Distributed Algorithmic Mechanism Design for Scheduling on Unrelated Machines[*]

Thomas E. Carroll
Dept. of Computer Science
Wayne State University
5143 Cass Avenue, Detroit, MI 48202.
tec@cs.wayne.edu

Daniel Grosu
Dept. of Computer Science
Wayne State University
5143 Cass Avenue, Detroit, MI 48202.
dgrosu@cs.wayne.edu

Abstract

In classical mechanism design setting the outcome of the mechanism is computed by a trusted central party. In this paper we consider distributed implementations in which the outcome is computed by the agents themselves. We propose a distributed mechanism for solving the problem of scheduling on unrelated machines. This mechanism, called Distributed MinWork (DMW) is a distributed implementation of the MinWork mechanism proposed by Nisan and Ronen [12]. DMW is resilient to agents deviating from the protocol since the cheaters are detected and eliminated from further participation. We show that DMW is truthful, since it preserves the incentives and allocation policies from MinWork. Finally, we show that DMW has polynomial communication and computation costs.

1. Introduction

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* (also called *implementation theory*)[13]. This theory helps design protocols in which the agents are given incentives to tell the truth and follow the rules. Such mechanisms or protocols are called *truthful* or *strategyproof*. Each participating agent has a privately known function called *valuation* which quantifies the agent benefit or loss. The valuation depends on the outcome and is reported to a centralized mechanism. The mechanism chooses an outcome that maximizes a given objective function and makes payments to the agents. The valuations and payments are expressed in some common unit of currency. The payments are designed and used to motivate the agents to report their true valuations. Reporting the true valuations leads to an optimal value for the objective function. The goal of each agent is to maximize the sum of her valuation and payment.

Traditionally, we were mostly concerned with the *computational tractability*, but now we must jointly address *incentive compatibility*. This led to the development of *algorithmic mechanism design (AMD)* [12]. The AMD model 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)* [6] has the same objectives as AMD, but in addition the model is characterized by the distribution of the agents, information, and computation. In DAMD agents themselves determine the outcome of the mechanism. This solution creates new opportunities for the agents to manipulate the protocols. Any distributed implementation must address this problem in addition to the other issues form the standard AMD.

In their seminal paper, Nisan and Ronen proposed the *strongly truthful MinWork* [12] mechanism. The objective of MinWork is to allocate a batch of tasks to a set of agents such that the completion time of the last assignment is minimized. It assumes a *trusted central administrator* with which the agents freely communicate. The administrator accepts the agents' bids and computes the schedule as a function of the bids. The problem associ-

[*] This research was supported in part by Wayne State University Faculty Research Grant

ated with this approach is that the administrator must be trusted by all parties and has limited scalability and reliability.

We propose a distributed variant of MinWork called *Distributed MinWork (DMW)*. DMW does not use a central administrator, but instead, the agents themselves compute the schedule. Our solution is based on the idea presented in [2, 8]. A set of simultaneous distributed auctions determine the allocation and the payment function, while protecting the anonymity of the losing agents and the privacy of their bids. The mechanism is *resilient* to deviation in the sense that cheaters are detected and eliminated from participating. DMW has low communication and computation complexity.

Related Work. Nisan and Ronen [12] introduced *algorithmic mechanism design (AMD)*, which bridged *computational tractability* with *incentive compatibility*. In their work, they used the celebrated *Vickrey-Clarke-Groves (VCG)* [3, 7, 18] mechanism in designing mechanisms for several standard problems in computer science. They proposed a *polynomial-time* mechanism called MinWork, which is a n-approximation (where n is the number of agents) to the scheduling problem.

Feigenbaum *et al.* [4, 5] initiated the study of DAMD for cost sharing in *multicast trees* and lowest-cost *unicast routing*. Mitchell and Teague [9] extended the multicast mechanism by devising a new model where the nodes are controlled by the strategic agents and are permitted to deviate from the protocol. Cheating is detected through the use of digital signatures and when detected, results in heavy fines. Monderer and Tennenholtz [10] studied a single-item auction problem considering a model in which agents communicate with the trusted central party by forwarding messages through other agents.

Ng *et al.* [11] broadens DAMD to envision competing distributed mechanisms in an open marketplace. Shneidman and Parkes [17] studied the problem of creating distributed system specifications that will be faithfully implemented in systems composed of rational nodes. A faithful implementation guarantees that no node will deviate from the specifications. Parkes and Shneidman [14] considered distributed mechanism implementations. They proposed a set of principles that can be used to develop faithful implementations. Brandt and Sandholm [1] investigated how existing cryptographic building blocks can be used to ensure the correctness and the privacy in distributed mechanisms.

