Collaborative Resource Planning with Distributed Agents

Erhan Kutanoglu
Department of Industrial Engineering
University of Arkansas
4207 Bell Engineering Center, Fayetteville, AR 72701
erhank@engr.uark.edu

and

S. David Wu
Manufacturing Logistics Institute
Department of Industrial and Systems Engineering
Lehigh University
200 W. Packer Ave., Bethlehem, PA, 18015
david.wu@lehigh.edu

Abstract

We study collaborative resource planning that arises when resource managers must coordinate their planning with internal or external customers. We envision a setting where geographically dispersed decision makers (agents) coordinate their resource planning using asynchronous, web-enabled mechanisms. We propose the design of such a mechanism where resource allocation is communicated in the form of schedules over which each agent involved has different preferences and financial incentives. Specifically, we design a "schedule selection game" where all participating agents state their preferences via a valuation scheme, and the mechanism selects a final schedule based on the collective input. We address the issue of incentive compatibility, recognizing that independent agents may not reveal their private information truthfully, and they may not behave in a way that is aligned with the overall system efficiency. Based on the principles of Vickrey-Groves-Clarke, we show that the proposed web-based mechanism is a *direct revelation mechanism* that implements the optimal resource allocation under agents' dominant strategies. We analyze main properties of the mechanism such as budget balancedness and individual rationality. We illustrate the operation of the mechanism using a real-world example in coordinating electronics component manufacturing.

1 Introduction

The communication technology and the web are changing fundamentally how firms conduct their core business. Among various e-business applications is the potential to enhance collaborative planning, information sharing, and decision making via cost effective, web-based mechanisms. These developments not only facilitate intra-firm coordination across divisional/functional boundaries, but also inter-firm decisions between customers and suppliers. In either case, the web-based platforms provide universal access across geographically disperse areas where decision makers would be able to communicate internally using corporate intranet, externally using extranet, electronic marketplaces, or the Internet. While these e-business applications are quickly emerging in the software industry, many current developments are simple extensions of the existing ERP paradigm, a paradigm that seeks "total visibility" of system details in a top-down, hierarchical manner. This is accomplished by maintaining painfully detailed information of all perceivable aspects of the organization using sophisticated information and database management systems. This information must be kept up-to-date since it serves as a basis for decision making throughout the system. For large-scale operations involving multiple facilities or firms in a supply chain, this approach encounters major difficulties in scaling up. To realize the full potentials of e-business, new design principles must be formed that address the decentralized, heterogeneous, and asynchronous nature of modern enterprise decision making. A fundamental departure is that in place of a centralized decision entity that collects and dictates solutions, a diverse set of players with distinctly different preferences and incentives must be accommodated and catered to.

In this paper, we intent to take a step toward this direction. We focus our attention on resource coordination and scheduling, although many of the same principles apply to decision coordination in general. We consider a simple setting where multiple agents must coordinate their use of a set of resources. The coordination takes place via an electronic bulletin board or a web site that generates and posts potential resource schedules. The decision makers state their preferences and criteria via software agents. Each software agent votes for its choice of schedules based on its decision maker's preferences. After all agents state their preferences, the system determines the "best" schedule based on a certain mechanism. Our aim is to use this simple *schedule selection game* to characterize the main design principles of a distributed and asynchronous resource planning and coordination mechanism.

While the mechanism itself may be simple, significant difficulties may arise in practice as decision makers choose to "game" the system and rip unfair benefits for their local interests. For instance, a decision maker may state his (untruthful) preferences in such a way that more resources, or scarce resources at a particular time, are allocated to him. This destroys credibility of the mechanism while introducing instability that ultimately leads to inefficiency. Thus, the main focus of our analysis is on the design of *incentive compatible* mechanisms, mechanisms that not only diminish agents' incentive to lie, but also provide incentives for agents to work toward the common good, i.e., system efficiency.

These issues have been examined in other multi-agent systems where it is crucial to consider the perspective of each agent and their economic incentives. Without proper incentives, agents may not perform in a way that is aligned with the overall system goals. To ensure *incentive compatibility* we need a reward mechanism for agents that would align their individual (local) interests with the system-wide (global) objectives. To this end, we refine the description of the *schedule selection game* as follows: the system proposes resource allocations for a particular period of time by posting a number of alternative resource schedules (e.g., specifying which *resources* will be allocated for what type of *jobs* over which particular *time periods*). Agents competing for the resources for their jobs have private information concerning the true benefits of each schedule to the job types they represent. Agents are to provide a valuation of each schedule and this information will be ultimately used to identify a final schedule. Since each schedule allocates resources differently, agents could have drastically different preferences over the schedules under consideration. As discussed earlier, there is no guarantee that each agent's reported valuation will be truthful as some may benefit from reporting a misleading valuation. A primary issue is to design incentives that would elicit agents' true preferences so that the overall (true) best schedule could be selected. We will outline a mechanism that identifies the optimal solutions regardless of the fact that individual preferences are known privately by each individual, while various forms of manipulations or misrepresentations are possible.

Designing an incentive compatible procedure in such an environment is known as the *mechanism design problem*, a subject of extensive study in the microeconomics and game theory literature [17]. We will focus on a specific subarea of this literature known as the *direct revelation mechanisms*. Similar problems are studied in many different contexts and applications. The origin of all these studies is the "optimal control problem" of a central authority who tries to control the agents in an organization. The problem was first posed and studied as an incentive problem by Groves in the early seventies [9, 10]. A reflection of this mechanism on a resource allocation problem in a decentralized organization can be found in [13]. These procedures have been referred to as *Groves mechanisms* or *Groves incentive schemes*. In a different context, Groves studies the problem of coordinating economic agents' decisions considering external effects [11]. This line of research goes back to Vickrey's seminal work on "counterspeculation" and design of second-price auction mechanisms [22]. Although there are many applications of Groves mechanisms in the economics literature (see, e.g., Brewer and Plott [3]), applications to modeling and optimization did not exist until more recently. A recent implementation of this sort considers an incentive-compatible budget allocation problem for site decontamination projects [5].

Groves [12] proposes a related but different approach around the same time he developed Groves mechanisms. In this study, the presence of central control is no longer needed. He analyzes a collective "public choice" problem where agents try to implement a "socially efficient" (or optimal) public project. In this paper, we follow an approach closely related to this latter line of research where the mechanism involves collective decision making and self-coordination of job agents rather than a

center-based control approach (we provide a brief discussion on the latter issue in Section 4). As common in all previous incentive compatibility studies, we will assume the existence of a transferable commodity which is usually represented by "money." We will consider self-interested agents who try to maximize their local (net) utility as a combined function of their original utility and the amount of reward (money) that the mechanism allocates. We interpret the original utility as "profits" that an agent makes when a certain schedule is implemented. Without loss of generality, this assumption simplifies the discussion of the mechanism since the original utility and the transfer have the same unit ("money" or dollars). In Section 4, we discuss the implications and possible alternatives of this assumption. In Section 2, we first outline a direct revelation mechanism for the schedule selection game, and then discuss various properties and relevant game theoretic results such as the dominant strategy equilibrium. We then investigate related issues such as the generation of candidate schedules and the quality of the selected schedule through the mechanism. We give an example to illustrate a typical implementation of the approach in a real-life scheduling environment in Section 3.

Using a multi-agent (or distributed) approach for decision making problems including resource planning and scheduling is not new. For example, Sandholm [20] provides an overview of several mechanisms such as Groves-Clarke mechanisms and bargaining in artificial intelligence-based distributed problem solving. He explains the Groves-Clarke mechanism in the context of voting. He also uses a task allocation problem as an example of Contract Nets where resource agents interact to accept or reject individual contracts that allocate tasks to agents using a series of requests for bids and awards. Similarly, Nisan and Ronen [18] gives a scheduling example to illustrate the incentive issues in Vickrey-Groves-Clarke (VGC) mechanisms. Their scheduling problem focuses on single-stage jobs on multiple, unrelated parallel machines. Again, agents represent machines and the main goal is to devise mechanisms that elicit true processing time (or speed) information from machine agents so that a machine-job allocation with minimal makespan is achieved.

In all these studies one common assumption is that there are multiple resources (machines or processors) that are capable of performing jobs (tasks) and the agents that represent these resources compete to win the jobs' requests. This underlying assumption is the driving force behind the machine-agent-based mechanisms whether they are contract nets or generalized VGC mechanisms. A study that is closer to our study is conducted by Suijs [21] who uses a single-machine weighted flow time scheduling problem to illustrate the difficulty of balancing the budget when a VGC mechanism is used. In this example, the resource agent knows the job processing times, but the weights (or costs) are private information of the job agents. The VGC mechanism is designed to extract cost information from the job agents. Our study contributes to the existing literature by studying a more general case of multi-stage, multi-machine environment with a set of pre-generated candidate schedules. Instead of resource agents competing to win job requests, we examine the case where job agents compete for scarce resources, as is the case in most manufacturing environments. We also discuss open research questions of the scheduling mechanism such as budget balancedness, asynchronous decision making, and optimal

schedule generation. We illustrate the use of our proposed mechanism in a web environment using a numerical example drawn from an industry case study.

