Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Mechanism Design for Scheduling with Uncertain Execution Time*

Vincent Conitzer
Department of Computer Science
Duke University
Durham, NC 27708, USA
conitzer@cs.duke.edu

Angelina Vidali
Department of Computer Science
Duke University
Durham, NC 27708, USA
vidali@cs.duke.edu

Abstract

We study the problem where a task (or multiple unrelated tasks) must be executed, there are multiple machines/agents that can potentially perform the task, and our objective is to minimize the expected sum of the agents' processing times. Each agent does not know exactly how long it will take him to finish the task; he only knows the distribution from which this time is drawn. These times are independent across agents and the distributions fulfill the monotone hazard rate condition. Agents are selfish and will lie about their distributions if this increases their expected utility.

We study different variations of the Vickrey mechanism that take as input the agents' reported distributions and the players' realized running times and that output a schedule that minimizes the expected sum of processing times, as well as payments that make it an ex-post equilibrium for the agents to both truthfully report their distributions and exert full effort to complete the task. We devise the ChPE mechanism, which is uniquely tailored to our problem, and has many desirable properties including: not rewarding agents that fail to finish the task and having non-negative payments.

Introduction

Task (re)allocation is a crucial component of many multiagent systems. When these systems consist of multiple self-interested agents, it is important to carefully consider the incentives that agents are given to take on tasks. The wrong incentives may, for example, lead an agent to report that it will complete the task much faster than it actually can. As a result, the task may not be allocated to the agent that can complete the task most efficiently in actuality, resulting in a loss of social welfare. Mechanism design provides the natural framework for considering such incentive questions. Another key issue is uncertainty: a cloud provider that is considering running a program provided by an unknown user in general cannot perfectly predict how long the execution will take. However, the cloud provider can reasonably have a probability distribution over the length of execution.

*We thank NSF and ARO for support under grants CCF-1101659, IIS-0953756, CCF-1337215, W911NF-12-1-0550, and W911NF-11-1-0332.

In this work, we construct mechanisms for scheduling a single[1] job, in the setting where each of the n selfish machines only knows the distribution from which its own running time is drawn. The mechanism then outputs a schedule that determines which machine gets to compute when (allowing preemption). It also outputs payments, which may also be based on which machine completed the job and when. We focus on efficient mechanisms, i.e., mechanisms that schedule the machines to minimize the total expected processing time, and require them to be truthful. We propose multiple such mechanisms, but our favorite is the ChPE mechanism, which has the most desirable properties. It is a type of Groves mechanism, but one in which an agent can affect the h term of its own payment (while in the classical Groves mechanism (see Section 9.3.3 of (Nisan et al. 2007) for definition) the h part can only depend on the values of the other players). This would seem problematic, except with a neat technical trick we are able to prove that the agent cannot affect the *expectation* of this h term.

Related work. (Nisan and Ronen 2001) proposed, among other paradigmatic algorithmic mechanism design problems, a mechanism design version of the problem of scheduling unrelated machines. In our work the payments are sometimes based in part on performance, rather than only on the machines' reports. This relates to the literature on mechanism design with partial verification, where the center can detect some, but not all of the lies. This was first formalized by Green and Laffont (Green and Laffont 1986), and subsequently studied in a number of papers. (Nisan and Ronen 2001) as well as several follow-up papers (e.g., (Auletta et al. 2006; Krysta and Ventre 2010; Fotakis and Zampetakis 2013; Caragiannis et al. 2012)) constructed mechanisms with verification for scheduling and auction settings. Our setting can be thought of as having "noisy" verification, where we have only probabilistic evidence that an agent was lying, because its performance is unlikely (but not impossible) given the distribution it reported.

The general phenomenon of execution uncertainty in mechanism design, a key concept in our work, has already been addressed in a number of other papers (Porter et al.

[1] It will be easy to see that if we have multiple *unrelated* tasks, our results extend naturally by applying the mechanisms we propose to each one of the tasks, one after the other.

2008; Johnson 2013; Ramchurn et al. 2009; Stein et al. 2011; Feige and Tennenholtz 2011). For example, in (Porter et al. 2008), each machine has a probability of failure and has a cost for attempting the task. In fact, in all these papers, each machine has a fixed cost for attempting the task, whereas in our case, the cost depends on how many time steps we allow the machine to run (we assume the cost is 1 per unit of time). (Porter et al. 2008; Johnson 2013) also deal with the case where the machines are correlated. In contrast to our work (Porter et al. 2008; Johnson 2013; Ramchurn et al. 2009) only allow the task to be allocated to a single machine The closest paper to our work is (Stein et al. 2011), which does allow the task to be allocated to multiple machines. However, in their work, once a machine is invoked it cannot be stopped or preempted, and the only reason to invoke multiple machines is to meet a deadline. In contrast, we allow the schedule to re-assign the task to different machines at different time steps, simply to minimize expected total processing time. Although Vickrey-type mechanisms with verification are also proposed in these papers, they do not obtain all the desirable properties of the ChPE mechanism that we propose for our setting.

