THE COMPLEXITY OF RESOURCE ALLOCATION AND PRICE MECHANISMS UNDER BOUNDED RATIONALITY

by

Eric J. Friedman and Shmuel S. Oren

Memorandum No. UCB/ERL/IGCT M92/100

October, 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering

University of California, Berkeley

94720

The Complexity of Resource Allocation and Price Mechanisms Under Bounded Rationality

Eric J. Friedman and Shmuel S. Oren
Department of Industrial Engineering and Operations Research,
University of California at Berkeley,
Berkeley, CA 94720.

(Draft)

October 28, 1992

Abstract

We develop a framework for designing and evaluating the complexity of mechanisms that allocate resources in a distributed setting. This framework is applicable to economics and distributed computing. We discuss several mechanisms and describe the construction of efficient price based mechanisms, which exploit the structure in the problem. For the case of one resource our mechanism is the fastest known distributed algorithm for the problem. For two resources this is the only known method which exploits the structure of the problem and is the most efficient serial or parallel algorithm known. We conjecture that this is also true for price mechanisms for more than two resources, and propose a method for constructing them.

This research has been funded by National Science Foundation Grant IRI-8902813.

Contents

1 Introduction

In this paper we consider the problem of allocating resources among a large group of agents in a distributed setting. We provide a framework for studying such mechanisms, which we use to evaluate several different ones. Our goal is to design efficient mechanisms that can compute reasonable allocations rapidly in a distributed setting.

The design of mechanisms for the allocation of scarce resources among a large group of agents has been a fundamental topic in modern economics [AH60, Hur73, HM85]. This problem is also of great importance in distributed computing, where resources may refer to CPU time, network bandwidth, and data storage. In fact much recent work has involved the application of economic ideas to problems in distributed computing. (See e.g. [FNY88, Hub88, IK88, KS89, San85, San88, She90].)

The formal study of 'efficiency' for different mechanisms was instituted by Reiter and Hurwicz [Hur86b, MR74] and has produced a vast literature on optimal mechanisms. However, most of these analyses consider mechanisms which find optimal or Pareto efficient allocations at equilibrium [Hur77]. The incorporation of dynamic elements has also been attempted in a limited manner [MR87]. However, the time required to actually find a good (or optimal) allocation has not previously been studied. Also, the complexity of a mechanism has been defined as the number of messages required at equilibrium, not the computational effort or total number of messages required overall. We believe that the total time and effort required to compute an allocation should be considered when designing mechanisms, because these directly determine how well they function.

Furthermore, most of the work in this field has implicitly assumed unlimited computational ability by the agents, who are assumed to exactly solve difficult optimization problems instantaneously. This viewpoint is certainly unrealistic and has recently been questioned; typically, agents in an economy are not infinitely wise nor is the designer of the mechanism. Such limitations have been referred to as 'bounded rationality.'[Sim86, RW91]

Several approaches have attempted account for the decision maker's bounded rationality. Simon [Sim86] considers 'satisficing' mechanisms, where mechanisms must perform in an informally defined satisfactory manner. Mount and Reiter[MR90] consider the design of mechanisms when the participants are infinite dimensional generalizations of finite automata.

In this paper we consider a very natural model of 'finite rationality'. We model the participants as 'computers' where each one has finite 'speed of computation.' For our definition of 'computer' we use the recent idea of 'computation over the real numbers' as discussed in [BSS88, Meg83, Blu89]. Our 'computers' can manipulate real numbers and do the following operations in unit time: addition, subtraction, multiplication, division, and comparison. This model of computation is both elegant and useful analytically. It is also, perhaps, a better model of the way modern computers work and numerical analysis. Nonetheless, most of our discussion and results are also valid in standard models of computation, such as Turing machines, with only slight modifications.

We consider the problem of constructing mechanisms which compute allocations that are guaranteed to be good, but not perfect, in a reasonable amount of time. The most important aspect of these mechanisms is that they scale well with the number of agents. That is, even if the number of agents is very large our mechanism still operates in a reasonable amount of time[1].

Previous results on such mechanisms are scarce. The case of a single resource has been well studied, and several efficient algorithms for it exist [IK88, Hoc]. For multiple resources Nemirovsky and Yudin [NY83] provide the framework for constructing such mechanisms, but only explicitly consider the single resource case. In [Fri92] we describe the construction of a mechanism for a generalization of the multi-resource allocation problem.