2 The Schedule Selection Game

We first define some notations used throughout the chapter: For a generic vector n-vector x, x_i denotes the ith element of the vector, i.e., $x = (x_1, \ldots, x_n)$, and x_{-i} denotes $n-1$-vector of the elements of x without x_i, i.e., $x_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. We use this notion of "others" $(-i)$ to represent the other agents' actions/decisions with respect to the agent under consideration, agent i. Similarly, for a given set $X \equiv \cup X_i$, we will use X_{-i} to denote the set X without X_i. Thus, we can interpret $-i$ in the subscript as the original vector or set without the ith component, i.e., $-i \equiv \mathcal{N} \setminus \{i\}$.

The schedule selection game can be described as follows. There is a set of job agents $\mathcal{N}$, indexed by $i = 1, \ldots, N$, and a set of feasible shop schedules Y known to all agents. Each schedule is a complete and feasible job-shop schedule where each job receives a particular block of resource time for each of its operations. A job agent represents one and exactly one job in the schedule (the analysis of this to the case where each agent is responsible for more than one job is possible and left for future research). The impact (valuation) of any particular schedule to a job agent is private information only known to the agent. The job agents must collectively choose an *outcome* consisting of (1) a shop schedule y from the set of feasible schedules Y, and (2) *transfers* of a numeraire commodity ("money") from/to all agents, $\{\delta_i\}_{i \in \mathcal{N}}$. Here δ_i denotes the amount of money transferred to (if positive) or from (if negative) agent i. We assume that the preferences of each job agent over the outcomes are summarized by a utility function u_i. We also assume that an agent i only cares about the transfer δ_i and the profit π_i he receives from the schedule y:

$$u_i(y, \delta_i) = \pi_i(y) + \delta_i. \tag{1}$$

Note that the utility u_i of agent i does not depend on other agents' transfers, δ_{-i}. Thus, agent i prefers outcome (y, δ) with schedule y and transfer vector $\delta \equiv (\delta_1, \ldots, \delta_N)$ to another outcome (y', δ') if and only if $u_i(y, \delta_i) \geq u_i(y', \delta'_i)$. For a classical JSP with total weighted tardiness objective, the profit π_i of an agent i can be interpreted as negative weighted tardiness costs that is converted to monetary units. In the following discussion, we will use the generic term "profit" while keeping in mind that it can represent a variety of objectives including weighted tardiness. Simply stated, the schedule selection game is a decision procedure (a mechanism). Through the use of proper transfer functions (incentives) agents will always reveal their true preference while the collection of all agents' decisions correspond to the best solution possible for all.

2.1 The Mechanism Design Problem

We now discuss several important properties of the *schedule selection game* using a game theoretic framework. We assume that a job agent's preferences are known only by the agent himself. In a manufacturing context, the preferences may be motivated by delivery requirements or contractual agreement specific to the job, the performance evaluation a product manager is subject to, or any other constraints that may be considered "local." Any decision procedure that is responsive to individual preferences will require some type of communication or signaling of preference information. The decision mechanism uses the communicated information to choose the "best" schedule among the alternatives in Y.

We first define the schedule selection game as an N-person noncooperative game with incomplete information. A *mechanism* Γ is a collection $(\{M_i\}_{i\in\mathcal{N}}, h(\cdot))$ where M_i is a set of alternative messages μ_i for agent i, and $h(\cdot)$ is an *outcome function* or *allocation rule*. Function $h(\cdot)$ specifies for every joint message $\mu \equiv (\mu_1, \ldots, \mu_N)$, (1) a collective schedule y, and (2) a transfer profile δ, that is $h(\mu) = (y, \delta)$. In the following discussion, we separate the outcome function into two parts, i.e., $h(\mu) = (y(\mu), \delta(\mu))$, where $y(\cdot)$ and $\delta(\cdot) \equiv (\delta_i(\cdot), \ldots, \delta_N(\cdot))$ denote the *schedule selection function* and *transfer function*, respectively. Thus, we have a game composed of N players corresponding to the job agents. The strategy set of player i is the message space M_i of agent i, and the preferences of player i over joint strategies μ (messages) are defined by the utility function u_i and by the outcome function $h(\cdot)$, i.e., player i prefers μ over μ' if and only if $u_i(h(\mu)) \geq u_i(h(\mu'))$. This defines an N-person non-cooperative game of incomplete information since each player's preferences are known only to himself.

Typically, in a game theoretic framework, the goal of a mechanism is to choose a particular outcome using the outcome function $h(\cdot)$ for a particular realization of agents' utility functions, say $\{\pi_i\}_{i\in\mathcal{N}}$. From a system optimization point of view, a desirable result would be to choose an *optimal*, or *socially efficient*, schedule y^*, i.e., find a schedule y^* in Y such that $\sum_i \pi_i(y)$ is maximized (Note that here "optimal schedule" is equivalent to "socially efficient schedule" which is different when the auction mechanism design is considered where "optimality" and "efficiency" differ in their objectives).

We say that mechanism Γ with outcome function $h(\cdot) = (y(\cdot), \delta(\cdot))$ *implements* an optimal schedule y^* if there is an *equilibrium* joint message μ^* of the game induced by Γ whose outcome is y^*. Note that the implementation of a particular outcome depends on the equilibrium concept used.

Since the specific form of the outcome function is a mechanism design choice, we seek outcome functions that implement a particular schedule, more specifically an optimal one. From this point of view, we are particularly interested in *direct revelation mechanisms* which normally involve two stages: (1) In the first stage, each agent is asked to reveal his true profit from the schedules, i.e., each $\mu_i(y)$ is the value reported by i that supposedly represents the profit obtained by agent i if y is selected. (2) After all agents report their values, the outcome function $h = (y(\cdot), \delta(\cdot))$ chooses the

maximizer of the total reported profit, i.e.,

$$y^*(\mu) = \arg\max_{y \in Y} \sum_i \mu_i(y), \tag{2}$$

as if all agents have reported their true profits, and the payments are made according to transfer function $\delta(\mu)$.

Note that if all agents truthfully report their profits, then the schedule selection function (2) finds an optimal schedule y^* in Y. In fact, this requirement of having the agents communicate true profits is the essence of direct revelation mechanisms. However, because of the dispersed information and individual preferences over schedules, it may not be in any agent's best-interest to "tell the truth". For example, by reporting falsely his valuations of the schedules y, an agent may be able to secure a larger transfer than necessary to compensate him for any change in y that his false reporting causes.

Thus, without a proper *incentive scheme* that induces "truth telling" behavior, an agent may lie about his profits, with the intent to maximize his local utility, $u_i(y^*(\mu))$. Hence, the primary focus in designing direct revelation mechanisms is on constructing a proper incentive scheme. Such schemes are characterized by carefully designed *transfer functions*, $\delta(\cdot)$, which is a part of the outcome function. With such an incentive scheme, the agents voluntarily report their true profits. To formally study the aforementioned issue of *incentive compatibility*, i.e., inducing truth telling as an equilibrium behavior, we need to define a specific equilibrium for the game induced by the direct revelation mechanism. In the following, we summarize the concept of dominant strategy equilibrium for the mechanism and analyze its properties.

2.2 Dominant Strategy Equilibrium

Dominant strategy equilibrium states that each player plays his best strategy that is individually best for him regardless of the strategy chosen by any other players.

Definition 1 ***(Dominant Strategy Equilibrium)*** *For mechanism Γ with message space $\{M_i\}_{i \in \mathcal{N}}$ and outcome function $h(\cdot)$, a joint message $\mu^* = (\mu_1^*, \ldots, \mu_N^*)$ constitute a dominant strategy equilibrium if, for every i,*

$$u_i(h(\mu_i^*, \mu_{-i})) \geq u_i(h(\mu_i, \mu_{-i})) \; \forall \; \mu_i \in M_i, \; \forall \; \mu_{-i} \in \mathcal{M}_{-i} \tag{3}$$

where $\mu_{-i} \equiv (\mu_1, \ldots, \mu_{i-1}, \mu_{i+1}, \ldots, \mu_N)$ and $\mathcal{M}_{-i} \equiv \prod_{i' \neq i} M_{i'}$.

Thus, if a dominant strategy equilibrium exists for a mechanism, then each player would play his dominant strategy regardless of other agents actions. For a direct revelation mechanism with proper incentives, this means that every job agent has incentive to report his true valuation of the schedules no matter what the other agents report since doing so maximizes his own (true) utility, i.e., $\mu_i^*(y) = \pi_i(y) \; \forall y \in Y$.

Assuming the schedule selection (2) of an outcome function, agent i, for every y, solves the utility maximization problem

$$\mu_i^*(y) = \arg\max_{\mu_i \in M_i} u_i(y^*(\mu_i, \mu_{-i}), \delta_i(\mu_i, \mu_{-i})) \tag{4}$$

or

$$\mu_i^*(y) = \arg\max_{\mu_i \in M_i} \pi_i(y^*(\mu_i, \mu_{-i})) + \delta_i(\mu_i, \mu_{-i}) \tag{5}$$

for any $\mu_{-i} \in \mathcal{M}_{-i}$. Note that through messages the agent is able to affect both the final schedule selection and his transfer which in turn influence the total payoff to agent i. As mentioned above, it is the goal of the transfer function $\delta(\cdot)$ to provide incentives to the agents to reveal their true valuations regardless of what the other agents report.