From an economic perspective, in our setting, an interesting type of *interdependence* in the agents' utility functions emerges. That is, a machine's expected utility depends not only on its true type (=distribution) and all agents' reported types, but additionally on the *true* types of the other agents. This is why we only obtain *ex-post* equilibrium rather than dominant strategies. Interdependence arises in a very similar manner as in our setting in (Stein et al. 2011), while in (Ramchurn et al. 2009) interdependence exists due to a different reason, namely agents reporting on each other's likelihood to succeed.

Model and definitions There is a single job that needs to be processed and a set $N = \{1, \ldots, n\}$ of machines/players. The objective is to minimize the expected processing time. A machine does not know the exact time that it would need to finish the task, but it does know a distribution over this time; this distribution will be its private information in the mechanism design problem. Note that this is not the distribution *over* its type; the distribution *is* its type.

If machine i spends a unit of time on the task, it will either finish, or learn that it has not finished. Formally, let the discrete random variable T_i denote the *running time* of machine i, which is the time it would need if it continued to run on the job until its completion. We will use $t_i \in \{1, 2, \ldots\} \cup \{\infty\}$ to denote a realization of T_i. Let f_i denote the probability density function of T_i, i.e., $P(T_i = t_i) = f_i(t_i)$. Furthermore let F_i denote the cumulative density function of T_i, i.e., $P(T_i < t_i) = F_i(t_i)$. Let $\zeta_i(t_i) = f_i(t_i)/(1 - F_i(t_i))$ denote player i's *hazard rate*, i.e., the probability, conditional on not having finished strictly before time t_i, of finishing at time t_i. (The distributions of two different players i and j are independent from each other.) The hazard rate function is very often used in risk management and reliability theory because it has been found useful and in clarifying the relationship between physical modes of failure.

We say player i's distribution has a *monotone hazard rate (MHR)* if $\zeta_i(t_i)$ is non-increasing as a function of t_i. That is, the longer the machine has already run on the job, the less likely it is that it will finish immediately. We assume MHR throughout the paper. This reflects that, if player i has multiple approaches that it can take towards solving the problem—for example, it can attempt one heuristic in each time unit—it is optimal for the player to try those that are most likely to succeed first. Decreasing hazard rate distributions include the hyperexponential, Pareto, and some cases of Weibull.

The objective is to minimize the expected processing cost. We assume that the cost of processing per unit of time is 1 for all machines. The machines are selfish so they will not do any processing if they are not given an incentive (payment) to do so. A selfish player wants to maximize his expected utility, which is payment minus processing cost. Under this model, even though we allow the task to run on any number of machines at each time step, it is always suboptimal to schedule two machines to run in parallel during any time unit: one would be better off sequentializing the two units of processing time, just in case the job gets completed during the first time unit, so the second is no longer necessary. Hence, only one machine will run in each time unit.

We do allow preemption: e.g., the scheduler can let machine 1 run for two time units first, then let machine 2 run for three time units, then machine 1 for one time unit, etc. We emphasize that, with this schedule, the probability that we finish in the sixth time unit given that we have not finished earlier is $\zeta_1(3)$, not $\zeta_1(6)$, because this is only the third time unit in which machine 1 is actively processing. We will generally use t_i to denote time on i's "clock", that is, i's local time is $t_1 = 3$ when the global time is $t = 6$ in the above example. Once any machine finishes, all computation stops.

In a (direct-revelation) mechanism, each player i reports $\hat{f}_i$ (or, equivalently, $\hat{\zeta}_i$) to the mechanism. The mechanism, based on $(\hat{f}_1, \ldots, \hat{f}_n)$, produces as output a schedule $s : \{1, 2, \ldots\} \to N$, where $s(t)$ is the machine that is assigned to process at (global) time t. $\mathbf{s}$ denotes the infinite vector $\mathbf{s} := (s(1), s(2), \ldots)$. Note that $\mathbf{s}$ does not contain the information of when the computation finishes. Ω is the random variable that denotes the (global) time at which the computation finishes (and ω for a realization of it).