Contributions. We propose a distributed scheduling mechanism called *Distributed MinWork (DMW)* which is a distributed implementation of the centralized scheduling mechanism proposed in [12]. The agents themselves compute the allocation and payment functions by participating in simultaneous Vickrey auctions. Our design is based on the idea proposed in [2, 8]. An agent encodes its bid in the degree of a randomly chosen polynomial and then transmits the shares to the other agents. The shares are used by each agent as input to a degree resolution procedure. By determining the degree, an agent is able to determine the winner, the first price, and the second price. The identities and the bids of the loosing agents are not revealed. DMW is *resilient* to agents deviating from the protocol since the cheaters are detected and eliminated from further participation. We show that DMW is a truthful mechanism, since it preserves the incentives and allocation policies from MinWork. Finally, we show that DMW has polynomial communication and computation costs.

Considering the scheduling on unrelated machines problem, DMW provides insight into *Open Problem 10, 11 and 12* from [6]. Regarding Open Problem 10, the centralized MinWork can be simply distributed among *obedient nodes*. By using *cryptographic protocol-design*, we are able to develop a distributed mechanism with a much more realistic *strategic model* that tolerates *strategic* and *adversarial nodes*. Regarding Open Problem 11, DMW tolerates protocol deviations. As long as the number of agents obeying the protocol remains above a threshold, the mechanism is computable. If the number of agents drops below the threshold, the mechanism cannot be executed. Regarding Open Problem 12, DMW efficiently protects the privacy of the agents. It is, however, imperfect since it reveals some information (see Theorem 4.2). The set of participating agents must be known, but this knowledge is intrinsic to the scheduling problem.

Organization. The paper is structured as follows. In Section 2 we present the theory and primitives on which our distributed scheduling mechanism is based. In Section 3 we present and detail our proposed mechanism. In Section 4 we discuss the merits and costs. In Section 5 we draw conclusions and present future directions.

2. Preliminaries

2.1. The scheduling problem

We consider here the problem of scheduling $m \geq 1$ independent tasks $T_1, T_2, \ldots, T_m$ on $n \geq 1$ agents (machines) $A_1, A_2, \ldots, A_n$. Each task T_j is characterized by its processing requirement of r_j units of time. Each agent A_i can process task T_j at speed $s_{ij} > 0$. Thus task T_j would take $t^i_j = r_j/s_{ij}$ time to be processed by A_i. A schedule $\mathbf{S}$ is a partition of the set of tasks indices into disjoint sets $S^i, i = 1, \ldots, n$. Partition S^i contains the indices corresponding to all the tasks allocated to A_i. The goal is to obtain a schedule $\mathbf{S}$ minimizing a given objective function such as makespan, sum of completion times, etc. In the scheduling literature, this problem is known as *scheduling on unrelated machines* [15].

2.2. Scheduling Mechanisms

In this section we first formally describe the scheduling mechanism design problem and then present the scheduling mechanism.

Definition 2.1. (Mechanism design problem) The problem of designing a scheduling mechanism is characterized by:

(i) A finite set $\mathcal{S}$ of allowed outputs. The output is a schedule $\mathbf{S}(\mathbf{b}) = (S^1(\mathbf{b}), S^2(\mathbf{b}), \ldots, S^n(\mathbf{b}))$, $\mathbf{S}(\mathbf{b}) \in \mathcal{S}$, computed according to the agents' bids, $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$. Here, $\mathbf{b}_i = (b^i_1, b^i_2, \ldots, b^i_m)$ is the vector of values (bids) reported by agent A_i to the mechanism.

(ii) Each agent A_i, $(i = 1, \ldots, n)$, has for each task T_j a privately known parameter t^i_j $(j = 1, \ldots, m)$ called the *true value* which represents the time required by agent A_i to execute task T_j. The preferences of agent A_i are given by a function called *valuation* $V_i(\mathbf{S}(\mathbf{b}), \mathbf{t}_i) = -\sum_{j \in S^i(\mathbf{b})} t^i_j$ (i.e. the negation of total time it takes to complete all tasks assigned to it). Here $\mathbf{t}_i = (t^i_1, t^i_2, \ldots, t^i_m)$.