Consider the case where transfers are all zero, i.e., $\delta(\mu) = 0 \ \forall \ \mu \in \mathcal{M}$. In this case, $u_i(y, \delta_i = 0) = \pi_i(y)$. Thus, without transfers, each job agent will try to maximize his original profit when he is asked to report his valuation of the schedules. As a self-interested agent, job agent i will not tell the truth by using the following reporting rule (assuming minimum profit of 0 from any schedule):

$$\mu_i^*(y, \delta_i = 0) = \begin{cases} L_i & \text{if } y = \arg\max_{y' in Y} \pi_i(y') \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

where L_i is a (large) number which may or may not be equal to true value $\pi_i(y)$. Hence, without transfer, the agents tend to inflate the profit for their local best schedule, and report minimum possible profit (0) for all others. Using these reported values, it is clear that the outcome function as defined in (2) and $\delta(\cdot) = 0$ cannot implement an optimal schedule. In fact, the following proposition is a special case of a result known as Gibbard-Satterthwaite Impossibility theorem [12, 17].

Proposition 1 *A direct revelation mechanism* Γ *with schedule selection function defined in (2) cannot implement an optimal schedule in dominant strategies if transfers are not allowed.*

In fact, a specific form of transfer function known as *Groves-Clarke incentive scheme* (Groves [10, 12], and Clarke [4]) is sufficient to guarantee the implementation of an optimal schedule in dominant strategies as outlined in the following proposition (Note how the net utility functions of the agents and the joint profits are aligned through schedule selection and transfer functions).

Proposition 2 *A direct revelation mechanism* $\Gamma = (\{M_i\}_{i \in \mathcal{N}}, h = (y^*(\cdot), \delta^*(\cdot))$ *with schedule selection function defined by (2) implements an optimal schedule in dominant strategies if, for all* $i \in \mathcal{N}$*,*

$$\delta_i^*(\mu) = \sum_{i' \neq i} \mu_{i'}(y^*(\mu)) - \sum_{i' \neq i} \mu_{i'}(y_{-i}^*(\mu_{-i})) \tag{7}$$

where $y_{-i}^*(\cdot)$ *denotes the outcome that would be selected without agent* i*'s messages, i.e.,*

$$y_{-i}^*(\mu_{-i}) = \arg\max_{y \in Y} \sum_{i' \neq i} \mu_{i'}(y) \tag{8}$$

Proof: See Appendix A.

Note that any joint message μ with $\mu_i(y) = \pi(y) + L_i \ \forall i$ where L_i is now a constant does not change the selection of an optimal schedule. Hence, the above proposition holds even if the agents increase or decrease their respective profits by a constant amount. A direct revelation mechanism Γ with outcome function $h(\cdot) = (y^*(\cdot), \delta^*(\cdot))$ (see equations (2) and (7) respectively) is known as *Groves-Clarke mechanism* ([9, 12, 4]. Since the G-C mechanism can be viewed as a generalization of Vickrey's second price auction, it is sometimes called Vickrey-Groves-Clarke (VCG) mechanism (c.f., Krishna and Perry [14]). Here, the second term on the right hand side can be interpreted as a charge equal to the total profit that would be possible if agent i did not participate in the decision making process at all (or equivalently, if he reported indifference, $\mu_i(y) = 0$ for all i).

Note that agent i's transfer δ_i in a Groves-Clarke mechanism is zero if his announcement does not change the schedule selection relative to what would be optimal for agents $\mathcal{N} \setminus \{i\}$, i.e., if $y^*_{-i}(\mu_{-i}) = y^*(\mu)$. It is negative if his participation causes the others to choose a different schedule, where the agent is called *pivotal*. In this case, he pays the difference he makes on the others' profits. In a way, the payment represents the *externality* that i exerts on the other $I - 1$ agents by his (and his job's) presence in the process. Thus, in Groves-Clarke mechanism, all agents end up paying zero or some positive amount of money (another AI-oriented application of Groves-Clarke mechanism can be found in Sandholm [20]).

We showed that if the agents use a direct revelation mechanism with the outcome function defined by (2) and (7), they will have a strong incentive to report their true valuations of the schedules, be able to coordinate their individual preferences, and select an overall optimal schedule. However, we should note that there are many issues to cover for a successful implementation of such a direct revelation mechanism. We will analyze some of these in the following sections.

2.3 Participation and Asynchronous Decision Making

In the previous sections, we showed that each agent has an incentive to truthfully reporting his local profit when asked. However, we did not question if the agents have an incentive to *participate* in the "schedule selection" process in the first place. We implicitly assumed that all agents are required to participate in the game without other alternatives. If the agents are free to participate while full participation is necessary for overall system performance, then the *individual rationality* or *participation* constraints should be considered in the design stage so that the agents are provided with incentives to participate in the process.

We first analyze if total participation is necessary for the ultimate selection of the optimal schedule (in Y). This is also an important issue if the scheduling game is to be implemented asynchronously, say on an Internet environment. We first note that even if an agent chooses not to participate, the schedule selected without his participation will include his job. By design, a schedule selected with full

or partial participation will be a complete and feasible schedule inclusive of all operations of all jobs. Since the agents' strategies are defined as messages, not as individual scheduling decisions, the only part missing from a nonparticipant is his messages. In this case, the agent does not pay or receive any transfer but is bound to accept the schedule that other agents collectively choose without his input. Thus, we treat the non-participation of an agent as the case where he reports zero profits for all schedules in Y, i.e., he is indifferent for all schedules.

This fact allows us to analyze a form of asynchronous decision making in the schedule selection game. For instance, the schedule selection function can be activated any time to find the current best schedule with potentially partial messages. Without waiting for all of the messages from all of the agents, the outcome function can select a schedule with respect to the currently reported values from currently participating agents. More importantly, once having submitted his message, there is no need for an agent to revise his message, regardless of which other agents will end up participating. This results in two properties potentially useful for asynchronous applications: (1) the agents do not have to report their profit functions simultaneously, and (2) the agents do not have to report their profits for all the schedules, i.e., they may report on a subset of the schedules.

Before starting a formal analysis, we recall that with a Groves-Clarke mechanism if *all* agents report their profits for *all* schedules, an optimal schedule in set Y will be selected. We define this as the *full participation case* in which y^* is the selected schedule. Consider also an alternative, *partial participation case*, where a subset of agents do not participate (by choice, due to communication delays, or early activation of allocation function) while the rest report for *all* of the schedules. To analyze this case in more detail, let I and I' be a partition of set $\mathcal{N}$, i.e., $I \subset \mathcal{N}$, and $I' = \mathcal{N} \setminus I$.

Consider first the case where agents in I' participate the game, and collectively select a schedule y' which might potentially be different from y^*:

$$\sum_{i' \in I'} \pi_{i'}(y') \geq \sum_{i' \in I'} \pi_{i'}(y) \; \forall y \; \in Y \tag{9}$$

This is also valid for $y = y^*$. From the optimality of y^* for all agents, we also know that

$$\sum_{i \in \mathcal{N}} \pi_i(y^*) \geq \sum_{i \in \mathcal{N}} \pi_i(y) \; \forall y \; \in Y. \tag{10}$$

Writing inequality (10) for $y = y'$ and splitting the summations on both sides into partitions I and I', we obtain:

$$\sum_{i \in I'} \pi_i(y^*) + \sum_{i \in I} \pi_i(y^*) \geq \sum_{i \in I'} \pi_i(y') + \sum_{i \in I} \pi_i(y') \tag{11}$$

From inequalities (9 for $y = y^*$) and (11), we can obtain

$$\sum_{i \in I} \pi_i(y^*) \geq \sum_{i \in I} \pi_i(y') \tag{12}$$

This means that when $\mathcal{N} = I \bigcup I'$ collectively select y^*, if agents in I' collectively select $y' \neq y^*$ then I as a group must choose y^*. Hence, if a subset of I does not participate the game, the selection by

participating I' might be suboptimal. On the other hand, the fact that the selected schedule would be optimal when participating group is I shows it is possible to obtain an optimal schedule without full participation.

Consider agent i' in I that is pivotal in full participation case, i.e. without agent i', the rest would choose some schedule $\tilde{y}$ instead of y^*,

$$\sum_{i\in\mathcal{N}:i\neq i'} \pi_i(\tilde{y}) \geq \sum_{i\in\mathcal{N}:i\neq i'} \pi_i(y) \; \forall y \; \in Y. \tag{13}$$

In particular, this inequality is true for $y = y'$ and $y = y^*$. Writing this for $y = y'$ and splitting the summation on both sides to the subsets I and I' lead to the following:

$$\sum_{i\in I:i\neq i'} \pi_i(\tilde{y}) + \sum_{i\in I':i\neq i'} \pi_i(\tilde{y}) \geq \sum_{i\in I:i\neq i'} \pi_i(y') + \sum_{i\in I':i\neq i'} \pi_i(y') \tag{14}$$

Observing that i' is not in I' and that I' collectively selects y' (Inequality 9), we obtain

$$\sum_{i\in I:i\neq i'} \pi_i(\tilde{y}) \geq \sum_{i\in I:i\neq i'} \pi_i(y') \tag{15}$$

which means that $I \setminus \{i\}$ selects schedule $\tilde{y}$ which could be different from y' and y^*. This in turn leads to conclude that i' also becomes a pivotal agent in partial participation case when the participating subset is I.