Nothing we have said so far precludes the possibility that there is a positive probability of never finishing, resulting in infinite expected cost. For the sake of simplicity, in most of the paper, we simply assume this away by assuming that all hazard rates are bounded below by a positive constant. In the full version of our paper we also abandon this assumption and consider the possibility that it may be better to give up on the task at some point, modeling that completion of the task has a value of Φ.

Our Results

The optimal schedule with MHR players

The MHR assumption makes it easy to find the optimal schedule given the machines' true hazard rates: always greedily use the machine that currently has the highest hazard rate. In this subsection, which has no game-theoretic

considerations, we prove this and discuss. Let $t_i(t, \mathbf{s})$ be the number of occurrences of i in $\mathbf{s}$ before and at t. That is, it denotes i's local time at global time t under schedule $\mathbf{s}$. Then, let $\zeta^{\mathbf{s}}(t) = \zeta_{s(t)}(t_{\mathbf{s}(t)}(t, \mathbf{s})))$ denote the hazard rate at time t under schedule $\mathbf{s}$.

Lemma 1. *The probability that none of the machines has completed the task before time t is given by the formula: $1 - F^{\mathbf{s}}(t) = \prod_{t'=1}^{t-1}(1 - \zeta^{\mathbf{s}}(t'))$.*

Proof. Let $T^{\mathbf{s}}$ be a random variable that corresponds to the time at which we finish the task under schedule $\mathbf{s}$. From $P(T^{\mathbf{s}} < t' + 1) = P(T^{\mathbf{s}} < t') + P(T^{\mathbf{s}} = t')$ we get $F^{\mathbf{s}}(t'+1) = F^{\mathbf{s}}(t') + f^{\mathbf{s}}(t')$. Consequently, $1 - F^{\mathbf{s}}(t'+1) = 1 - F^{\mathbf{s}}(t') - (1 - F^{\mathbf{s}}(t'))\frac{f^{\mathbf{s}}(t')}{1-F^{\mathbf{s}}(t')} = 1 - F^{\mathbf{s}}(t') - (1 - F^{\mathbf{s}}(t'))\zeta^{\mathbf{s}}(t') = (1 - F^{\mathbf{s}}(t'))(1 - \zeta^{\mathbf{s}}(t'))$. From this recurrence we easily obtain the desired equality $1 - F^{\mathbf{s}}(t) = \prod_{t'=1}^{t-1}(1 - \zeta^{\mathbf{s}}(t'))$. □

A *greedy schedule* $\mathbf{g}$ at each time step t assigns the machine that maximizes $\zeta_{s(t)}(t_i(t, \mathbf{s})))$ by being so assigned. Under the MHR assumption, this corresponds to obtaining the schedule simply by sorting all the machines' individual hazard rates. We now prove it is optimal assuming MHR.

Proposition 1. *Under the MHR assumption, for every t, $\mathbf{g} \in \arg\max_{\mathbf{s}} F^{\mathbf{s}}(t)$ (and,* a fortiori, *$\mathbf{g} \in \arg\min_{\mathbf{s}} E[T^{\mathbf{s}}]$).*

The proof is deferred to the full version of our paper.

Remark 1. *Note that the greedy algorithm is not optimal without the MHR assumption. Consider an example with two machines whose hazard rates are given according to the table below. The greedy schedule assigns the job to machine 2 forever, resulting in an expected completion time of 5. On the other hand, assigning the job to machine 1 forever results in an expected completion time of only 2.*

	$\zeta_i(1)$	$\zeta_i(2)$	$\zeta_i(t)(t \geq 3)$
machine i=1	0.1	0.9	0.9
machine i=2	0.2	0.2	0.2

The following property will help us with the analysis of the ChpE mechanism and relies on MHR.

Remark 2 (Consistency Property). *If $\mathbf{s}$ is optimal for agents N, then we can obtain an optimal schedule for agents $N \setminus \{i\}$ simply by (1) removing i from the schedule, and (2) if this results in a finite schedule (because the infinite tail end of the schedule was assigned to i), sorting the previously unused hazard rates of $N \setminus \{i\}$ and appending them to the end of the finite schedule, as in the following example.*

Example 1. *For the following instance:*

	$\zeta_i(1)$	$\zeta_i(2)$	$\zeta_i(t)(t \geq 3)$
machine $i = 1$	0.9	0.7	0.6
machine $i = 2$	0.8	0.5	0.3
machine $i = 3$	0.4	0.2	0.1

The ordered hazard rates are $(0.9, \mathbf{0.8}, 0.7, 0.6, 0.6, \ldots)$ and the corresponding optimal schedule is $(1, \mathbf{2}, 1, 1, 1, \ldots)$. If we then take out player 1, the ordered hazard rates become $(\mathbf{0.8}, 0.5, 0.4, 0.3, 0.3 \ldots)$ and the corresponding optimal schedule is $(\mathbf{2}, 2, 3, 2, 2, \ldots)$.