In the next section we formally define our model and some measures of complexity and efficiency for mechanisms. We then consider quantitity based mechanisms. These are often denoted primal algorithms. We describe two different such mechanisms. These are only discussed briefly as they have been well treated elsewhere [Fri92, NY83]. However, neither of these mechanisms satisfies our definition of efficiency. This is a standard problem with primal algorithms as they do not appear amenable to distributed implementation.

Our main focus is on price-based mechanisms and the construction of efficient mechanisms using prices. These can be interpreted as finite versions of primal-dual algorithms. These mechanisms are naturally implemented in a distributed system, as primal-dual algorithms typically distribute well. For example Arrow and Hurwics use a primal dual method for constructing descentralized mechanisms in [AH60]. Also, by taking advantage of the structure inherent in resource allocation problems, they are very efficient[2]. For example the price mechanism for a single resource is the most efficient distributed algorithm of which we are aware.. For 2 resources it is the only algorithm we know of which exploits the structure of this problem and is the most efficient serial or parallel algorithm known to us. For more than two resources we conjecture that the same is true.

Finally, we note that the mechanisms here can be seen as extensions of a theory of 'global' optimization expounded in [Fri92] which is based on the work of Nemirovsky and

[1]Similar ideas can be found in Marschak [Mar86]. He explicitly considers the running time of a mechanism, but allows agents to use infinite computation.

[2]The formal description of mechanisms reflects the interplay between the theory of mechanisms and that of mathematical programming. In fact many of the mechanisms found in the economics literature have their inception in mathematical programming, and currently many algorithms for mathematical programming are using ideas from economics for their inspiration. Our work is no exception.

Yudin [NY83] and recommended in [Son85]. This should be contrasted with the standary asymptotic results in nonlinear programming. In those results, convergence is only guaranteed in a small neighborhood of the solution. Also, the asymptotic theory relies very strongly on the analytic properties of the functions being optimized. Global results are only based on convexity propeties of the functions. Convergence results are global and guarantee the (approximate) solution of the problem will be found in a specified finite time.

The specific results of this paper is that the global theory can be applied to problems with special structure, in order to reduce the number of computations required to find a solution.

This paper is structured as follows. In section two we define a model of complexity for resource allocation, both with and without a center. Then chapter three constructs two quantity based mechanisms, which are modeled after known methods of optimization. Neither of these mechanisms are efficient, so we are led to the construction of new, price based, mechanisms in section four. In the first part of this section we construct a price mechanism for a single resource. Much of this part is meant to be pedagogical, but none the less this mechanism is a significant improvement over known algorithms for the distributed allocation of a single resource. In the second part we construct a price based mechanism for two resources and propose a similar construction for any number of resources. The proofs for this part are quite involved, so we provide a sketch in the appendix. Section five describes the extension of our mechanisms to the important case of externalities and the final section contains our conclusions.

2 Model

We consider the problem of allocating r resources among n agents. The basic data of an economy is the environment or set of utility functions for the agents. This is denoted by $U = (U_1, U_2, \ldots, U_n) \in \mathbf{U}$, where $\mathbf{U}$ is the set of possible environments. We will assume that $\mathbf{U} = \mathbf{U}_1 \times \mathbf{U}_2 \times \cdots \times \mathbf{U}_n$ where each $\mathbf{U}_i$ is the set of all C^2 nondecreasing concave (utility) functions $U_i : \Re^r \rightarrow \Re^+$ with $|\frac{\partial U_i}{\partial x_i^j}| \leq 1$, and $U_i(0) = 0$. Note that as U_i is concave, the condition that $|\frac{\partial U_i}{\partial x_i^j}| \leq 1$ is quite mild and is essentially a normalization.

As is typical, we assume that U_i is private information; it is only known to agent i. However, we assume a much stronger form of privacy than is typical. We assume that, in some sense, utility function i is not known to agent i. The basic idea is that an agent consists of two parts: 'heart' and 'head.' The head does all the thinking, and the heart does the evaluating. Thus the head is a computer and the heart is a utility function oracle. Given a bundle of goods x_i the head can ask the heart for $U_i(x_i)$ and any local information about $U_i(x_i)$, such as it's derivatives. However, each query requires a constant amount of time. This procedure is an attempt to model the information that an agent has about her utility function and to avoid the 'smuggling' of information and computation that can occur if the agent is 'aware ' of her entire utility function. This also seems realistic as an agent typically can only know how much she likes a specific bundle and what she would like more of.