A similar analysis for an agent that is not pivotal in full participation case reveals that the agent is also not pivotal in partial participation case with participating group of I. Hence I without non-pivotal agent i' would still choose y^*.

Although this *ex post* analysis is useful, in practice the schedule selected without some critical agents may be suboptimal, i.e., $y^*(\mu_{I'} = \pi_{I'}) \neq y^*(\mu = \pi)$ when participating agents I' collectively choose a schedule different from y^*. Hence, for the full "utilization" of the mechanism (to select the best schedule in Y), we need to guarantee participation from a set of critical agents. In general, as mentioned above, the mechanism designer needs to consider *individual rationality* or *participation* constraints to have (all) the agents participate the process. Thus, an efficient mechanism must guarantee that agents will not be worse off by participating in the mechanism. Since we cannot discriminate agents, we cannot consider the participation constraints only for certain agents, i.e., the constraints must be considered for all the agents. One common assumption is that non-participants are not rewarded or charged any transfer.

Normally, these constraints take the form of

$$u_i(y, \delta_i) \geq IR_i(\cdot) \tag{16}$$

where IR_i is the "break-even" payoff of agent i which in general can be a function of his profit function, outcome function, etc. In its simplest form, IR_i can be 0, for example, if we assume that every agent's

utility from not participating is 0 (In the scheduling context, this can be the case where the agent's job is left to the next planning horizon, where the profit from the schedule can be normalized to lowest possible profit 0). Another simple form is to have IR_i depend on a default schedule that would be chosen when there is no full participation (coordination). If we denote this by $\bar{y}$, then

$$IR_i(\bar{y}) = \pi_i(\bar{y}). \tag{17}$$

This means that, for every agent, the mechanism should guarantee a positive gain from participation with respect to the "uncooperative" solution. In a more general case, IR_i can depend on the outcome function, y^*. This might be the result of externalities exerted by participants on non-participants, which is definitely a case in our setting. Hence, if each agent is to consider a "worst case scenario" for IR_i, i.e., the profit from a schedule that can be selected by the others without his input, the participation constraints take the form

$$u_i(\cdot) \geq \pi_i(y^{'}_{-i}(\mu_{-i})) \tag{18}$$

where $y^{'}_{-i}(\mu_{-i})$ denotes the schedule selected without i's messages that would possibly minimize his profit.

There are different levels of implementation of the individual rationality constraints, such as *ex ante*, *interim,* and *ex post* individual rationality [17]. In general, satisfying all IR conditions at the same time may not be possible. We can show however, some interim and ex post participation constraints are readily satisfied by using the Groves-Clarke mechanism. For example, if we let $y^{'}_{-i}(\mu_{-i})$ to be the schedule selected by the other agents in the dominant strategy equilibrium, i.e.,

$$y^{'}_{-i}(\cdot) = y^*_{-i}(\cdot), \tag{19}$$

then we can show that *ex post* IR constraints are satisfied. To show this, we only need to prove that the payment by agent i in equilibrium does not exceed the potential increase in his profit between the schedule selected by the other agents and the optimal schedule, i.e.,

$$-\delta^*_i(\pi) \leq \pi_i(y^*(\pi)) - \pi_i(y^*_{-i}(\pi_{-i})). \tag{20}$$

Satisfying the more demanding participation constraints is not obvious, and there has been some negative results reported in this regard [12]. This is magnified when the agents have a large negative impact on other agents' profits as in our case: Job agents compete for resources to finish early, and in general, a high profit for a job might mean a low profit for another. Thus, an effect of cooperation might be to choose a schedule that significantly reduces an agent's (before transfer) profits from what the agent could achieve when there were no cooperation. In this case, if the participation is voluntary, then the agent would not have an incentive to participate in the mechanism. One possible solution of this difficulty is to constrain the mechanism such that the agents have positive benefits from cooperation in long-term expectation. This means that the agents will gain from participation

in the long run over repeated runs of the mechanism. See [11] for an example. We will leave this subject for future study.

2.4 Budget Balance

An issue in transfer-based mechanisms is to have the total transfers satisfy the budget balance constraint. For example, one mechanism designer may need to have a nonnegative total transfer or non-positive total transfer, or even more demanding, zero total transfer, i.e., fully balanced budget. In a centerless Groves-Clarke mechanism such as ours, one reasonable requirement might be that no outside source of financing is allowed, i.e.,

$$\sum_i \delta_i \leq 0, \tag{21}$$

This is easily satisfied by the Groves-Clarke transfer function outlined above. In this case, all agents are charged some nonnegative amount, usually total transfer being strictly negative. A more demanding requirement is that of having the total transfer zero. This requirement correspond to a *fully balanced budget,* denoted by the constraint

$$\sum_i \delta_i = 0. \tag{22}$$

Note that, in this case, some δ_i will have to be positive, i.e., some agents will be paid a positive amount. Since the total monetary transfer is zero, we can interpret this as having some agents pay other agents; but there is no fund enters or leaves the system. However, it has been shown that, in general, there is no direct revelation mechanism that both implements an optimal solution in dominant strategies and satisfies balanced budget constraint for every possible message profile [8]. Thus, the presence of private information means that the agents must either accept some waste of funds ($\sum_i \delta_i \leq 0$) as in the Groves-Clarke transfers, or give up on finding the optimal schedule at all times. If Bayesian incentive compatibility (as opposed to dominant strategy incentive compatibility) is acceptable for mechanism design in this context, then it is possible to devise a transfer function that will induce incentive compatibility on the expected values. Such mechanism is first proposed by Arrow [1] and d'Aspremont and Gerard-Varet [6] (AGV mechanism that does not guarantee IR). This is recently refined by Krishna and Perry [14] to include individual rationality.

A rather simple way to overcome the budget balancing problem in dominant strategies is to introduce a center agent whose preferences are known, or who has no preferences over the schedules. This center agent's utility might be set as $u_0 = t_0(\mu)$ which summarizes that he only cares the transfer but not the schedule selection. In this case, if we set $\delta_0^*(\mu) = -\sum_{i \in \mathcal{N}} \delta_i(\mu)$, then the Groves mechanism both satisfies the balanced budget requirement and selects the optimal schedule. The center agent collects all the payments from the agents which guarantees a nonnegative profit for him. In fact, we can interpret the center agent as one who designs and implements the mechanism, while managing the schedule selection and the transfers among the job agents.

3 An Industry Case and Illustrative Example

3.1 Background

In this section, we use an industry case study to illustrate how the schedule selection game outlined in the previous section can be implemented in a web-enabled environment. We base our example on real scenarios and data collected from an electronics assembly plant and their customers. The plant makes electronic components for automotive manufacturers worldwide. These components include electronic engine control (EEC) modules, anti-lock brake systems (ABS), and others. Products made in the plant are organized by product families, each having a *product manager* who not only oversees the production of her products, but also keeps close communication with her customers and suppliers. By nature of the business, each product manager has a distinctly different set of constraints and preferences driven by delivery arrangements, demand characteristics, quality and technological specifications, and other contractual agreements with the customers and suppliers. Nonetheless, the product managers must compete for a set of common resources while trying to maximize her own interests and preferences. In our example, the shared resources are a collection of automated production lines called the consolidated surface mount device (SMD). Approximately 80% of all electronic products in the plant go through this area, where a pick and place process using a series of SMD machines and testing stations form the main operation. The focus has been on maximizing the efficiency of the capital intensive SMD area while aligning plant-wide objectives with product level requirements so as to maintain responsiveness to customers.

The current system relies on a centralized scheduling system which collect all relevant information from the product managers and store this information in a scheduling database. For the most part, the scheduling software ignore the preferences of the product managers and considers only hard constraints such as the availability of raw material or subassemblies. A schedule with a three-week horizon is generated. The first-week schedule is implemented while the remaining two-weeks are overridden by the three-week schedule generated in the following week. It is a frequent complaint by the product managers that this system is too rigid to take into consideration preferences and special considerations that would better serve the customers. The latter is typically handled by expediting orders when a special mandate is given to speed up a certain customer order. The process of ad hoc expediting diminishes the benefit of systematic schedule optimization, among other unintended drawbacks. In spite of significant investment in the scheduling software, the overall customer satisfaction is low. In several occasions, the plant lost major customers due to its inability to compete with smaller more flexible operations elsewhere.

A scheduling system which takes the customers' preferences into consideration (via the product manager who represent the customer, or directly from the customers) is desperately needed. A web-based platform would be ideal in this situation but the system must be sufficiently transparent such that the decision makers involved understand the basic operation of the system. Further, the system

must accommodate asynchronous decision making if possible in that players do not need to show up at the same time, and the exact point in time they participation (before a certain deadline) does not influence the outcome significantly. The *schedule selection game* have desirable properties in this regard (although not perfect) in that the system is relieved from collecting detailed specific information concerning the customer preferences, but instead proposes a large number of feasible alternatives (from the resource utilization point of view), and allowing the decision makers to assess the merit of the alternatives based on their preferences. This mode of operation is similar to that of an open auction, where the players communicate their private information via a simple valuation. The process is asynchronous as *when* the valuation is provided do not influence the schedule selection so long as it is provided within a certain deadline.