(iii) Each agent goal is to maximize its *utility*. The utility of agent A_i is $U_i(\mathbf{b}, \mathbf{t}) = P_i(\mathbf{b}, \mathbf{t}) + V_i(\mathbf{S}(\mathbf{b}), \mathbf{t}_i)$, where P_i is the payment handed by the mechanism to agent A_i. Here $\mathbf{t} = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n)$.

(iv) The goal of the mechanism is to select a schedule $\mathbf{S}$ that minimizes the make-span, $C_{max} = \max_i \sum_{j \in S^i(\mathbf{b})} t^i_j$.

An agent A_i may report a value (bid) b^i_j for a task T_j different from its true value t^i_j. The true value characterizes the actual processing capacity of agent A_i. The goal of a truthful scheduling mechanism is to give incentives to agents such that it is beneficial for them to report their true values. Now we give a formal description of a mechanism and define the concept of truthful mechanism.

Definition 2.2. (Mechanism) A *mechanism* is a pair of functions:

(i) The *allocation function*: $\mathbf{S}(\mathbf{b}) = (S^1(\mathbf{b}), S^2(\mathbf{b}), \ldots, S^n(\mathbf{b}))$. This function has as input the vector of agents' bids $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ and returns an output $\mathbf{S} \in \mathcal{S}$.

(ii) The *payment function*: $P(\mathbf{b}, \mathbf{t}) = (P_1(\mathbf{b}, \mathbf{t}), P_2(\mathbf{b}, \mathbf{t}), \ldots, P_n(\mathbf{b}, \mathbf{t}))$, where $P_i(\mathbf{b}, \mathbf{t})$ is the payment handed by the mechanism to agent A_i.

Definition 2.3. (Truthful mechanism) A mechanism is called *truthful* or *strategyproof* if for every agent A_i of type $\mathbf{t}_i$ and for every bids $\mathbf{b}_{-i} = (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n)$ of the other agents, the agent's utility is maximized when it declares its real value $\mathbf{t}_i$ (i.e. truth-telling is a dominant strategy).

Nisan and Ronen [12] designed a truthful mechanism for this problem. The authors provided an approximation mechanism called MinWork, minimizing the total amount of work. MinWork is a truthful mechanism. In terms of makespan it achieves an approximation ratio of n. In the following we present the MinWork mechanism.

Definition 2.4. (MinWork Mechanism) [12] The mechanism that gives an approximate solution to the scheduling problem is defined by the following two functions:

(i) The *allocation function*: each task is allocated to the agent who is able to execute the task in a minimum amount of time (Allocation is random when there are more than one agent with minimum type).

(ii) The *payment function* for agent A_i given by:

$$P_i(\mathbf{b}, \mathbf{t}) = \sum_{j \in S^i(\mathbf{b})} \min_{i' \neq i} t^{i'}_j \tag{1}$$

The MinWork mechanism can be viewed as running separate Vickrey auctions simultaneously for each task. In this paper we design a distributed version of MinWork.

2.3. Polynomial Degree Resolution

In our distributed mechanism each agent encodes its bid in the degree of a random polynomial. The agents will collaboratively determine the winning bid by performing a polynomial degree resolution procedure based on Lagrange interpolation. In the following we present the basic concepts related to polynomial degree resolution.

Definition 2.5. (Lagrange interpolation) Let f be a polynomial of degree d, $f(x) = a_0 + a_1 x + \ldots + a_d x^d$, where $a_0, a_1, \ldots, a_d \in \mathbb{Z}^*_p$ and $\mathbb{Z}^*_p$ is the multiplicative group of integers modulo p. The s-th Lagrange interpolation of f, denoted by $f^{(s)}(0)$, is defined as:

$$f^{(s)}(0) = \sum_{j=1}^{s} f(\alpha_j) \prod_{i \neq j} \frac{\alpha_i}{\alpha_i - \alpha_j} \tag{2}$$

where $\alpha_1, \ldots, \alpha_s \in \mathbb{Z}_p^*$ are distinct, non-zero interpolation points.

We can determine the degree of f as the least s that satisfies $f^{(s)}(0) = f(0)$. If $s = d$ then the d-th interpolation always satisfies $f^{(d)}(0) = f(0)$. If $s > d$ and assuming random picking of the polynomial coefficients from $\mathbb{Z}_p^*$ then the degree resolution mistakenly succeeds with probability $\frac{1}{p}$. This probability is very small assuming a very large prime p.

An efficient algorithm for computing $f^{(s)}(0)$ is given below [2].