For ease of exposition we will assume that there is 1 unit of each resource. The set of feasible allocations is

$$\mathrm{F} = \{x = (x_1, \ldots, x_n)\ |x_i \in \Re^r,\ \sum_{i=1}^{n} x_i^j = 1\}.$$

These restrictions are easily relaxed at the cost of additional notation.

We will consider both central mechanisms and distributed mechanisms. Central mechanism are very common in economics. For example the standard Walrasion tattonement process consists of a center shouting prices to a large group of agents. Central mechanisms can also arise in distributed computing, when many agents of are sharing a single large resource. However, center-less mechanisms are often more relevant for distributed computing. This models a large collection of processors (such as workstations or personal computers) interconnected on a large network. We will see that any of the central mechanisms we design are easily implemented in a distributed setting with a negligible loss of efficiency.

2.1 Central Mechanisms

We will assume that there are many agents interacting with a computaionally powerful center. Thus we assume that the 'center' has computing speed proportional to the number of agents, and each agent has unit speed of computing. The center communicates with the agents by broadcasting a single real number to all the agents simultaneously. The agents can respond by each one sending a single real number to the center. (see figure 1.)

We will define a mechanism as a vector of computer programs (or algorithms)

$$m = (m_1, \ldots, m_n, m_c) \in M_1 \times \cdots \times M_n \times M_c$$

where each m_i represents a program that computer i follows. Let $x(m, U) = (x(m_1, U_1), \ldots, x(m_n, U_n))$ be the output of mechanism m in environment U. (We will only consider the case of mechanisms which have feasible outcomes $x(m, U) \in F$.)

Define the error of mechanism m in environment U to be

$$\epsilon(m, U) = |U(x(m, U)) - U(x^*(U), U)|$$

where

$$x^*(U) = \text{argmax}_{x \in F} U(x)$$

So $\epsilon(m, U)$ is the error of m on U. Now define the error of a mechanism to be

$$\epsilon(m) = \max_{U \in \mathbf{U}} \epsilon(m, U).$$

We are interested in parametrized families of mechanisms, $\{m^{\epsilon,n,r}\}$, where $m^{\epsilon,n,r}$ is an mechanism that for n agents and r resources has $\epsilon(m^{\epsilon,n,r}) \leq \epsilon$ for all $U \in \mathbf{U}$.

Our mechanism must perform 3 different actions: computation, communication, and utility function evaluation. We represent the time required for each of these operations symbolically. Let X be the amount of time required for a single (real number) computation by an agent and X/n for the center, I be the time for the communication of a number, and U the time required for a single query of a utility function. Note that the agents and the center can each perform a single action simultaneously.

Now denoting the complexity of a mechanism to be the time required by that mechanism to compute an allocation. We will compare the asymptotic behavior of complexities by considering this time

$$T(m^{\epsilon,n,r}) = \mathcal{O}(G(\epsilon, n, r))$$

where $G(\cdot)$ is a (symbolic) function containing X, I, U's such that

$$T(m^{\epsilon,n,r}) = \kappa G(\epsilon, n, r)$$

for some constant κ and all ϵ, n, r.

Thus we can compare different mechanisms by comparing their complexities. As in standard complexity theory of parallel computing [Lei91], an algorithm will be deemed 'efficient' if it is bounded by a polynomial in r, $\log n$, and $\log(1/\epsilon)$. This models the assumption that there are a reasonable number of resources, a very large number of agents, and that the mechanism is at least (globally) linearly convergent.

2.2 Distributed Mechanisms

As we will see later, distributed mechanisms can easily duplicate central mechanisms with only a slight loss of efficiency. In a distributed mechanism there is no center. We imagine that each agent can do one computation in time X. We also assume that the agents are connected on a network. Thus there is a graph $G = (V, E)$ where each vertex corresponds to an agent $V = \{1, 2, \ldots, n\}$ and each edge connects two agents $(i, j) \in E$. However, we require that the graph not have too many edges, thus preventing everyone from talking to everyone else, which would be unrealistic. A useful (and non-restrictive) assumption is that each agent is connected to a small number of other agents independent of n.