3.2 A web-based Implementation

To illustrate the significance of private information and local constraints a job agent maintains, the issues surrounding order due-dates provide good examples. Order due-dates are at best moving targets subject to frequent revisions from the customer (via electronic schedule releases, or informal off-line communications). Worse, due-dates are often coupled with business issues such as pricing, potential discounts, consolidation of freights due to transportation or container restrictions, future order status, long-term good will, and inventory restrictions. In other words, behind a simple due-date performance measure there is a significant amount of private information only known to the product manager in charge. As one of the product managers stated, the managers may spend more than half of his/her time dealing with these customer related issues. This does not even include other product-specific issues related to component suppliers, transportation service providers, or the policies and reward systems specific to the facility, all of which influence a product manager's decisions.

We do not expect to capture all of these issues in an informal and substandard process such as the current practice used in the plant. Our suggested solution for this and related incentive issues is to implement the scheduling game-like environment using a web-based interface. The environment would include (1) a common database that would keep the resource requirements for each order/job and other relevant data, (2) a software engine that would reside on a host machine and generate the candidate solutions to post on the electronic bulletin board on the web, and (3) a web-based user-interface on local computers that can be customized depending on the manager's preferences. A visual representation of the system is depicted in Figure 1. Knowing that the schedule selection game has nice theoretical properties, we see the main implementation issue is to provide the right incentives to the managers and let them play the computerized version of the game. The reflection of that in the plant execution is to induce some form of "transfer" among the managers, for which we need to define "transferable commodity." Leaving the explicit discussion of the social and ethical issues of introducing monetary transfers among the managers for later, we will focus on the web-based

implementation here, which we explain below.

First, the software engine generates alternative resource schedules using the common database that provides resource requirements for each job. Each schedule has all the jobs of all the managers, i.e., each schedule is complete and feasible. We can think of these candidate schedules as collection of Gantt charts, each of which in turn is a feasible combination of job schedules. Assuming that each product manager is responsible from a distinct subset of jobs, the web-based user interface for a specific product manager displays the relevant portion of each schedule. That is, each product manager is able to see how the resources are allocated to their jobs in each of the alternative candidate schedule. For instance, Figure 2 presents how such a view can be constructed from the second agent's allocations in eight different schedules generated in the numerical example below. Not only hiding others' schedules is desirable for information privacy purposes, but also a manager need not to see the allocations of other jobs since each manager's performance depends only on his jobs' allocation and his transfer.

The managers (or possibly software agents that represent the managers) then provide, through the web-based interface, their own valuation for each schedule using their own criteria based on their private information. When all job agents provide the necessary valuation, the engine selects the schedule with the maximum total reported utility as an outcome of the mechanism, and the transfers are calculated and executed. (In the following example, to simulate this process, we use the negative value of weighted tardiness for all utility measures, or equivalently, we minimize weighted tardiness as "disutility").

Note that the effectiveness of the proposed web-based mechanism depends on the efficiency of the inter-agent communication for collecting and communicating schedule valuations, especially if the schedule selection has to be done quickly. First, there might be many schedules to be evaluated by the agents although we use 8 schedules in the following example that follows to illustrate the process with a manageable size. Second, there are software design and engineering issues that have to be addressed to facilitate efficient communication among the agents. We can list the following issues that have to be addressed in a successful web-based implementation:

- One has to address how the human decision makers will interact the overall mechanism. Mainly, how sophisticated will the software agents be and how is the human-software agent interface handled?

- One must also address the database issues: Where and in what format will the common database be held? Is the inter-operability of agents' systems essential or could they be cross-platform in data handling?

- How and in what format is the data communicated among the agents, and between the agents and the system? There have been several proposals for inter-agent communication in the past in the form of KQML (Knowledge Query Manipulation Language), and more recently in the

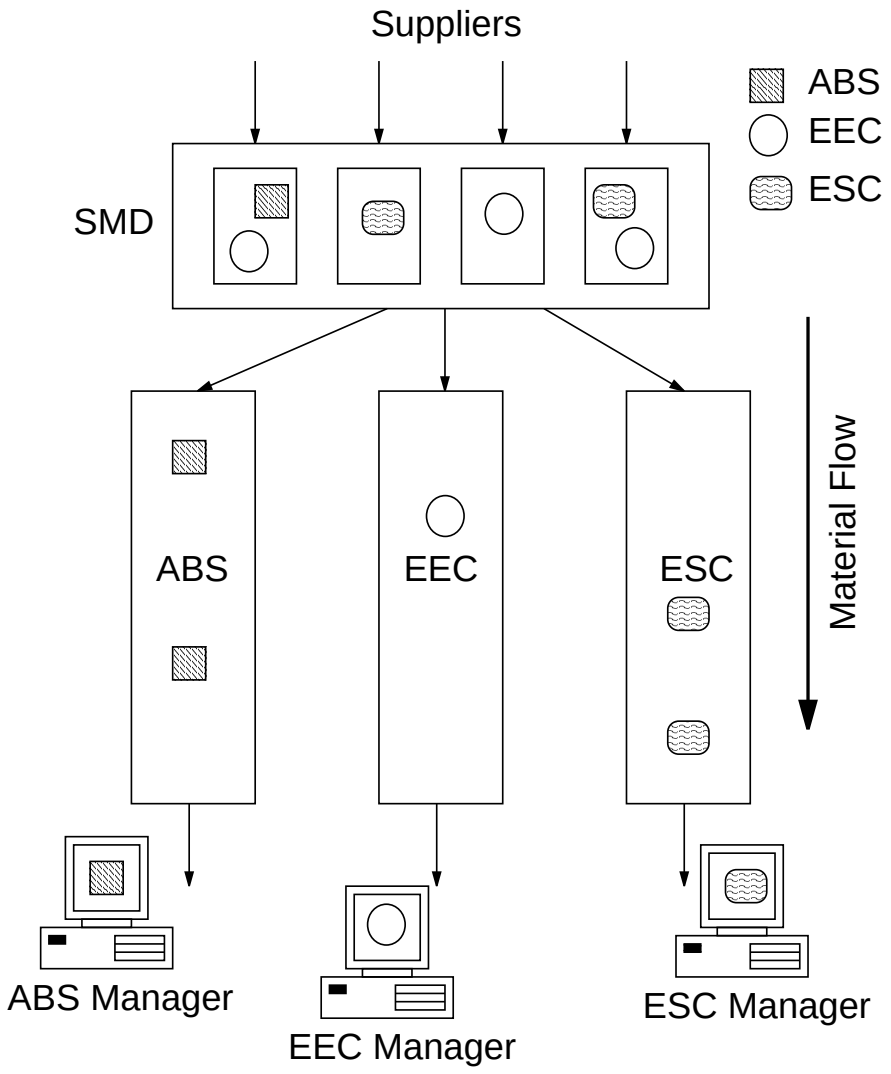

Figure 1: A schematic view of the electronics plant's distributed environment and shared SMD resources. Here we show only three product managers and their lines that share SMD resources. The three lines are Anti-lock braking system (ABS), Electronic engine control (EEC), and Electronic speed control (ESC).

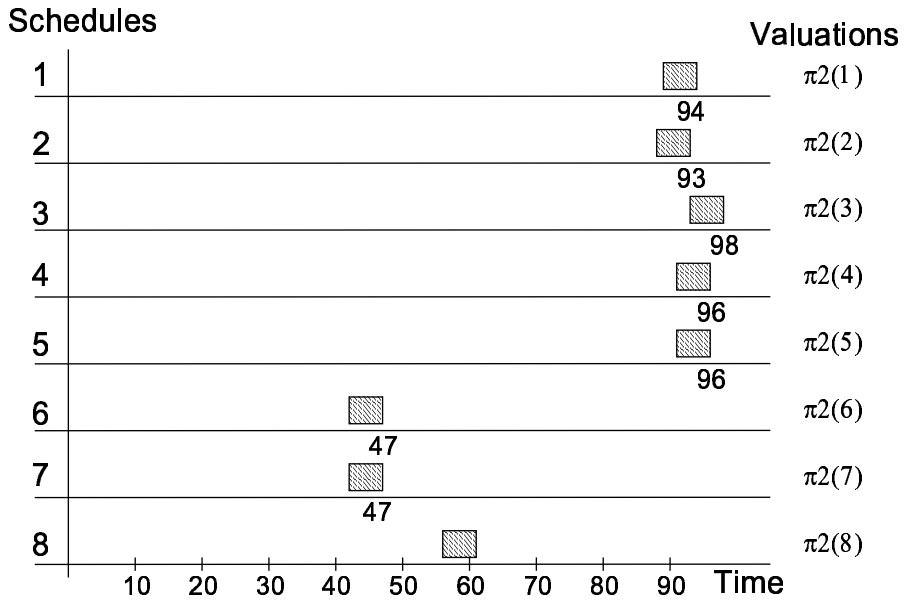

Figure 2: A schematic view of job 2's resource allocations extracted from 8 candidate schedules. Shaded box for each schedule shows the resource allocation for job 2, and the values on the horizontal axis denotes the completion time of the job if the corresponding schedule is used. Profit valuations are expected to be entered by agent 2 on a Web-based form.

form of XML. For example, Genesereth and Ketchpel builds a Agent Communication Languge (ACL)based on KQML specifications [7].

