# The Role of Information in Computational-Resource Allocation, for the TASK REF:
# Dynamic Control of Emergent Behavior in E-Commerce Ecologies

Jonathan Bredin, Daniela Rus, and David Kotz

Department of Computer Science, Dartmouth College

## Abstract

We examine the role of information in markets that allocate computation to software agents. The comparison of two types of markets illuminates the importance of information and the incentives for buyers and sellers to share their preferences with each other. In our comparison, the distinguishing feature of the two markets types is the alignment of agents' interests. We define a closed-interest market as one where resources are collectively owned among the agents. An open-interest market makes no assumptions on the interests of agents or resource owners.

The incentives of agents in the two markets drastically differ. The open-interest model motivates agents to be less trusting and to not share information. This aspect stems from the model's greater applicability to resource allocation, but has a deep impact on system efficiency. In this paper, we summarize some economic theory and allegorical evidence from our models and system implementations that support the claim, and conclude with guidelines for system development.

## 1 Introduction

We look at two types of markets where software agents purchase computational resources, such as computation, communication, and storage, from their hosts. In the first type of market, a closed-interest market, agents share ownership of the resources that they consume. The second type of market is an open-interest market that makes no assumptions on resource ownership. The major difference between the two types of markets is the motivation of hosts. With collectively owned resources, the principals' concerns are that the resources should be fully utilized. When we separate resource ownership and use, a principal's goal becomes revenue maximization. A dramatic result of the separation is evident in the differences in incentives for a software agent and its host to exchange information. While open markets are applicable to more scenarios, there are costs inherent to trade both in terms of the computation involved in allocation and the efficiency of the outcome.

First, we examine some preliminary related theoretical results that led us to consider simple allocation policies. We then summarize two models we developed and relate the applicability of each. Finally, we present our intuition and advice concerning market-based computation allocation.

## 2 Theoretical Background

Mechanism design, the study of rule construction to guide agent behavior, hints that operation in environments with diverse interests is difficult. Theoretical results state that in the presence of uncertainty of preferences, perfect efficiency is not possible. It may not even be feasible or possible to gather agent preferences. Further complications arise when we consider repeated interaction among agents.

The Myerson-Satterthwaite theorem states that complete efficiency is not possible when agents have asymmetric information and there is the possibility that trade yields zero gain to at least one agent [MS83]. Thus we may wish to consider methods to extract agents' preferences in our search for efficient systems. As system designers, we see a possible conflict of interest between the goal of system efficiency and individual rationality. For now, we consider the costs of preference extraction.

The Gibbard-Satterthwaite theorem states that without restrictions on agents' utilities and unless one agent dictatorially decides the outcome for all others, it is impossible to motivate more than two agents to reveal their preferences for allocation [Gib73]. We can, of course, restrict agents' utility functions to reasonable classes, but even so, truthful implementation usually involves payment to each agent to divulge its preference.

The minimum required payment can be large enough to dominate all other issues. An example is evident in the Clarke-Groves mechanism to allocate public goods. It is the only truthfully implementable mechanism for public-good allocation, but it cannot guarantee a balanced budget to the principal [GL77].

To help the selection of a mechanism to extract agents' preferences, we compare the cost of the use of the mechanism with alternative methods. Frequently, the com-

putational cost in implementing the mechanism is infeasible. Nisan applies dominant-strategy game-theoretic approaches to distributively solve traditional computer-science problems [Nis99]. Unfortunately, the allocation problems leverage Vickrey-Clarke-Groves mechanisms that require enormous amounts of computation. Given the computational costs of allocation and the loss of revenue in preference extraction, the principal may be indifferent to the outcome of a dominant-strategy mechanism and the outcome of a simpler one.

There is one more complication. Realistic agents have many goals and will likely repeatedly compete with each other. The aspect of repeated play complicates the analysis of rational behavior in the face of a particular mechanism. Little research on repeated-play mechanism design and the assumptions used to derive strategies in the one-shot game disappear in repeated play. An example of this is the value of an agent's private cost in a repeated auction.

