

Brief Announcement: Distributed Algorithmic Mechanism Design for Scheduling

Thomas E. Carroll
tec@cs.wayne.edu

Daniel Grosu
dgrosu@cs.wayne.edu

Department of Computer Science
Wayne State University
Detroit, Michigan 48202, USA

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems – *Sequencing and Scheduling*; J.4 [Computer Applications]: Social and Behavioral Sciences – *Economics*

General Terms: Design, Economics, Theory

Keywords: mechanism design, task scheduling, truthful mechanism, distributed computation

The Internet has emerged as the global platform for computation and communication. It offers nearly unlimited resources that could be utilized to solve a vast array of problems. The Internet resources are controlled and operated by a multitude of self-interested, independent parties. These agents may manipulate the protocols in their own benefit and their selfish behavior may lead to degraded performance and reduced efficiency. Solving such problems involving selfish agents is the object of *mechanism design theory*.

Algorithmic Mechanism Design (AMD) [2] combines incentive compatibility with computational tractability. AMD assumes a centralized trusted party which computes the outcomes of the mechanism. This model cannot be always used to solve problems that are inherently distributed, thus creating the necessity of distributed implementations. *Distributed Algorithmic Mechanism Design (DAMD)* has the same goals as AMD, but, in addition, it assumes a model in which the information and the computation are inherently distributed.

In their seminal paper [2], Nisan and Ronen proposed *MinWork*, an n -approximation scheduling mechanism. The mechanism objective is to allocate a set of tasks to a set of agents such that the completion time of the last assignment is minimized. The agents are compensated for performing the allocated work. It assumes a *central authority* that collects the bids and computes the schedule and payments. The central authority is problematic as it must be trusted by all parties and it has limited scalability and reliability.

We propose a distributed implementation of *MinWork* called *Distributed MinWork (DMW)*. DMW does not employ a central authority, but instead, the agents themselves compute the schedule. A set of distributed auctions determine the allocation and payment, while protecting the anonymity of the losing agents and the privacy of their bids.

DMW concurrently executes m secure distributed second-price auctions where m is the number of tasks to be scheduled. Our design is based on the idea proposed by Kikuchi [1]. The agents encode their bids in the degree of randomly chosen polynomials and then submit shares of the polynomials to the other participants. The shares are inputs to multi-party computations that reveal the winner and the second price. A description of the mechanism follows.

For each task, agent i (for $i = 1, \dots, n$) randomly chooses the polynomials $f_i(x), G_i(x)$, and $h_i(x)$ such that, for the largest permissible bid \bar{b} , the bid of agent i , b_i , and the maximum number of faulty agents c , $\text{degree}(f_i) = \bar{b} - b_i + c$, $\text{degree}(h_i) = \bar{b} + c$, and $\text{degree}(G_i) = \text{degree}(f_i) + \text{degree}(h_i)$. Agent i transmits the shares $f_i(\alpha_j), G_i(\alpha_j)$, and $h_i(\alpha_j)$ (for $j = 1, \dots, n$) to agent j , where α_j is a pseudonym for agent j . The agent anonymously publishes its commitments to the shares, which bind it to the terms in the polynomials, thus prohibiting it for altering or repudiating its bid. All agents implicitly synchronize, waiting until all shares are transmitted and commitments published.

The agents verify that the received shares are consistent with the commitments. If the commitments hold, agent i publishes $\Lambda_i = g^{F(\alpha_i)}$ where $g \in \mathbb{Z}_p^*$ is a generator of order q and $F(\alpha_i) = \sum_{j=1}^n f_j(\alpha_i)$. Let $F^{(s)}(0)$ be the s -th interpolation of polynomial F . The bid resolution protocol begins by computing the smallest t such that $\prod_{j=1}^t (\Lambda_j)^{\rho_j} = g^{F^{(t)}(0)} = 1$, where $\rho_j = \prod_{i \neq j} \frac{\alpha_i}{\alpha_i - \alpha_j} \pmod{q}$. The first-price bid is $b_* = \bar{b} - t + c$. The winner is decided by a subset of agents revealing $G_j(\alpha_i)$. There will be an agent such that $G_*^{(u)}(0) = 0$, which proves its bid is the lowest. The second price b_{**} is computed by excluding the winner from $\Lambda_i, \Lambda_i = \frac{\Lambda_i}{g^{F_*(\alpha_i)}}$, and then repeating the bid resolution protocol.

The schedule is obtained by allocating the task to the winner of the related auction. The payment handed to an agent is the sum of the second prices of all auctions that it won.

DMW exhibits the strategic properties inherent to *MinWork*. It preserves the anonymity of the losing agents and the privacy of their bids by using a secret-sharing scheme.

REFERENCES

- [1] H. Kikuchi. $(M + 1)$ st-price auction protocol. In *FC'01: Proc. of the 5th Intl. Conf. on Financial Cryptography*, pages 351–363. Springer-Verlag, 2002.
- [2] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1):166–196, Apr. 2001.