- Once the agent messages are communicated, in what format will the selected schedule be broadcast to the agents? Will there be an electronic mechanism to handle the transfers?

Of course we should note that these issues are inter-dependent, i.e., decisions made for one level affect the choices and then the decisions at the other levels. For example, the database choice affects directly what communication languages are available, and which one is chosen. We finally depict typical interdependencies and flow of data in Figure 3 using a 3-agent example.

3.3 A Numerical Example

We now construct a numerical example to illustrate some of the finer points of the schedule selection game and its associated web-based mechanism. The numbers used in the example are from real data. We make the following implementation assumptions:

- We assume that a primary concern of each job agent is to meet the job due-date.
- We assume that lot sizing and routing decisions are made before scheduling decisions. This assumption leads to two important simplifications: (1) setup times can be added to the processing times of the jobs, and (2) the problem can be decomposed into a series of single machine problems since the machine assignments are made through routing decisions. We use the lot-sizing and routing decisions made by the scheduling software that has been implemented in the plant during the test period. Note, however, any algorithm can be used instead. Hence, we can think of the scheduling process as an integral part of a larger planning/routing/scheduling decision structure where the higher-level decisions are an input for our scheduling process.
- At any one time there may be thousands of jobs in the system database. To limit ourselves to a reasonable number of jobs, we constructed a "candidate job set" which have due dates smaller than a certain threshold value. In this way, we eliminated jobs that would not be scheduled in the current planning horizon even if they were considered. In the following numerical example, we excluded the jobs (i.e., did not place them in the candidate job set) with due dates later than 10 days from current date.
- We assume hourly time buckets for our discrete time model of the scheduling problem. Due to this, we needed to consolidate really short jobs to form "super" jobs that have sizes comparable to our time bucket. We use the same planning horizon (5 days, 120 hours) that is employed in the plant's current practice.

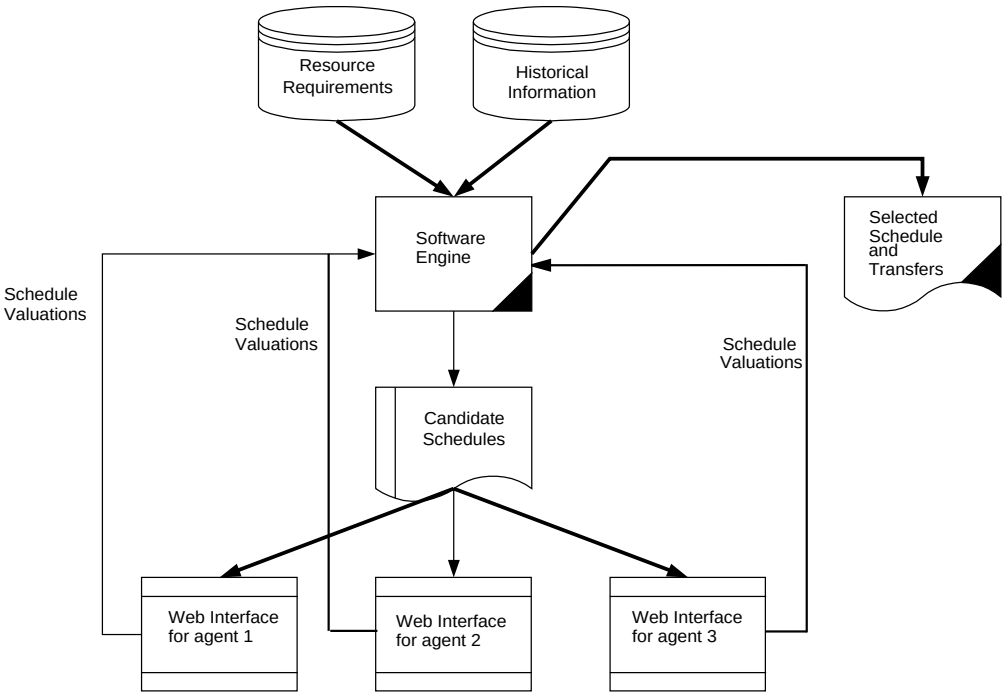

Figure 3: A potential architecture of a web implementation for the proposed Groves-Clarke mechanism for schedule selection game with three agents

Formally, each job agent i, and his job, can be represented by a 4-tuple (a_i, m_i, p_i, d_i), where a_i is the release time (or earliest start time) of job i, m_i is the SMD machine that the job is assigned, p_i is the processing time requirement on machine m_i including setup time, and d_i is the derived job due date. To construct the candidate schedules in set Y, we used the Lagrangean-based auction-theoretic algorithm developed in our earlier studies [15, 16]. Thus, we have a set of feasible schedules from which we form the schedule selection game.

We construct a "utility" evaluation table to be filled by the job agents which represents a depiction of potential web-based interface. Each job agent has access to his part of each schedule and fills a value for each schedule in a space reserved for it (see Figure 2. See Table 1 for an example of such a valuation table which could be easily implemented on a web-based system. In this example, we have 33 jobs to be scheduled on a single SMD machine. Note that similar tables can be constructed for other jobs and resources since they are independent in our current setting. In the more complex job shop scheduling problem, we would still have a single table for relating the jobs to each schedule. To have a manageable size example, we have created 8 schedules. Each job agent in this example fills the tardiness values for each of these 8 schedules. The table shows the (truly) reported tardiness values $(\mu_i^*(y))$ for each job under each schedule.

Using these reported values, the Groves-Clarke mechanism selects a schedule using the minimization version of the schedule selection function $y^*(\mu^*)$ (our goal is to minimize total disutility or total weighted tardiness). Thus, the selected schedule (y^*) is the last one (schedule 8) in the table with a total tardiness of 185. The mechanism next calculates the transfers from agents. Recall that, to calculate Groves-Clarke transfer for agent i, we need to find what the other agents would select without i's messages, $y^*_{-i}(\mu_{-i})$. The column titled y^*_{-i} lists the schedules that would be selected if the corresponding agent did not report any values. Note that if an agent does not affect the others' selection (i.e., $y^*_{-i} = y^* = 8$) then his transfer is zero, otherwise he pays the increase that he causes on others' total tardiness. Consider, for example, job number 6. Without job 6's reported values, the others would select schedule 1 with total tardiness of 172 $(= \min_{y \in Y} \sum_{i' \neq 6} \mu_{i'}(y^*_{-6}(\mu_{-6})))$. The difference between the tardiness of schedule 8 and that of 1 can be seen due to the involvement of job 6. Thus, the agent for job 6 pays the difference: $185 - 172 = 13$. Note that even after paying this amount, the agent's total cost (tardiness + payment) with the selection of schedule 8 is 13 $(= 0 + 13)$ which is still lower than what would be if the other agents collectively selected schedule 1 $(13 < 29)$ without his participation. We can show that this is true for every job agent. The agents with negative transfers (1, 4, 6, 8, 15, 19, 26, 27, 30, and 31) are charged 129 units in total, i.e., they pay this amount to collectively select schedule 8 instead of another alternative. The last column in the table lists the agents' total costs after transfer. Finally, note that, although each agent individually has a positive benefit with respect to what the others would choose, it may be worse off with respect to another schedule. For example, agent 6 pays 13 (with total cost of 13) but there are in fact schedules with his tardiness values of 8 or even 0 (assuming these are without transfers, recall individual rationality

Table 1: An example valuation table for the Groves-Clarke mechanism for 8 schedules of 33 jobs

	Schedules ($y \in Y$)										
Job (i)	1	2	3	4	5	6	7	**8**	y^*_{-i}	$-\delta^*_i$	$\pi_i - \delta^*_i$
1	36	4	4	4	4	0	0	0	1	20	20
2	7	6	11	9	9	0	0	0	8	0	0
3	4	11	0	0	0	21	0	4	8	0	4
4	0	14	0	0	0	0	15	0	7	9	9
5	0	0	0	0	0	0	0	22	8	0	22
6	29	0	0	8	8	0	0	0	1	13	13
7	8	19	24	10	10	0	0	0	8	0	0
8	0	16	0	0	65	0	17	0	7	11	11
9	16	0	5	12	12	0	0	0	1	0	0
10	0	2	7	14	14	0	0	0	8	0	0
11	0	25	30	14	18	0	0	0	8	0	0
12	0	27	32	16	15	0	0	0	8	0	0
13	0	6	11	16	16	0	0	0	8	0	0
14	0	6	11	16	16	0	0	23	8	0	23
15	0	14	0	0	0	21	15	0	7	9	9
16	0	7	12	17	17	0	0	24	8	0	24
17	26	21	1	21	0	63	68	67	8	0	67
18	0	32	37	17	16	0	0	0	8	0	0
19	29	37	8	8	8	0	0	0	1	13	13
20	0	36	38	18	17	0	0	0	8	0	0
21	11	39	41	13	13	0	0	0	8	0	0
22	0	17	19	20	20	20	0	0	8	0	0
23	0	0	0	0	0	0	5	5	8	0	5
24	0	18	20	21	21	21	0	24	8	0	24
25	0	0	0	0	0	0	4	4	8	0	4
26	0	18	0	0	67	0	19	0	7	13	13
27	0	20	0	0	69	0	21	0	7	15	15
28	14	48	10	16	16	0	0	0	8	0	0
29	0	21	1	21	70	0	0	0	8	0	0
30	1	25	25	26	26	26	0	0	6	5	5
31	8	29	29	30	30	30	27	0	7	21	21
32	0	52	52	19	18	0	0	0	8	0	0
33	12	0	61	11	54	4	0	12	8	0	12
Total	201	570	489	377	649	206	191	185		129	314

conditions). In fact, some agents (including 6) are worse off even when their total costs are compared to the mean tardiness for which a schedule is randomly (with equal probability across schedules) selected from set Y.