# 3  Market Applications

In the previous section, we presented a litany of problems with market design. We now present our pragmatic approach to computational-resource allocation: we restrict our attention to simple strategies and mechanisms and represent repeated play through optimization of a long game with simple decisions.

Our interests lie in the allocation of computational resources to software agents. Typically, the $i$-th agent must negotiate both the price and priority of its resource consumption in the face of a budget constraint to complete $M_i$ tasks with sizes $q_{ij}, j \in [1 \dots M_i]$. The motivations for studying markets are mainly three-fold:

- An agent with greater potential, expressed by its currency holdings, relative to its job size should have greater priority than an agent with less potential.

- Trade in valuable currency motivates a principal to make its resources publicly available in previously closed environments.

- Optimization of system performance is frequently intractable or infeasible. Markets can provide good distributed heuristics for resource allocation.

We identify the alignment of agents' interests as a parameter that characterizes most resource-allocation problems. In one extreme, a closed-interest environment, all agents cooperate to achieve a common global objective. At the other end of the spectrum, there are no restrictions on an agent's motivations and each agent competes to maximize its private utility. The clearest distinction between the two examples is how likely an agent is to reveal its underlying motivations. For example, an agent may misrepresent its deadlines, job-size estimations, or

priority so that it may achieve a more favorable allocation. The next two subsections summarize bodies of work at each end of the spectrum.

## 3.1  Collective Ownership

In our first example, we investigate a scenario where software agents represent the interests of the people who own a network of computing resources [BMI+00]. We assume that the resource owners wish to utilize the resources as efficiently as possible and that an individual agent's goal is to compute as quickly as possible. We employ a market for computation to prioritize agents as well as to provide a means to balance the load of computation throughout the network.

### 3.1.1  Allocation

In the interest of fairness, we allocate computation such that payment is proportional to allocation. We define the allocation mechanism such that the $i$-th agent submits a function that, given the price of computation at host $j$, returns $u_{ij}$, the rate in currency per unit of time that the agent pays the host for access. The amount of computation available at the $j$-th host the $i$-th agent visits is $c_{ij}$, the $i$-th agent receives a portion

$$x_{ij} = u_{ij} / \sum_{k=1}^{N_j} u_{kj}$$

of the computation, and computes at a rate of $v_{ij} = c_{ij} x_{ij}$ instructions per time unit at a host with $N_j$ agents.

The host determines a price for its computation to satisfy all present agents' bidding contracts. Figure 1 shows an example of the search for an equilibrium price. We plot the price that the host could present for all of its computation, $\theta_1$, versus the amount agents would pay for computation at that price. The point where the line with slope one intersects the market's response satisfies all agents' contracts. In the event that only one agent visits the host, the agent receives all of the computation at no charge.

Our mechanism produces a unique non-trivial allocation so long as each agent's bid contract, $g_i(\theta)$, satisfies the following constraints:

- $g_i(0) = 0$,    - $\partial g / \partial \theta \big|^{\theta=0} = 1$,
- $g_i(\theta > z) = 0$,    - $\partial^2 g / \partial \theta^2 \big|^{\theta \in [0,z]} \leq 0$.

Furthermore, no agent may return its unused endowment to its user.

### 3.1.2  Agent Participation

An agent's utility stems from minimizing its end-to-end latency. Ignoring network latencies, the $i$-th agent's goal is to maximize $U_i$, where
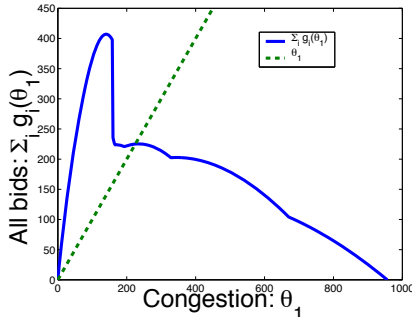
$$U_i = -\sum_{j=1}^{M_i} \frac{q_{ij}}{c_{ij} x_{ij}}.$$

Figure 1: A sample of the sum several agents' bid contracts that form the market's response to the price of computation, $\theta_1$, at a host.

