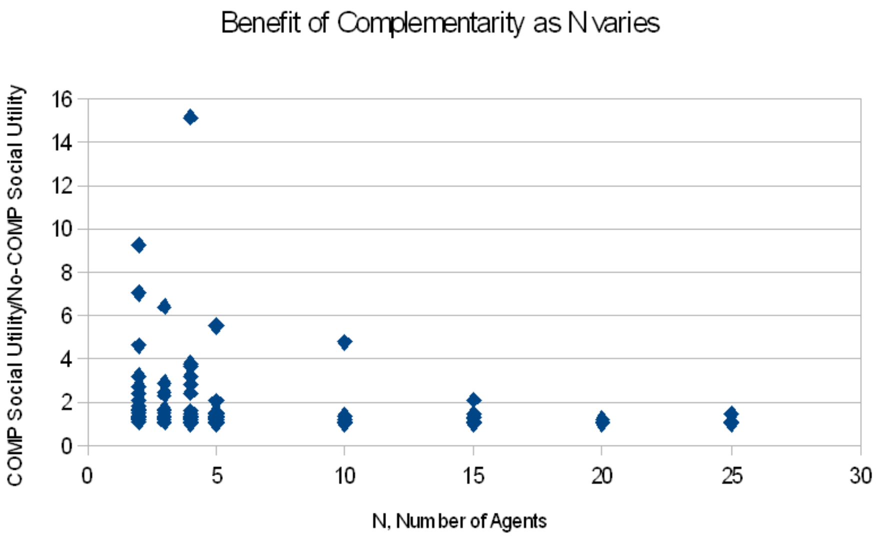

Figure 9: Effect on varying Number of Agents on Complementarity Advantage ratio

We observe higher benefits of complementarity for lower values of agents. We see that the maximum benefits are observed when $N = 4$ and 2. The results indicate that when there are fewer agents involved in the scheme, the advantage of allowing agents to bid on bundles of tasks is more pronounced. When there are many agents, it is more likely that the minimum bidders for individual tasks are different. That is, if there are M tasks, it is more likely that the tasks will be allocated in singletons to different agents (or at least, in small bundles to many agents). When there are few agents, then agents are more likely to get assigned bigger bundles of tasks. The likelihood that the minimum bidders for separate tasks are the same agent is higher in this case.

8.3 Varying the Number of Subtasks

In a similar manner to the above setting, we observed the effect of the benefit of complementarity as M, the number of substasks, changes. We varied M in the range [2,8]. We said the number of agents N was 10 and the number of times T was 12. For each value of M, we ran 40 trials, reinitializing utility functions of the manager and the agents at the beginning of each run. For each trial, we first ran the optimization algorithm for c-values in the range [1,3] (COMP), and next ran the optimization with c-values equal to 1 (No-COMP). We then recorded the net social utility of each allocation at the end of the run, both for the COMP and No-COMP settings, and graphed the complementarity advantage ratio (as described earlier) against M.