In examples like this, it can be helpful to, in a slight abuse of notation, insert all of the hazard rates into the schedule from the outset: $(0.9, 0.8, 0.7, \overline{0.6}, 0.5, 0.4, \overline{0.3}, 0.2, \overline{0.1})$ and $\mathbf{s} = (1, 2, \overline{1}, 2, 3, \overline{2}, \overline{3})$, where $\overline{x}$ denotes an infinite number of occurrences of x. The next machine after $\overline{i}$ will take over the computation only in the case that machine i is absent.

Additional notation

Now that we have a better understanding of optimal scheduling in our context, we introduce some further notation. Let random variable R_i denote the time that player i actually gets to run (and r_i to denote a value to which it realizes). We always have $R_i \leq T_i$; it is strictly smaller if another player finishes before player i would have finished. Thus, it is a function of the time at which the task is completed as well as the schedule; we will use $r_i(\omega, s)$ to denote this where it is not clear from context.

For any subset of the players $S \subseteq N$, let T_S denote the time at which we would have finished if we had only had the players in S available. We also define R_S as the sum of the realized times of a group of players S until the task gets completed, if it can only be assigned to players in S. Note that $R_S = T_S$. Let $\zeta_S(t)$ denote the corresponding sorted hazard rates, which define the distribution f_S over T_S. We will refer to this distribution as the *group distribution of S*.

Solution Concept and Revelation Principle

In this section, we review the solution concept we need and prove the corresponding revelation principle for our setting. We note that the players' strategies allow them not only to lie about their types, but also to decide at each time step if they want to process or not, and whether they wish to announce having found a solution.

In an *ex-post equilibrium*, every agent plays a strategy that is optimal for any type it may have and for any types that the other agents may have, as long as the other agents tell the truth. This solution concept is stronger than Bayes-Nash equilibrium but weaker than dominant strategies.

Definition 1. *We say that a direct-revelation scheduling mechanism m is (ex-post)* truthful *if it is an ex-post equilibrium for the machines to truthfully report $\hat{f}_i = f_i$, always process when asked, and always announce a solution immediately when it is found.*

Theorem 1 (Revelation Principle). *Suppose that the scheduling game defined by payment rule r has an ex-post equilibrium given by behavioral strategies $\sigma_1, \ldots, \sigma_n$. Then there exists an equivalent ex-post truthful direct-revelation scheduling mechanism m.*

We defer the technical definition of scheduling games and the proof of Theorem 1 to the full version.

Failure of Straightforward Adaptations of VCG

It is natural to try to adapt VCG mechanisms (Vickrey 1961; Groves 1973; Clarke 1971). In this section we present the most obvious way to adapt such mechanisms and show that the resulting mechanisms fail to be truthful. The payments do *not* depend on the execution (realizations of the processing times), as we will see later, this is their weakness.

Expected Pure Groves (EPG): The basic idea behind Groves mechanisms is to add to player i's payment a term that corresponds to the (reported) welfare of the other players, aligning the player's utility with the social welfare. (A term h, that does not depend on player i's report, can be added; we will refer to the mechanism where this h term is set to 0 as "Pure Groves.") In the context we are considering here, this would mean that the payment that machine i receives is $-E_{T_1\sim\hat{f}_1,\ldots,T_n\sim\hat{f}_n}[R_N - R_i]$ (Note that this is negative.) That is, machine i pays the expected social cost, omitting the term that corresponds to its own cost. The idea is to align the machine's utility with the social welfare.

It is natural to add an h term to obtain:

Expected Clarke (EC): In the Clarke mechanism, the h_i term is set to reflect the welfare that the other players would have had experienced without i, so that i's total payment is the difference i causes in the welfare of the other players. In the context considered here, this would play out as follows. Player i receives the EPG term $-E_{T_1\sim\hat{f}_1,\ldots,T_n\sim\hat{f}_n}[T_N - T_i]$, plus the following h term: $h_i(\hat{f}_{-i}) = E_{T_{-i}\sim\hat{f}_{-i}}[T_{N\setminus\{i\}}]$.

Its total payment is now nonnegative. Note that i cannot do anything to affect its h_i term.

Misreporting and miscomputing examples

We now show that under EPG and EC, a machine may have an incentive to misreport its distribution. Moreover, they generally have an incentive to "miscompute", that is, not actually process on the task when they are supposed to.