Here our utility has a concrete meaning since an agent's currency has no value outside its immediate set of tasks. We apply Lagrangian relaxation to compute the closed form equation for the agent's bid in terms of the price of computation. The optimization process implies that if an agent bids, it can complete its task set and the bid is optimal in the absence of information relevant to the competing bids.
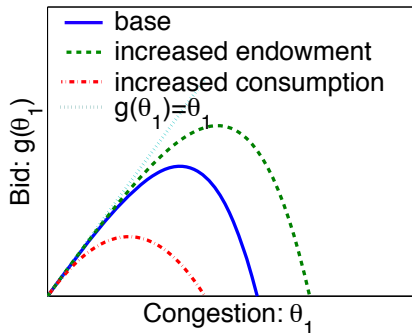


Figure 2: Some sample bidding contracts as functions of the price of computation.

Figure 2 illustrates an example agent bid. Increased endowment has the effect of scaling the bid function, while increasing the agent's future load makes the function's curvature softer and decreases the range over which the agent may bid. Decreasing the agent's future consumption makes the bid function sharper and better approximates the function $g_i(\theta_1) = \theta_1$ along the interval where the agent can afford to compute.

### 3.1.3   Results

We have implemented our allocation policy in a simulated network and compared it with other allocation policies. Empirically, we find 8% to upper bound on the cost of our prioritization and that we improve throughput by 18% in comparison to round-robin resource allocation.

Figure 3 shows how our allocation performs when agent requests exceed system capacity. Our allocation policy determines which agents can be suspended or dropped and still allocates computation to important agents that complete their tasks with reasonable performance.
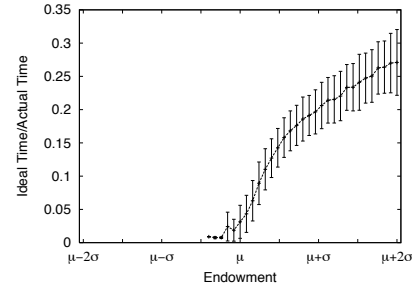


Figure 3: Agent endowment versus performance in an overloaded system with shared resource ownership. We plot endowment/job-size ratios two standard deviations, $\sigma$, around the mean endowment/job-size ratio, $\mu$.

Our bidding algorithm relies on estimations of the agent's task sizes. We find, however, that our bidding procedure is robust to errors in estimation error. Figure 4 demonstrates the effect of estimation error upon agents' performance. We plot error as the standard deviation of the estimate versus average agent performance. The graph plots error for five multiples of the actual job-size mean and shows that every factor only decreases performance by about 3%.
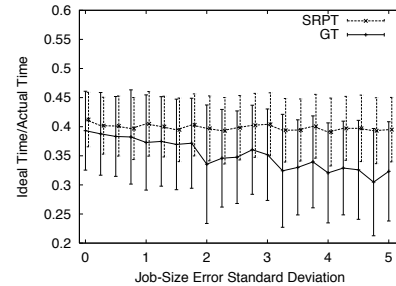


Figure 4: Agent job-size estimation error versus throughput for our game-theoretic and shortest-job-remaining optimal allocation policies.

Cooper and Gray [CG00] implemented the allocation policy for a version of D'Agents that runs on top of QLinux.

## 3.2   Revenue Maximization

We now move to resource allocation in an environment where a host's sole objective is revenue maximization. We implement a simple mechanism with this objective.

### 3.2.1   Allocation

The allocation mechanism that we choose is a sealed-bid auction. The $j$-th host has a private value, $v_j$, for its

own computation and holds an auction for access to that computation for the next $\Delta$ seconds. Each agent submits a single price-quantity pair, $(t_{ij}, x_{ij})$, to the host. The host can maximize its revenue through selection of a set of bids where each has a price-per-unit density, $t_{ij}/x_{ij} \geq v'_j$, where $v'_j$ is greater than $v_j$ and depends on the host's prior beliefs on the demand for computation [FT96]. The host's utility is then

$$\begin{aligned} & \max \left(\sum_i t_{ij} x_{ij} y_i \ + \ v_j - \sum_i x_{ij} y_{ij}\right) \\ \text{such that} \quad & \sum_i x_{ij} y_i \leq 1, \quad y_i \in \{1, 0\}. \end{aligned}$$

Optimization requires the host to solve an NP-complete integer programming problem. For reasonable numbers of visiting agents, we can apply approximation algorithms to solve the problem with little overhead.