4 Discussion

We now discuss several interesting issues related to incentive compatible schedule selection in light of the illustrative example.

4.1 Generation of Schedules

A direct revelation mechanism for distributed scheduling involves many inter-related issues in order for it to be implemented successfully: First of all, we need to address the question of *how* the schedules collected in set Y are generated. Obviously schedules in this set affect the quality of the final schedule selected in a direct revelation mechanism. It is also important since it affects the computational implication of the overall mechanism. We already argued that a schedule generation engine on a host computer with a common database would be able to generate candidate schedules and post them using a web interface to each agent. Here we discuss the selection of the algorithm used in this engine so that we achieve certain final quality and control the computational burden.

We first assume that each schedule in Y represents a feasible resource (time periods) allocation from the shared resources to the job agents, i.e., in each schedule, non-overlapping contiguous blocks of time slots from each machine are assigned to the job agents for their respective operation on that machine. We may interpret the set of the candidate schedules as a collection of Gantt charts. Thus, we can identify a schedule y in set Y by using job - machine - time slot assignment indicators such as $X_{ikt}(y) = 1$ if job i is assigned to time slot t on machine k in schedule y (or equivalently a starting time and a completion time for each operation). In this representation, a job agent does not need to know the full schedule (related to the others) to compare the schedules; knowing his own assignment is enough to compute his net utility. A job agent can compute the value of a schedule (profit that it can make or tardiness costs) by using the part of the schedule that is related to his job. If we collect all the time slots assigned to agent i in $X_i(y) \equiv \{X_{ikt}(y)\}_{k=1,\ldots,M,t=1,\ldots,T}$, then the job agent can compute his profit by using X_i:

$$\pi_i(y) = \pi_i(X_i(y)) \tag{23}$$

We know that the number of schedules to be presented to the agents are limited by computational resources, such as memory and computer time. One possibility is to consider all feasible schedules that make up the original scheduling problem among the job agents. For example, if the problem is a classical job shop scheduling problem (JSP) then there are exponential number of schedules (all semi-active schedules for regular performance measures such as weighted tardiness) from which

we can choose our candidate set of schedules. An important issue is then which schedules for the original scheduling problem should be put in the set of "candidate schedules" Y. An interesting but somewhat trivial result regarding the quality of the final schedule collectively selected is that if the set of candidate schedules Y consists of an optimal schedule of the original scheduling problem, then this schedule will definitely be selected through the mechanism, i.e., if Y is guaranteed to have a certain performance guarantee on the total profit, then the schedule selected using the schedule selection function has the same performance guarantee. This is summarized in the following proposition.

Proposition 3 *The Groves-Clarke mechanism is guaranteed to choose a global optimal schedule of the original problem and achieve the optimal joint profit if the set Y consists of an optimal schedule, i.e., $\sum_i \pi_i(y^*) = v(JSP)$, where $v(JSP)$ denotes the value of a (global) optimal schedule.*

Then the remaining question is: How do we select a subset of good schedules to be placed in set Y? What if we are interested in including the optimal schedule in this set? We list several alternatives for the construction of the candidate schedule set as follows:

- *Historical information*: In most practical environments, schedule selection is not a one time process, but an exercise to be repeated time and again. In this case, an important source of information for the generation of quality schedules is the performance of those selected in the past. Schedules that have been implemented in the past, with their realized performance measures kept in a common database, may help identify promising schedules for the current selection process. Of course the usefulness of the historic information depends on the variability of problem instances over time. Suppose that every week a similar set of jobs (similar routing, processing, release and shipment requirements) compete for shared machines. It is likely that the past performance might reveal patterns in the schedules that can be used to generate candidate schedules.

- *Preprocessing*: We can view the schedule selection game as a stage in a distributed scheduling procedure preceded by a preprocessing or screening stage. In this view, a set of high-quality schedules are collected in set Y in the preprocessing stage. The two methods outlined in our previous works [15, 16] will be useful to isolate good quality candidate schedules. These are in fact what we used to generate the candidate schedules in the example above. The feasibility restoration routine applied to a Lagrangean schedule as outlined in [15] will produce a number of schedules which can be placed in set Y. In [16], we showed that one of a small number of precedence (or ranking) constraints must hold in an optimal solution. A preprocessing stage will involve fixing one out of these alternative precedence constraints among the jobs and generating different schedules, each with a different alternative ranking fixed. Since, we know that this preprocessing does not eliminate the optimal schedule, the schedule selection mechanism will be able to select the optimal schedule for the original problem. The only prohibitive element of

this type of preprocessing is the number of the candidate schedules allowed, that is, the size of Y. This may be controlled to a certain degree by effectively excluding some of the ranking from consideration although a fully optimal schedule cannot be guaranteed in this case.

- Another rather different alternative is to have the agents propose full feasible schedules that are supposed to improve a default schedule that will be implemented in case any better schedule is not proposed for a certain time period. One such implementation for train scheduling can be found in [2] where the agents are provided with the incentives, i.e., rebates from the global performance improvement, to propose high-quality schedules. Note that, in this case, although the generation of schedules are carried out along with the on-line selection of a schedule, the private information assumption is rather destroyed since each agent is assumed to have full information to propose a fully feasible schedule. We should also note that this will increase the web-based interaction (communication) between the agents and the scheduling engine that processes the proposed schedules.

We should point out that these methods are not mutually exclusive, i.e., they can be combined in a single process to make the candidate schedule set smaller and more effective in finding an optimal schedule.

4.2 Aligning Original Performance and Transfers

As we have outlined above, an issue is to align the "normal" performance measure (e.g., weighted tardiness) with the unit of transfers. This is a prerequisite for a sensible "utility" definition of the agents, i.e., after-transfer (net) performance, $u_i(y) = \pi_i(y) + \delta_i$. The agents should take into account the effects of transfers as well as the initial performance. In the mechanism design literature, both the initial performance and the transfers are interpreted and expressed in monetary terms, which is what we did in our discussion. Earlier studies overlooked this important issue. For example, Nisan and Ronen [18] ignored this alignment issue although machine agents announce processing times for their valuation functions and transfers end up being in time units in their mechanism.

For this problem, we think there is not a single universal solution since the original performance and transfers can be in any units. We at least observe that when the two are not in the same units, we need a normalizing function to convert the initial performance such that the overall utility makes sense for the agents. For example, one can use a function that calculates actual tardiness costs including any late charges or express shipments costs. Thus, the "profit" of agent i is defined as

$$\pi_i(y) = f(T_i(y)), \tag{24}$$

where T_i is tardiness of job i (which itself is a function of completion time) in schedule y and f is a conversion function. Note that the argument of the function can be generalized to include other

parameters (price, flow time, resource usage, etc.) that might be affected by a schedule. A simple conversion function might be as simple as a linear revenue function:

$$f(T_i(y)) = P_i - w_i T_i(y), \tag{25}$$

where P_i is the price of the job (i.e., the agent is paid P_i when the job is completed and delivered), and w_i is the penalty that the corresponding agent i has to pay for every time unit that his job is late (over the job's due date).

Furthermore, the transfers should be explicitly processed through actual transactions so that the transfers can affect overall utility of the agents. Considering that actual monetary transactions can be difficult to implement in a intra-plant competition and coordination, we may imitate the environment by distributing some type of "currency" (chips or tokens) to the agents and allow them to spend for better scheduling selections. In this case, each agent has a number of tokens that could be used to "pay" transfer amounts over a certain time period (say a month or a quarter) that spans more than one "scheduling game" instance. If the agents get a new set of tokens every scheduling instance, then there would be no effect of tokens on the valuation revelation. The limited number of tokens will force the agents strategize about how they allocate the tokens over multiple scheduling instances: If an agent who has spent most of his tokens in early games may not have enough to support his preferences later, which will have desirable effects on the fairness too. An issue that might be important in the implementation of this approach is the question of how the tokens are allocated to the agents in the first place. Knowing that some jobs and customers are more important than others, uniform (or equal) distribution of tokens may jeopardize the performance of certain job agents unnecessarily. For example, Pinedo [19] proposes to allocate the agents a budget that depends on the important characteristics of their jobs: total processing time requirement, weight, due date tightness, etc.

Another alternative is to align the incentives of the agents with their original performance using an "evaluation measure." In this way, the compensation to an agent will directly depend on the evaluation measure, which is the net utility of the agent after transfers. Note that in the latter interpretation, we assume the existence of a controller or a "center" that rewards or punishes the agents depending on their realized evaluation measures. This is the original view of Groves when he developed his incentive scheme for "optimal" organizational control. Here the major assumption is that the center has the capability to observe the agents' performances without any error so that he can calculate agents' compensation whether it is a reward or punishment. We give a more detailed discussion about this in the next subsection.