Proposition 2. *Under both EPG and EC, a machine can sometimes benefit from misreporting its distribution (even when the others report and compute truthfully).*

Proof. We consider the case where the true types of the players are given by the following table:

	$\zeta_i(1)$	$\zeta_i(t)(t \geq 2)$
machine $i = 1$	0.4	0.1
machine $i = 2$	0.3	0.1
machine $i = 3$	0.2	0.2

The optimal schedule is $(1, 2, 3, 3, \ldots)$.

Now suppose that machine 1 decides to overreport its probability of finishing in the first time slot, which does not change the allocation. The intuition is that by lying and overreporting, player 1 makes the expected sum of processing times of the other players seem smaller.

	$\hat{\zeta}_i(1)$	$\hat{\zeta}_i(t)(t \geq 2)$
machine i=1	0.5	0.1
machine i=2	0.3	0.1
machine i=3	0.2	0.2

The expected utility of machine 1 then becomes equal to $-1-[(1-0.5)\cdot 0.3\cdot 1+(1-0.5)(1-0.3)0.2\cdot 2+\ldots]$, where the part in brackets is the payment, which is larger than the same expression where 0.5 is replaced by 0.4. This shows that player 1 has an incentive to misreport. The exact same example works to show the same for EC. (There will be an additional h term in 1's utility, but this term will be the same with or without misreporting.) □

In fact, these mechanisms suffer from an even more basic problem: because the payments do not depend on performance, there is no incentive to actually process when asked.

Proposition 3. *Under both EPG and EC, a machine can benefit from failing to process when it is supposed to (even when the others report and compute truthfully).*

Proof. Consider again the example from the previous proof. If player 1 never computes, the first (-1) term in his utility will disappear, and his payment will be unaffected. □

Good mechanisms

As we have seen, some adaptations of VCG mechanisms have poor incentive properties in our context. However, as we will now show, VCG mechanisms can be adapted in other ways that do have desirable incentive properties. We introduce three specific mechanisms. All these mechanisms, except for ChPE, can be defined without assuming MHR.

Realized Pure Groves (RPG) The payment that player i receives is $-(r_N - r_i)$ (a negative amount). That is, once the job has been completed, machine i pays the realized social cost, omitting the term that corresponds to its own cost. The idea is to align the machine's utility with the social welfare.

As we will see later, this mechanism is in fact truthful. Unfortunately, it has other undesirable properties: machines always have to *make* payments, in addition to performing work, so their utility is always negative. In standard mechanism design contexts, this can be addressed by adding an h term to the payment function which depends only on the others' reports, to obtain the Clarke mechanism or one of its cousins. In our context, however, it is not immediately clear whether this h term should depend only on the reports, or also on the realized processing times. Our next mechanism does the former.

Clarke, h in Expectation (ChE): Player i receives the RPG term $-(r_N - r_i)$, plus the following h term: $h_i(\hat{f}_{-i}) = E_{T_{-i}\sim\hat{f}_{-i}}[T_{N\setminus\{i\}}]$ That is, the h term is the expected processing time that would result without i, taking the others' reports at face value.

Since there is nothing that machine i can do to affect its h term, the incentives are the same as for RPG. This is a familiar argument in mechanism design. What is perhaps more surprising is that in our next mechanism, the h term does depend on the realization, and player i will be able to affect it.

Clarke, h Partially in Expectation (ChPE): A natural thought at this point would be that we should use the realized value of h instead of its expectation. However, some reflection reveals that this does not make sense: at the point where the task is finished, we generally will not be able to determine how much longer the other machines would have taken. By forcing them to continue to run on the task in order to determine this, we would jettison efficiency. However, we do know *something* about how long the other agents would have taken. If agent i was the one to finish, and the other agents had incurred $r_{N\setminus\{i\}}$ at that point, then we know that the other agents would have taken longer

than $r_{N\setminus\{i\}}$ without agent i. This follows from the consistency property: all of the computations that the other agents did, they also would have done in the world without i. What we do not know is how much more time they would have needed from that point on. The idea behind ChPE is to take the expectation only over this part. Specifically, if machine i is the one to complete the job, it receives the RPG term $-(r_N - r_i)$, plus the h term: $h_i(r_{N\setminus\{i\}}, \hat{f}_{-i}) = E_{T_{-i}\sim\hat{f}_{-i}}[T_{N\setminus\{i\}}|T_{N\setminus\{i\}} > r_{N\setminus\{i\}}] = r_{N\setminus\{i\}} + E_{T_{-i}\sim\hat{f}_{-i}}[T_{N\setminus\{i\}} - r_{N\setminus\{i\}}|T_{N\setminus\{i\}} > r_{N\setminus\{i\}}]$. On the other hand, if machine i is not the one to complete the job, its payment is 0, because in this case, the h term is $h_i(r_{N\setminus\{i\}}, \hat{f}_{-i}) = E_{T_{-i}\sim\hat{f}_{-i}}[T_{N\setminus\{i\}}|T_{N\setminus\{i\}} = r_{N\setminus\{i\}}] = r_{N\setminus\{i\}} = (r_N - r_i)$, which cancels out the RPG term.