### 3.2.2 Agent Utility

Until now, we have not considered an agent's utility in the open-interest market. We would like to have savings be a positive value in our model given that resource owners and agent owners are separate entities. We choose a quasi-linear utility for the $i$-th agent to be

$$U_i = E_i \bigg/ \left(1 + \exp\left(\sum_{j=1}^{M_i} \frac{\kappa_i q_{ij}}{c_{ij} x_{ij}} - \tau_j\right)\right) - \sum_{j=1}^{M_i} t_{ij},$$

where $x_{ij}$ is the agent's average share of computation at the $j$-th host, $t_{ij}$ is the payment to the host for the $j$-th task, $E_i$ is the agent's initial endowment, $\tau_j$ is the user's expectation for the $j$-th task's completion time, and $\kappa_i > 0$ is the precision of the expectations. The left-hand side of the difference is a sigmoid with respect to execution time and the right is the total currency expenditure.

This utility function is different from the closed-interest model in several ways. The function models savings benefits, but quasi-linear utility functions have no wealth effect. That is, an agent's consumption does not change with its endowment. For this reason, we have changed the utility to correspond with a weighted difference of the agent's ability to complete its tasks in time less than expectation with the currency spent to complete the tasks.

### 3.2.3 Agent Participation

The domain over which an agent can bid is large. In this subsection, we show a heuristic that prunes the bid space. We then show how the agent can build a belief function that can optimize its utility.

The host's algorithm to decide which bids to accept leverages the price per unit of computation, or density, of a bid. For our utility function, we can derive the highest density bid for any level of utility. Figure 5 plots several isoquants, a set of payment-allocation pairs that yield identical utility, and a curve that plots the densest bid for each level of utility. This curve shows the one dimensional space upon which we collapse the agent's bidding space.
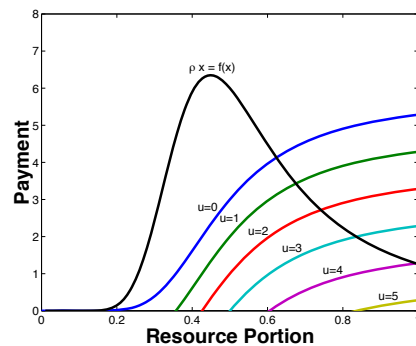


Figure 5: A set of isoquants and the set of bids that maximizes price-per-unit of computation at any level of utility.

The agent can look at the history of submitted bids and map each bid to one of equal utility, but higher density. We use the resulting set of mapped bids build a belief function that returns the probability of a bid's acceptance in a similar fashion used by Gjerstad and Dickhaut [GD98].

With a belief function and a single dimensional bidding space, the agent can fix its expectations and optimize its utility by maximization of

$$\begin{aligned} & \mathrm{E}[U_i(x_{i1}, t_{i1} | x_{j \neq 1}, t_{j \neq 1})] \\ & = \sum_{k=a}^{\infty} \binom{k-1}{a-1} p^a q^{k-a} U_i(ac_{ij}k/q_{ij}, t_{i1} | x_{j \neq 1}, t_{j \neq 1}), \end{aligned}$$

where $a$ is the number of auctions that the agent must win to complete its next task, $p$ is the probability that the agent's bid is accepted, and $q$ is the complement. Both probabilities are functions of $x_{i1}$ and $t_{i1}$. The program computes the sum of the products of the probabilities that an agent will take exactly $k$ auctions to win $a$ of them and the utility from finishing the task in $k$ attempts. We can approximate the value of the sum using our knowledge of hyper-geometric series and Taylor series expansions to compress the optimization to a one-dimensional search.

### 3.2.4 Results

We simulate agents that use our bidding procedure and operate under the repeated sealed-bid auction allocation policy. In light of the closed-interest model's behavior to over-constrained resource allocation, we ran the same experiment in our open-interest market model under a similar load. Because the utility function differs between the two models, instead of throughput, we measured an agent's probability of successfully completing its itinerary.