4.3 Coordinated Scheduling with a Central Authority

We have presented the mechanism in the previous sections as a collective decision making process without a center. However, it may be modified to accommodate a central authority. For example, if we assume that (1) the center is responsible for generating the schedules in set Y, (2) selects a

mechanism with a message space and an outcome function that chooses a schedule and arranges the transfers, and (3) collects money from the agents with negative transfers and pays to the agents with positive transfers, then there are only technical differences which are not critical for our analysis.

However, if we consider a central agency who makes decisions that may influence the agents' decisions and who has authority over the agents' actions, the issues differ significantly. The center here may allocate resources (e.g., time windows on machines assigned to job agents) that limit the decisions left to the local agents, and he/she may choose to reward or punish the agents depending on their actual performance (note that in the extreme case, the center's decisions may induce exactly one solution in the agent's feasible space). The main difficulty for an "authoritarian center" while making these decisions is that he/she may not have perfect information about local constraints and preferences so that sound decisions may not be possible, or alternatively, it may be essential to elicit private information from the agents in order to maximize the center's utility. Since the overall performance is the result of decisions by both the center and the agents, the nature of the coordination and the balance of the authority present a significantly different design problem.

We note the following differences as compared to the centerless design:

- The main issue here is to elicit private information from the local agents so that (1) the center makes the best allocation of shared resources, and then (2) the agents make best local decisions that are aligned with the global objectives.

- The center might have different incentive schemes that are compatible with the overall objective. One common tool is an evaluation measure for the agents that has been used in other Groves mechanisms (see, e.g., [13]). When the compensation to an agent is tied to the evaluation measure of the agent, the incentive compatibility is assured. As expected, the form of the evaluation measure function is similar to the net utility function defined in previous sections.

- For this mechanism to be implemented, we need to assume that (1) the center can enforce resource assignments but he has no control on the local decisions of the agents, and (2) the center can measure the final performance of the agents at the end of the execution of a schedule (realized values of profits) and reward or punish them according to these realized performances. Hence, the center is assumed to have a role of both a decision maker and a *controller*. Thus, the mechanism is not only a direct revelation mechanism from a game theoretic point of view, but also a *control mechanism* from an organization control view [13].

- Possible steps for such mechanism may be as follows: (1) The agents are asked to report *maximum profit* that they can achieve for possible time window assignments. (2) The center makes time window assignment decisions based on the reported values by the agents. (3) The agents make the local decisions under the center's time window restrictions for their respective jobs, and they use their assigned time windows to process their jobs. (4) At the end of schedule

execution, each agent's performance is measured, and rewards are calculated and distributed to the agents.

A more formal analysis of the mechanism design problem with a central agent is left for future study.

5 Conclusions

In this study, we present a simple and robust model that would enhance a distributed and possibly web-based decision making process for resource allocation and scheduling problems. Although, we primarily focus on intra-plant machine scheduling problem by considering agents who share resources, an extension to the inter-plant general resource allocation problem is possible. Collaborative planning, information sharing and distributed resource allocation across multiple firms are likely to be enhanced by the use of web-based decision support and collaboration tools. An analysis in this broader context may lead us to new and interesting results, especially in the context of collaborative supply chain management and coordination of multi-firm activities.

We address the issue of incentive compatibility in a "schedule selection game", which could be easily implemented in a web-based system with geographically dispersed players. We showed that despite the difficulties involved in balancing the budget and ensuring individual rationality, it is possible to devise a simple incentive compatible mechanism for distributed resource allocation and scheduling. We discuss additional challenges and open research questions posed by the scheduling game, such as alternative schedule generation, asynchronous scheduling, and the role of the central agent.

References

[1] K. J. Arrow. The property rights doctrine and demand revelation under incomplete information. In M. Boskin, editor, *Economics and Human Welfare*, pages 23–39. Academic Press, New York, NY, 1979.

[2] P. J. Brewer. Decentralized computation procurement and computational robustness in a smart market. Technical report, Department of Economics, Georgia State University, Atlanta, GA, 1997.

[3] P. J. Brewer and C. R. Plott. A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks. *International Journal of Industrial Organization*, 14:857–886, 1996.

[4] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[5] C. J. Corbett, F. J. C. Debets, and L. N. Van Wassenhove. Decentralization of responsbility for site decontamination projects: A budget allocation approach. *European Journal of Operational Research*, 86:103–119, 1995.

[6] d'Aspremont and L. A. Gerard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11:25–45, 1979.

[7] M. Genesereth and S. Ketchpel. Software agents. *Communications of ACM*, 37(7):48–53, 1994.

[8] J. Green and J. J. Laffont. Characterization of strongly individually incentive compatible mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[9] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[10] T. Groves. Incentive compatible control of decentralized organizations. In Y. C. Ho and S. K. Mitter, editors, *Directions in Large-Scale Systems: Many-person Optimization and Decentralized Control*, pages 149–185. Plenum Press, New York, NY, 1976.

[11] T. Groves. Information, incentives and the internalization of production externalities. In S. A. Lin, editor, *Theory and Measurement of Economic Externalities*, pages 65–83. Academic Press, New York, NY, 1976.

[12] T. Groves. Efficient collective choice when compensation is possible. *Review of Economic Studies*, 46:227–241, 1979.

[13] T. Groves and M. Loeb. Incentives in a divisionalized firm. *Management Science*, 25(3):221–231, 1979.

[14] V. Krishna and M. Perry. Efficient mechanism design. Technical report, Department of Economics, Pennsylvania State University, 1998.

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

[16] E. Kutanoglu and S. D. Wu. Proactive equilibrium for distributed job shop scheduling. Technical Report 99T, Department of Industrial and Manufacturing Systems Engineering, Lehigh University, 1999.

[17] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, New York, NY, 1995.

[18] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC99), to appear at Games and Economic Behaviour*, Atlanta, Georgia, 1999.

[19] M. Pinedo. *Scheduling: Theory, Algorithms and Systems*. Prentice Hall, Englewood Cliffs, NJ, 2 edition, 2001.

[20] T. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*, pages 11–68. MIT Press, 1987.

[21] J. Suijs. On incentive compatibility and budget balancedness in public decision making. *Economic Design*, 2:193–209, 1996.

[22] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

A Proof of Proposition 2

The proof is a modified version of the argument used in [12]. To prove the proposition, it is sufficient to show that the agents reveal their true profits in a dominant strategy equilibrium, i.e., for every $i \in \mathcal{N}$, $\mu_i^*(y) = \pi_i(y)\ \forall\ y \in Y$, where μ^* are dominant strategy equilibrium messages. We first denote the schedule that would be selected if $\mu_i(y) = \pi_i(y)\ \forall\ y \in Y$ as $y^*(\mu_i = \pi_i, \mu_{-i})$ when the others' joint message is shown by μ_{-i}. From definition of schedule selection function, we recall that $y^*(\mu_i = \pi_i, \mu_{-i})$ maximizes $\pi_i(y) + \sum_{i' \neq i} \mu_{i'}(y)$ over all schedules $y \in Y$. Then,

$$\pi_i(y^*(\mu_i = \pi_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i = \pi_i, \mu_{-i})) \geq \pi_i(y) + \sum_{i' \neq i} \mu_{i'}(y)\ \forall\ y \in Y. \tag{26}$$

The inequality (26) holds for any y in Y including $y^*(\mu_i, \mu_{-i}) \in Y$ for any $\mu_i \in M_i$. Thus,

$$\begin{aligned} \pi_i(y^*(\mu_i = \pi_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i = \pi_i, \mu_{-i})) \geq \\ \pi_i(y^*(\mu_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i, \mu_{-i}))\ \forall\ \mu_i \in M_i. \end{aligned} \tag{27}$$

To make the discussion simpler let us define

$$R_i(\mu_{-i}) = \sum_{i' \neq i} \mu_{i'}(y^*_{-i}(\mu_{-i})).$$

Note that the left hand side of inequality (27) is the utility of i from $y^*(\mu_i = \pi_i, \mu_{-i})$ plus the term $R_i(\mu_{-i})$:

$$\pi_i(y^*(\mu_i = \pi_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i = \pi_i, \mu_{-i})) = u_i(y^*(\mu_i = \pi_i, \mu_{-i}), \delta^*) + R_i(\mu_{-i})$$

Similarly, for the right hand side of the inequality,

$$\pi_i(y^*(\mu_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i, \mu_{-i})) = u_i(y^*(\mu_i, \mu_{-i}), \delta^*) + R_i(\mu_{-i})$$

Then,

$$u_i(y^*(\mu_i = \pi_i, \mu_{-i}), \delta^*) + R_i(\mu_{-i}) \geq u_i(y^*(\mu_i, \mu_{-i}), \delta^*) + R_i(\mu_{-i})$$

or, since R_i is independent of μ_i,

$$u_i(y^*(\mu_i = \pi_i, \mu_{-i}), \delta^*) \geq u_i(y^*(\mu_i, \mu_{-i}), \delta^*)\ \forall\ \mu_i \in M_i.$$

This holds for every i and for every μ_{-i} which proves the result. $\diamondsuit$