No Dominant-Strategies Truthfulness

In this section, we show that our mechanisms do not attain dominant-strategies truthfulness. Intuitively, this is because if a player knows that another player has (for whatever reason) misreported, the former can have an incentive to misreport as well to "correct" for the latter's misreport.

Proposition 4. *None of RPG, ChE, and ChPE is truthful in dominant strategies.*

Proof. To show that a mechanism is not truthful in dominant strategies, it suffices to find an instance where the following holds. Given that one player decides to deviate and lie, the best response for another player is to deviate as well.

We assume that the true distributions of the machines are the same as described in the proof of Proposition 2. Suppose that machine 2 (for whatever reason) lies and reports a distribution with hazard rate $\zeta_2(1) = 0.5$. In doing so, it gets to compute before machine 1, but otherwise the schedule is unchanged. If machine 1 is aware of this, it has an incentive to also lie and report $\zeta_1(1) = 0.6$, so that the mechanism again produces the allocation that is optimal with respect to the *true* distributions. This is precisely because the mechanisms in the statement of the proposition align the machine's utility with the social welfare.

	$\hat{\zeta}_i(1)$	$\hat{\zeta}_i(t)(t \geq 2)$
machine i=1	0.4	0.1
machine i=2	0.5	0.1
machine i=3	0.2	0.2

$\rightarrow$

	$\hat{\zeta}_i(1)$	$\hat{\zeta}_i(t)(t \geq 2)$
machine i=1	0.6	0.1
machine i=2	0.5	0.1
machine i=3	0.2	0.2

Under RPG, the situation on the left-hand side results in an expected utility for machine 1 of $-(1-0.3)\cdot 1 - [0.3\cdot 1 + (1-0.3)0.4\cdot 1 + (1-0.3)(1-0.4)0.2\cdot 2 + \ldots]$, where again the first term corresponds to 1's processing cost and the second to its payment. Note that, because this is RPG rather than EPG, this expression uses the actual probabilities used in the proof of Proposition 2 rather than any misreported ones; all a machine can affect by changing its report is the schedule. On the other hand, the situation on the right-hand side results in an expected utility for machine 1 of $-1 - [(1-0.4)\cdot 0.3\cdot 1 + (1-0.4)(1-0.3)0.2\cdot 2 + \ldots]$. This is exactly as if everyone had reported truthfully. Because $-(1-0.3)\cdot 1 - [0.3\cdot 1 + (1-0.3)0.4\cdot 1] = -0.7 - 0.3 - 0.28 = -1.28$ and $-1 - [(1-0.4)\cdot 0.3\cdot 1] = -1.18$, and the remaining terms are the same in the expressions, machine 1 prefers the second scenario, and thus has an incentive to misreport under RPG. The same example works for ChE because it adds the same h term in both cases; it also works for ChPE because the *expected* value of h will be the same in both cases (by Lemma 2). □

Ex-Post Truthfulness

We now show that RPG, ChE, and ChPE are ex-post truthful, that is, it is an ex-post equilibrium for each machine to report truthfully, to process when it is supposed to, and to announce a solution as soon as it is found. We first need the following lemma in order to deal with the case of ChPE.

Lemma 2. *In the ChPE mechanism, assuming truthful behavior by the other machines, machine i cannot affect the* expected *value of its h term, $E[h_i(r_{N\setminus\{i\}}, \hat{f}_{-i})]$, whether by lying, failing to compute, or withholding the solution.*

Proof. Recall that in ChPE, $h_i(r_{N\setminus\{i\}}, \hat{f}_{-i}) = r_{N\setminus\{i\}} + E_{T_{-i}\sim\hat{f}_{-i}}[T_{N\setminus\{i\}} - r_{N\setminus\{i\}}|T_{N\setminus\{i\}} > r_{N\setminus\{i\}}]$ if i finishes, and $h_i(r_{N\setminus\{i\}}, \hat{f}_{-i}) = r_{N\setminus\{i\}}$ otherwise. (We can ignore the RPG payment term throughout this proof.) Machine i in fact can do things to affect the *realization* of its h_i payment. For example, if it has finished the task, it can announce it at that point and get the corresponding h_i payment; alternatively, it can choose to never announce it and wait for another machine to finish, in which case it might get a lower h_i payment if another machine finishes the task shortly after, or a higher h_i payment if it takes the other machines much longer. That is, it can interrupt the other agents' processing and receive the expected value of their remaining processing time, or it can let them continue to process and receive the realization of their remaining processing time.