Figure 6 plots endowment versus observed success rates. As in the closed-interest model, an agent's performance is related to its relative endowment, though unlike the closed-interest market, many poorer agents sneak through. In the open-market model, there is another factor that correlates with performance: the number of sites an agent must visit. Figure 7 plots the success rates of
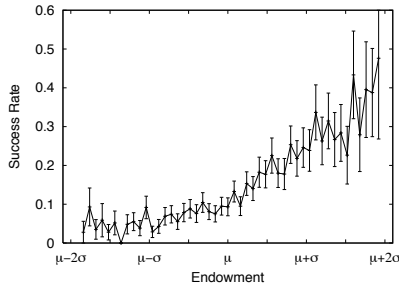
Figure 6: The relationship between endowment and agent completion rate in an overloaded simulation of the open-interest market.
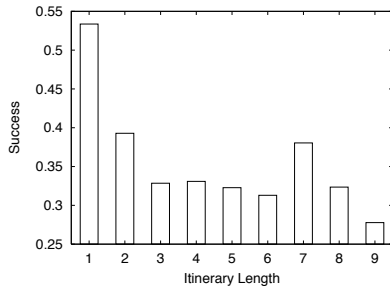


Figure 7: Agent success rate conditioned on itinerary length in an overloaded simulation.

agents in the simulation versus the number of hosts that they intended to visit. The repeated, independent encounters with different hosts add risk to an agent's performance.

## 4 Conclusions

We reason about applications of markets to solve resource allocation in distributed systems where resource owners and consumers either share or do not share utility. We look at allocation of computation to software agents in two scenarios: a "closed-interest" scenario where users own resources; and an "open-interest" situation where users and resource owners are distinct groups.

Markets provide a means of prioritization of jobs through endowments. Because resource consumption is tied to currency expenditure, markets implement an element of fault tolerance and load balancing. Additionally, an individual agent can optimize its performance through budgeting its expenditures.

We observe that we can achieve more efficient allocation in the closed-interest markets. In our examples, the closed market allocates all available resources, while the open market typically only allocates 40% of the available resources, despite agent requests are 160% of the resources. Part of the efficiency disparity is due to the open-market agents' greater constraints, but clearly if agents and their hosts were more articulate in exchanging goals, we could arrive at a better allocation.

While open-markets apply to more scenarios, the implementation costs are higher than closed markets. An agent's performance in a closed market depends heavily on the accuracy of its prior beliefs of the state of the market. Evaluation of an agent's savings is also complicated and simple, commonly-used quasi-linear utility models do not capture the wealth effect. On the seller's, as well as the buyer's side, open markets frequently involve greater computation than analogs in closed markets. From these observations, we conclude that utility choice and market structure are important system design issues. Engineers should carefully consider alternatives to markets, of course, but should also keep in mind agent goals and co-locate resource ownership and usage interests as much as possible.

## References

[BMI+00] Jonathan Bredin, Rajiv T. Maheswaran, Cagri Imer, Tamer Başar, David Kotz, and Daniela Rus. A game-theoretic formulation of multi-agent resource allocation. In *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, June 2000.

[CG00] Ezra E. K. Cooper and Robert S. Gray. An economic CPU-time market for D'Agents. Technical Report TR2000-375, Dartmouth College, June 2000. Undergraduate honors thesis. Advisor: Bob Gray.

[FT96] Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, Cambridge, MA, 1996.

[GD98] Steven Gjerstad and John Dickhaut. Price formation in double auctions. *Games and Economic Behavior*, 22(1):1–29, January 1998.

[Gib73] Allan Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, July 1973.

[GL77] Jerry Green and Jean-Jacques Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45(2):427–438, March 1977.

[MS83] Roger B. Myerson and Mark A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28:265–281, 1983.

[Nis99] Noam Nisan. Algorithms for selfish agents – mechanism design for distributed computation. In *Proceedings of the Symposium on Theoretical Aspects in Computer Science*, Trier, Germany, March 1999.