1. Step 1: $\psi_k = \frac{f(\alpha_k)}{\prod_{i \neq k}(\alpha_k - \alpha_i)}$, for $i = 1, \ldots, s$
2. Step 2: $\phi(0) = \prod_{k=1}^{s}(\alpha_k)$
3. Step 3: $f^{(s)}(0) = \phi(0) \sum_{k=1}^{s} \frac{\psi_k}{\alpha_k}$

If we assume that the cost of multiplication operation is equal to the cost of computing the multiplicative inverse, it is obvious that the computational complexity of the algorithm is $s^2 + s$. In our proposed mechanism we will use a distributed version of this algorithm. In the distributed version, each agent A_i broadcasts ψ_i/α_i to the other agents. The rest of the computation is performed locally by each agent.

3. The Distributed Scheduling Mechanism

The MinWork mechanism proposed in [12] requires a *trusted central administrator* to which each agent reports its private information. Based on the information received from the agents the central administrator determines the allocation and informs the agents about the allocation. There are several drawbacks of using a central administrator. First, the administrator must be trusted by all the agents that it will properly accept bids and compute the resulting schedule. This is problematic when several autonomous administrative domains must cooperate to choose an administrator. Second, it offers limited scalability. The administrator is a bottleneck, it performs most of the computations and it is either the sender or receiver in all communications. Third, the administrator is a single point of failure which offers very limited reliability.

In a *Distributed scheduling mechanism* there is no central administrator involved, but instead, the agents collaborate among themselves to compute the schedule and the payments. Agents' bids implicitly contain private information (*e.g.*, the bids provide insight into an agent's processing capabilities). Scheduling mechanisms must ensure the privacy of the agents and their bids, and that any exposed information cannot be used to manipulate the outcome.

We propose a distributed scheduling mechanism called *Distributed MinWork (DMW)*. To design the DMW mechanism we exploit the fact that MinWork can be viewed as running separate Vickrey auctions for each task simultaneously. We base our design on the idea presented in [2, 8]. An agent encodes its bid in the degree of a randomly chosen polynomial and then transmits the shares to the other agents. The shares are used by each agent as input in the degree resolution procedure. By determining the degree, an agent is able to determine the winner, the first price, and the second price. Disclosing all this information is unnecessary, but in Theorem 4.2, we show that for this specific problem the risk is minimal. DMW is truthful as it preserves the incentives and allocation policies of MinWork.

Bid encoding. In the existing secret sharing protocols [16], the information is encoded in the free term of a polynomial, but in the scheme devised by Kikuchi [8], the bids are encoded in the degree of the polynomial. The benefit is that the polynomials can be summed and degree resolution on the sum of the polynomials reveals the minimum bid while concealing the other bids. In our mechanism the degree of the polynomial is inversely related to the bids, *i.e.*, small bids are encoded in large-degreed polynomials and vice versa. This secret sharing technique is based on the *threshold trust* model. To increase resilience, the bids are summed before encoding with the *maximum number of faulty agents* denoted by c. At least c agents are required to collude in order to expose any bids. If fewer agents collude, no bids are exposed.

In the following we describe our Distributed MinWork (DMW) Mechanism.

Notations. The following notations are used in describing DMW.

- p, q are large primes such that $q \mid p - 1$.
- All operations, unless indicated, are $\pmod p$.
- $g_1, g_2 \in \mathbb{Z}_p^*$ are distinct generators of order q.
- $\{T_1, \ldots, T_m\}$ is the set of m tasks, where T_j is task j.
- $\{A_1, \ldots, A_n\}$ is the set of n agents, where A_i is agent i.
- $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$ is the set of pseudonyms, where $\alpha_i \in \mathbb{Z}_p^*$ is a pseudonym for A_i such that $\alpha_i \neq \alpha_k$ for all $i \neq k$.

- c is the maximum number of faulty agents.
- $\mathcal{W} = \{w_1, \ldots, w_k\}$ is the set of discrete bid values, where $0 < w_i \leq n + c$.

Distributed MinWork mechanism (DMW):

Initialization: The parameters $p, q, g_1, g_2, c, \mathcal{A}$, and $\mathcal{W}$ are published.