It is straightforward to check that in fact *any* manipulation that i has available can only affect the h_i payment by changing the distribution over the time at which i would interrupt the others' processing by announcing it has finished the task (thereby receiving the expectation of the others' remaining processing time rather than the realization). Let us grant our manipulator i even more power, allowing it to choose the precise point t at which to interrupt the others' processing. (t here is according to the other machines' combined "clock", i.e., it measures how much time the other machines have processed, but not i itself..) Let π_i^t denote the h_i payment i receives given that it interrupts at t. We obtain:

$$\pi_i^t(T_{-i}) = \begin{cases} T_{-i} & \text{if } T_{-i} \leq t \\ t + E[T_{-i} - t | T_{-i} > t] & \text{otherwise} \end{cases}.$$

The first case materializes if another player finishes the computation before t, and the second otherwise. (Note that this notation is implicitly using the fact that the others report truthfully—otherwise the expectation in the second case should be taken according to the reported distribution.) Now, we note that for any t, we have $E[\pi_i^t(T_{-i})] = P(T_{-i} \leq$

	Efficient	DS truthful	Ex-post truthful	No incentive to miscompute	ZPUF	IR	NNP	NNEP
EPG	✔	×	×	×	×	×	×	×
EC	✔	×	×	×	×	✔	✔	✔
RPG	✔	×	✔	✔	×	×	×	×
ChE	✔	×	✔	✔	×	✔	×	✔
ChPE	✔	×	✔	✔	✔	✔	✔	✔

Figure 1: The properties attained by the various mechanisms.

$t)E[T_{-i}|T_{-i} \leq t] + P(T_{-i} > t)E[T_{-i}|T_{-i} > t] = E[T_{-i}]$. Hence, i cannot affect the expectation of its h_i payment. □

We can now prove the main positive result.

Theorem 2. *Any direct-revelation scheduling mechanism where i receives the RPG payment plus an h term whose expectation i cannot affect is ex-post truthful. Hence, RPG, ChE, and ChPE are all ex-post truthful.*

Proof. Because machines are assumed to be risk-neutral, they care only about the expectation of their h_i term, and since they cannot affect this, the h_i term does not affect their incentives at all. Hence, it suffices to prove that RPG is ex-post truthful. In RPG, machine i's utility is $-r_i-(r_N-r_i) = r_N$. Hence, the best that i can hope for is the social planner's solution that minimizes the expected processing time. But, machine i can achieve exactly this by behaving truthfully, because the mechanism chooses the social planner's solution for the reported distributions, and in ex-post equilibrium, everyone else can be assumed to behave truthfully. □

Additional Properties of Proposed Mechanisms

The desirable properties that we consider here are:

- *No negative payments (NNP)*: the agents receive payments that are never negative.
- *No negative expected payments(NNEP)*: for any profile of (true) distributions, each agent's *expected* payment is nonnegative assuming truthful behavior (even if the realization can sometimes be negative).
- *Individual rationality (IR)*: for any profile of (true) distributions, each agent's *expected* utility is nonnegative assuming truthful behavior (even if the realization can sometimes be negative).
- *Zero payment upon failure (ZPUF)*: a machine that is not the one to finish the job always gets payment 0.

Naturally, NNP implies NNEP. IR also implies NNEP: contrapositively, if an agent has a negative expected payment for some profile, its expected utility must also be negative, because its processing costs can only bring its expected utility down further. A stronger version of IR—where an agent's *realized* utility is never negative—seems too much to ask for, for the following reason. Presumably, Clarke-type mechanisms should give an agent who has extremely low hazard rates, and thus will never be asked to process, an expected utility of zero. But if the mechanism satisfies this stronger notion of IR, such an agent would have nothing to lose, and presumably something to gain, from reporting higher hazard rates and participating in the processing. The same issue would occur for a stronger notion of truthfulness where an agent never regrets telling the truth even *after* learning T_N.

Proposition 5. *(a) EPG and RPG do not satisfy any of NNP, NNEP, IR, and ZPUF.*
(b) EC and ChPE satisfy NNP (and hence NNEP). ChE satisfies NNEP, but not NNP.
(c) EC, ChE, and ChPE satisfy IR.
(d) ChPE satisfies ZPUF; EC and ChE do not.

Proof. (a) Any machine that is not sure to finish the job completely by itself will have a negative expected payment, implying that NNEP (and hence NNP and IR) are violated. A machine that does not finish must also have a negative payment, so ZPUF is violated.