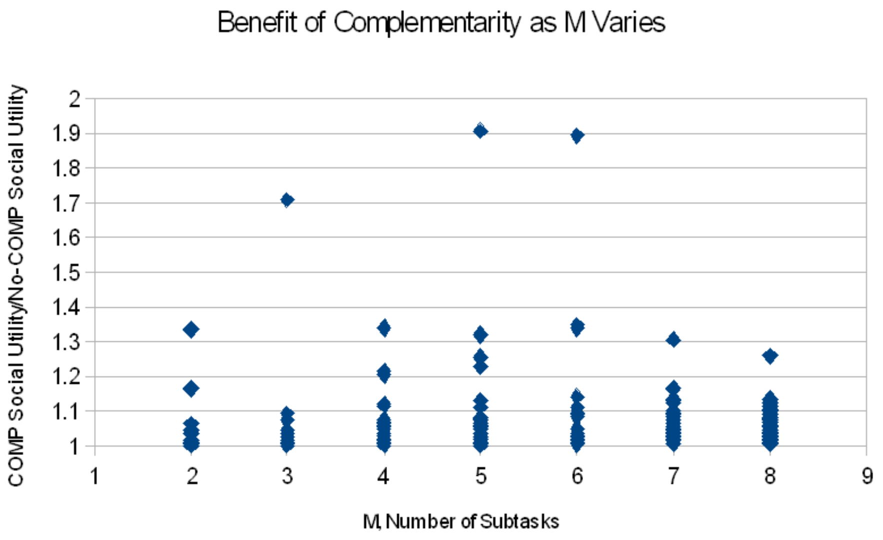

Figure 10: Effect on varying Number of Subtasks on Complementarity Advantage ratio

The graph depicts that as M increases, the complementarity benefit tends to increase. There are some extremely high ratios observed at $M = 5$ and $M = 6$, indicative of this overall trend. Additionally, there are many data points clustered at higher ratios (the number of ponts with high complementarity benefit ratios is increasing as M increases). An explanation for this is as follows. If there is a large number of subtasks, then it is more likely that bigger bundles will get allocated in the optimal outcome. (For example, with M tasks and and N agents, then there will be a bundle of size at least M/N that will be allocated; as M increases with respect to N, this value also increases.) Since the effect of complementarity is more pronounced in bigger bundles, the effect of complementarity is more pronounced.

9 Piecewise Constant Utility Functions

In addition to modeling agents' utility functions as described in Section 7.2, we also modeled agents' utilities using a more discrete piecewise "threshold" function. The threshold valuations $p_i(j,t)$ were as follows:

For agent i, we define his bids for singletons using the formula

$$p_i(j,t) = a_{ij}, \quad t < THRESH_{ij}$$

$$p_i(j,t) = b_{ij}, \quad t \geq THRESH_{ij}$$

where a_{ij} is chosen uniformly at random from $(1, C_{thresh})$ and b_{ij} is chosen uniformly at random from $(0, a_{ij})$ for every (agent, task) pair, and some $THRESH_{ij}$ chosen uniformly between (0,T). The valuations for bundles are then obtained using the $c-bids$ method, for random $c_i \in (1,2)$.

We model the manager's valuation function of time t to be

$$m(t) = M[\frac{C_{thresh}}{N}(1 - t/T) + \frac{C_{thresh}}{2N}(t/T)]$$

(We got this function by linearly interpolating between the expected minimum bids at $t = 0$ and $t = T$.)

We ran the optimizer with the piecewise utilities and the output is pasted below (we made C_{thresh} $= 50$).

```
 Maximum Social Utility for each possible time:

Time: 0    Social Utility: 11.644213098937083

Time: 1    Social Utility: 11.638824221996927

Time: 2    Social Utility: 15.872576130227621

Time: 3    Social Utility: 14.771674753583138

Time: 4    Social Utility: 13.521674753583138

Time: 5    Social Utility: 13.271674753583138

Time: 6    Social Utility: 14.882270005906502

Time: 7    Social Utility: 14.250263512762741

Time: 8    Social Utility: 14.559589055725334

Time: 9    Social Utility: 13.309589055725334

Time: 10    Social Utility: 12.059589055725334

Time: 11    Social Utility: 10.809589055725334


 *******

 Winning Allocation:

Winning Time 2

Bundle: (0,)              Agent Assigned this bundle: 8

His bid for this bundle at this time: 3.0000000000000004

Bundle: (3, 4)            Agent Assigned this bundle: 3

His bid for this bundle at this time: 4.14909862335552

Bundle: (1, 2, 5)         Agent Assigned this bundle: 9

His bid for this bundle at this time: 4.478325246416862



Payments:

Agent 0  gets paid: -0.0

Agent 1  gets paid: -0.0

Agent 2  gets paid: -0.0

Agent 3  gets paid: 6.475295487441215

Agent 4  gets paid: -0.0
Agent 5  gets paid: -0.0

Agent 6  gets paid: -0.0

Agent 7  gets paid: -0.0

Agent 8  gets paid: 3.990306124321119

Agent 9  gets paid: 16.04157583368189

Manager pays:  26.507177445444224

Manager's utility for this time:  27.500000000000004
```