Bidding: For each task T_j ($j = 1, \ldots, m$), agent A_i ($i = 1, \ldots, n$) determines its bid $b^i_j \in \mathcal{W}$. It chooses the following random polynomials:

$$f^i_j(x) = \sum_{k=1}^{t^i_j} a^i_{j,k} x^k, \quad G^i_j(x) = \sum_{k=1}^{s-t^i_j} d^i_{j,k} x^k, \quad (3)$$

$$h^i_j(x) = \sum_{k=1}^{s} c^i_{j,k} x^k$$

where $t^i_j = \max \mathcal{W} - b^i_j + c$ and $s = \max \mathcal{W} + c$. It anonymously and securely transmits the shares $f^i_j(\alpha_k), G^i_j(\alpha_k), h^i_j(\alpha_k)$ to agent A_k ($k = 1, \ldots, n$), who keeps the shares private. Agent A_i anonymously publishes its commitment $\mathcal{E}^i_j = \{E^i_{j,1}, \ldots, E^i_{j,s}\}$, where $E^i_{j,l} = (g_1)^{a^i_{j,l} d^i_{j,l}} (g_2)^{c^i_{j,l}}$. Agents implicitly synchronize at this point; they cannot continue until all shares are transmitted and commitments published. Due to the commitments, agents are unable to alter or repudiate their submitted bids.

Allocating tasks: The agents collaboratively compute the auction outcome for each task T_j ($j = 1, \ldots, m$). Agent A_i ($i = 1, \ldots, n$) verifies that the shares it received from A_k are consistent with the commitments as

$$(g_1)^{f^k_j(\alpha_i) G^k_j(\alpha_i)} (g_2)^{h^k_j(\alpha_i)} = \prod_{l=1}^{s} (E^k_{j,i})^{(\alpha_k)^l}, \quad (4)$$

If the commitments hold, A_i publishes $\Lambda^i_j = (g_1)^{F^i_j(\alpha_i)}$, where $F_j(\alpha_i) = \sum_{k=1}^{n} f^k_j(\alpha_i)$.

The smallest element $t^*_j \in \{ w \in \mathcal{W} \mid \max \mathcal{W} - w + c \}$ is computed such that

$$\prod_{k=1}^{t^*_j} (\Lambda^k_j)^{\rho_k} = (g_1)^{(F_j)^{(t^*_j)}(0)} = 1, \quad (5)$$

where $\rho_k = \prod_{i \neq k \in \mathcal{A}} \frac{\alpha_k}{\alpha_i - \alpha_k} \pmod q$. The first-price bid is given by $b^*_j = \max \mathcal{W} - t^*_j + c$.

The subset of agents of size $u = s - t^*_j$ collaborate to resolve the winners by disclosing $G^1_j(\alpha_k), \ldots, G^n_j(\alpha_k)$ for $k = 1, \ldots, u$. There is an agent A^*_j such that $(G^*_j)^{(u)}(0) = 0$, which proves its bid is the lowest. The second-price bid, b^{**}_j, is computed by excluding the bid of agent A^*_j such that

$$\Lambda^i_j = \frac{\Lambda^i_j}{(g_1)^{f^*_j(\alpha_i)}}, \quad (6)$$

and recomputing (5). The schedule $\mathbf{S} = \{S^1, \ldots, S^n\}$ is obtained by computing $S^i = S^i \cup \{T_j\}$, where agent A_i is the winner A^*_j.

Payments: As the auction for task T_j ($j = 1, \ldots, m$) completes, the payment to agent A_i ($i = 1, \ldots, n$) is computed as $p(i) = \sum_{j \in S^i} b^{**}_j$.

4. Properties and Complexity

In this section, we study the properties and the complexity of DMW. We provide proof sketches for the theorems characterizing the properties of DMW.

Theorem 4.1. (Truthfulness) The DMW mechanism is truthful.

Proof. (Sketch) DMW preserves the incentives and the allocation policies of MinWork, without altering any of the semantics of MinWork. Since MinWork is truthful [12], therefore DMW is truthful. □

Theorem 4.2. (Privacy) DMW protects the anonymity of the losing agents and the privacy of their bids.

Proof. (Sketch) The privacy of the identities and the bids is protected using a secret sharing scheme. *At least* c agents must collude in order to expose the bids. Additionally, the number of colluding agents necessary to successfully expose bids is inversely proportional to the bid value, *i.e.*, more colluding agents are required to violate the privacy of lower bids. The information cannot be used across mechanism executions as the information is only relevant to a specific execution as dictated by the *scheduling on unrelated machines* model. □

Remark. It is ideal to conceal the winner's identity, the first-price, and the second-price. The disclosure of the winner and the second-price is intrinsic to the distributed scheduling problem. The risk of divulging the winner is diminished by using *pseudonyms* to hide the real identities. The risk of disclosing the second-price is small as the information must be divulged to a third-party to receive payment. The disclosing of the first-price is a deficiency of the auction protocol that poses the greatest risk. The risk is mitigated as the agents cannot leverage the information to improve their positions during mechanism execution as all bids are submitted and committed before revelations. Furthermore, the information cannot assist the agents across mechanism executions since the information is only relevant to a specific execution as assumed in the scheduling model.