(b) For EC, because the schedule that minimizes expected processing time is always chosen, i's being present can never increase the expected processing time of the other agents $-i$. For ChPE, i's payment is either 0, or $-r_{N\setminus\{i\}} + r_{N\setminus\{i\}} + E_{T_{-i}\sim\hat{f}_{-i}}[T_{N\setminus\{i\}} - r_{N\setminus\{i\}}|T_{N\setminus\{i\}} > r_{N\setminus\{i\}}]$ which must be positive. Finally, for any given profile, assuming truthful behavior, ChE has the same *expected* payments as EC and ChPE. ChE can have negative *realized* payments when processing happens to take much longer than expected, which increases $r_N - r_i$ but does not affect $h_i(f_{-i})$.

(c) For any given profile, assuming truthful behavior, an agent i gets the same *expected* utility in all three of these mechanisms. Hence, WLOG, we can focus on ChE. The expected utility for agent i satisfies $u_i(f_1,\ldots,f_n) = E_{T_1\sim f_1,\ldots,T_n\sim f_n}[-R_i - (R_N - R_i)] + h_i(f_{-i}) = E_{T_1\sim f_1,\ldots,T_n\sim f_n}[-T_N] + E_{T_{-i}\sim f_{-i}}[T_{N\setminus\{i\}}] \geq 0$.

The last inequality holds because the schedule always minimizes expected processing time, so having an additional agent never increases the expected running time.

(d) It is immediate that ChPE satisfies ZPUF. EC results in positive payments for all machines, as long as each one reduces the expected processing time. For ChE, if the job gets finished in the first round by some machine, the others will receive positive payments. □

Future Research

Our work leaves several directions for future research. One is to fully characterize of the class of mechanisms that are ex-post truthful: is it the case that in every such mechanism, each machine receives the RPG payment, plus an h term whose expectation it cannot affect? Another is to find an axiomatic characterization of the ChPE mechanism. We conjecture that dominant-strategies truthfulness is unattainable under minimal assumptions.

References

Auletta, V.; Prisco, R. D.; Penna, P.; Persiano, G.; and Ventre, C. 2006. New constructions of mechanisms with verification. In Bugliesi, M.; Preneel, B.; Sassone, V.; and Wegener, I., eds., *ICALP (1)*, volume 4051 of *Lecture Notes in Computer Science*, 596–607. Springer.

Caragiannis, I.; Elkind, E.; Szegedy, M.; and Yu, L. 2012. Mechanism design: from partial to probabilistic verification. In *ACM Conference on Electronic Commerce, EC '12*, 266–283.

Clarke, E. 1971. Multipart pricing of public goods. *Public Choice* 8:17–33.

Feige, U., and Tennenholtz, M. 2011. Mechanism design with uncertain inputs: (to err is human, to forgive divine). In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, 549–558.

Fotakis, D., and Zampetakis, E. 2013. Truthfulness flooded domains and the power of verification for mechanism design. In *Web and Internet Economics - 9th International Conference, WINE'13*, 202–215.

Green, J., and Laffont, J.-J. 1986. Partially veriable information and mechanism design. *Review of Economic Studies* 53(1-2):447–456.

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

Johnson, T. R. 2013. Procurement with adverse selection and dynamic moral hazard. *working paper*.

Krysta, P., and Ventre, C. 2010. Combinatorial auctions with verification are tractable. In de Berg, M., and Meyer, U., eds., *ESA (2)*, volume 6347 of *Lecture Notes in Computer Science*, 39–50. Springer.

Nisan, N., and Ronen, A. 2001. Algorithmic mechanism design. *Games and Economic Behavior* 35(1-2):166–196.

Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press.

Porter, R.; Ronen, A.; Shoham, Y.; and Tennenholtz, M. 2008. Fault tolerant mechanism design. *Artif. Intell.* 172(15):1783–1799.

Ramchurn, S. D.; Mezzetti, C.; Giovannucci, A.; Rodríguez-Aguilar, J. A.; Dash, R. K.; and Jennings, N. R. 2009. Trust-based mechanisms for robust and efficient task allocation in the presence of execution uncertainty. *J. Artif. Intell. Res. (JAIR)* 35:119–159.

Stein, S.; Gerding, E. H.; Rogers, A.; Larson, K.; and Jennings, N. R. 2011. Algorithms and mechanisms for procuring services with uncertain durations using redundancy. *Artif. Intell.* 175(14-15):2021–2060.

Vickrey, W. 1961. Counterspeculations, auctions and competitive sealed tenders. *Journal of Finance* 16:8–37.