10 Conclusion

In this thesis, we consider the problem of efficient task allocation in the scenario where time is a flexible parameter. Suppose that there is a manager who has a major task to complete, and he has several workers to whom he may assign the task. The workers have the ability to complete the task at different speeds with different utilities. We design an auction based mechanism which incentivizes the workers to reveal true utilities and thus maximizes overall utility of the manager and the workers. Our mechanism was an adaptation of the VCG mechanism to this scenario. We then introduced the notion of complementarity of tasks,which is that some groups of tasks might be easier to complete in conjunction. Using the notion of complementarily, we allowed that agents may bid on bundles of tasks. In order to tackle the winner determination problem, we proposed an algorithm that can find the optimal allocation given agent's bids and manager's valuations over different times. This algorithm is faster than naive brute force search algorithm. We then evaluated the output of this algorithm in a simulation enviroment where complementarity is allowed, and analyzed the results when we varied the measure of complementarity, the number of agents, and the number of subtasks. Thus, we showed that in this new setting with the addition of the time parameter, our mechanism can effectively exploit complementarity. In particular, this results in better profits for the manager and an overall benefit in social utility. Therefore, we conclude that in real life task allocation problems, such mechanisms that are sensitive to time, or more generally, mechanisms that are sensitive to quality, could prove extremely useful.

References

[1] Chi-Kong Chan and Ho-Fung Leung. Multi-auction approach for solving task allocation problem. In Dickson Lukose and Zhongzhi Shi, editors, *PRIMA*, volume 4078 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2005.

[2] Vincent Conitzer and Tuomas Sandholm. Revenue failures and collusion in combinatorial auctions and exchanges with vcg payments. In *Proceedings of the 5th ACM conference on Electronic commerce (EC-04)*, pages 266–267, New York, May 1–8 2004. ACM Press.

[3] Peter Cramton, Yoav Shoham, and Richard Steinberg. *Combinatorial Auctions.*

[4] Panagiotis G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *ACM Crossroads*, 17(2):16–21, 2010.

[5] Benjamin Lubin, Adam I. Juda, Ruggiero Cavallo, Sébastien Lahaie, Jeffrey Shneidman, and David C. Parkes. ICE: An expressive iterative combinatorial exchange. *J. Artif. Intell. Res. (JAIR)*, 33:33–77, 2008.

[6] Parkes, Cavallo, Elprin, Juda, Lahaie, Lubin, Michael, Shneidman, and Sultan. ICE: An iterative combinatorial exchange. In *CECOMM: ACM Conference on Electronic Commerce*, 2005.

[7] David C. Parkes, Jayant Kalagnanam, and Marta Eso. Achieving Budget-Balance with Vickrey-Based payment schemes in exchanges. In Bernhard Nebel, editor, *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)*, pages 1161–1168, San Francisco, CA, August 4–10 2001. Morgan Kaufmann Publishers, Inc.

[8] Simon Parsons. Algorithmic game theory by noam nisan, tim roughgarden, éva tardos and vijay V. vazirani, cambridge university press, 754 pp, £32.00, ISBN 0-521-87282-0. *Knowledge Eng. Review*, 26(1):71–72, 2011.

[9] Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *AIJ: Artificial Intelligence*, 135, 2002.

[10] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *AAMAS*, pages 69–76. ACM, 2002.

[11] Kai Zhang, Emmanuel G. Collins Jr., and Adrian Barbu. A novel stochastic clustering auction for task allocation in multi-robot teams. In *IROS*, pages 3300–3307. IEEE, 2010.