Theorem 4.3. (Communication Complexity) The communication complexity of DMW for m tasks and n agents is $\Theta(\, m \times n^2\,)$.

Proof. (Sketch) We assume no explicit broadcast facilities. For each task, we observe that bidding requires $\Theta(\, n(n-1)\,)$ point-to-point message transmissions and n message broadcasts. Interpolation dominates the cost of subsequent phases, and since $\max \mathcal{W} \leq n$, it requires $O(n^2)$ message broadcasts. The overall communication cost is $\Theta(\, m \times n^2\,)$. □

Theorem 4.4. (Computation Complexity) The computation complexity of DMW for m tasks and n agents is $O(\, m \times n^2\,)$.

Proof. (Sketch) We followed the practice outlined in [2] for calculating the computational complexity assuming that the cost of multiplication is equal to the cost of obtaining the multiplicative inverse, which is more costly than both additions and subtractions. We observe that the cost of bidding dominates the cost of all other stages; an agent computes $\Theta(m \times n)$ shares by repeatedly applying Horner's Rule to solve (3). Since $\max \mathcal{W} \leq n$, each share requires $O(n)$ multiplications. Therefore, the overall computational cost is $O(\, m \times n^2\,)$. □

5. Conclusion

We proposed a distributed scheduling mechanism for solving the problem of scheduling on unrelated machines. The agents collaboratively execute a Vickrey auction for each of the tasks to determine the allocation and the payment. The anonymity of the losing agents and the privacy of their bids is protected by using a secret-sharing scheme. We show that our proposed mechanism exhibits the strategic properties inherent to MinWork and thus is truthful. Furthermore, we show that the mechanism preserves the privacy of the losing agents by not revealing their identities or their bids. Lastly, we show that it is polynomial in terms of communication and complexity costs. In future work we are planning to apply similar cryptographic protocol-design techniques to devise secure distributed versions of other centralized mechanisms.

References

[1] F. Brandt and T. Sandholm. On correctness and privacy in distributed mechanisms. In *Proc. of the Agent-Mediated Electronic Commerce Workshop*, 2004.

[2] A. T. Chronopoulos, D. Grosu, and H. Kikuchi. A new efficient polynomial degree resolution protocol and its application to the $(M+1)$-st price private auction. In *Proc. of the 2nd Intl. Conf. on Applied Cryptography and Network Secruity - technical track*, pages 358–367, June 8–11 2004.

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

[4] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proc. of the 21st ACM Symp. on Principles of Distributed Computing*, pages 173–182, July 2002.

[5] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, Aug. 2001.

[6] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proc. of the 6th ACM Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, Sept. 2002.

[7] T. Groves. Incentive in teams. *Econometrica*, 41(4):617–631, 1973.

[8] H. Kikuchi. $(M+1)$st-price auction protocol. In *FC'01: Proc. of the 5th International Conference on Financial Cryptography*, pages 351–363. Springer-Verlag, 2002.

[9] J. C. Mitchell and V. Teague. Autonomous nodes and distributed mechanisms. In *Proc. of the Mext-NSF-JSPS International Symp. on Software Security - Theories and Systems*, pages 58–83, Nov. 2003.

[10] D. Monderer and M. Tennenholtz. Distributed games: From mechanisms to protocols. In *Proc. of the 16th National Conference on Artificial Intelligence*, pages 32–37, July 1999.

[11] C. Ng, D. C. Parkes, and M. Seltzer. Strategyproof computing: Systems infrastructures for self-interested parties. In *1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, June 5–6 2003.

[12] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1):166–196, Apr. 2001.

[13] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, Mass., 1994.

[14] D. C. Parkes and J. Shneidman. Distributed implementations of Vickrey-Clarke-Groves mechanisms. In *Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multi Agent Systems*, pages 261–268, 2004.

[15] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, Englewood Cliff, NJ, 1995.

[16] A. Shamir. How to share a secret. *Communications of the ACM*, pages 612–613, Nov. 1979.

[17] J. Shneidman and D. C. Parkes. Specification faithfulness in networks with rational nodes. In *Proc. 23rd ACM Symp. on Principles of Distributed Computing (PODC'04)*, St. John's, Canada, 2004.

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