For concreteness we will assume that the agents are connected via a Butterly network[3]. (See figure 2.) In a butterfly network each agent is connected to four other agents in an array that (somewhat) resembles a butterfly. On a Butterfly network many distributed comutations can be performed rapidly. For example sums of numbers and matrix operations can be performed in $\mathcal{O}(\log k)$ time, where k is the number of real numbers involved in the computation [Lei91].

Now note that any central mechanism can be implemented as a distributed mechanism by simply designating some agent to act as the center. However, typically this would increase the complexity by a factor of $\mathcal{O}(n)$ as the agent computes more slowly than the center could thus making any efficient central mechanism into an inneficient distributed one.

However, this can be avoided by exploiting the power of distributed computations by the network. All of the mechanisms which we describe can be implemented in a distributed manner. In all of these mechanisms the complexity is increased by a factor of $\log n$ in the number computations on information exchanges. This does not effect the efficiency of any of the mechanisms.

[3]Actually any expander graph would suffice. See [Lei91].

3 Quantity Mechanisms

In this section we describe two quantity based, or direct, mechanisms. These are mechanisms which operate on the allocation space. These mechanisms have the nice property that (after a period of 'initialization') the intermediate allocation is always feasible. Therefore if the mechanism must be terminated before completion, it still supplies a feasible (and reasonably good) allocation.

Neither of these mechanisms, however, satisfy our definition of efficiency. The first one, which we call the Ellipsoid mechanism, fails as it does not distribute well, and uses very little of the agents' computing power. The second, Mirror Descent, fails because of it's poor dependence on ϵ; it converges sublinearly.

We present these mechanisms mainly for comparison, and history, as these appear to be the only mechanisms with finite complexity in the literature. Both are based on the work of Nemirovsky and Yudin [NY83], and are presented here with slight modifications and simplifications enabled by the structure of the resource allocation problem.

3.1 The Ellipsoid Mechanism

Like the ellipsoid method[4] the ellipsoid mechanism constructs a sequence of ellipsoids of decreasing volume that converge on to the optimal solution.

In this mechanism the center constructs an ellipsoid containing the optimal solution and a possible allocation which is at the center of the ellipsoid. If this allocation is infeasible then the center computes a seperating hyperplane for the feasibility constraints at this point. If this point is feasible then the center communicates this allocation to the agents. The agents then reply with their utility value and gradient at the point, reflecting their satisfaction and local tradeoffs at that point. The center then computes a new smaller ellipsoid that is guaranteed to contain all allocations better than the current one. The process is repeated a sufficient number of times to guarantee that the final allocation is ϵ-accurate. This is formalized as follows.

Center's Program — $m_c^{\epsilon,n,r}(ellipsoid)$

[4]The ellipsoid method was developed by Nemirovsky and Yudin [NY83] based on an idea of Levin [Lev65]. It is most well known from its use by Khachian [Kha79] to prove the polynomiality of linear programming. However, its original purpose was for convex programming. While it is useful theoretically for LP it does not seem to be of practical value; however, it may actually be useful for convex programming.[EK83]

```
1. Let $x = (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ and
       $$B = Diag(\frac{1}{2\sqrt{rn}}, \frac{1}{2\sqrt{rn}}, \ldots, \frac{1}{2\sqrt{rn}}).$$
2. Let $N = \lceil 4(rn)^2 \log \frac{rn}{\epsilon} \rceil$, $u_{out} = \infty$, and $x_{out} = 0$.
3. Repeat $N$ times.
   (a) If $x \in F$
       i. then
          A. For all i: transmit $x_i$ to agent $i$.
          B. For all i: receive $c_i = \nabla U_i(x_i)$ and $u_i = U(x_i)$ from agent $i$.
          C. Let $u = \sum_{i=1}^{n} u_i$.
          D. If $u \leq u_{out}$ then $u_{out} = u$ and $x_{out} = x$.
       ii. else choose any violated inequality in $F$ (this is of the form $\hat{c}^t x \leq b$) and let
           $c = \hat{c}$.
   (b) Calculate
       $$B' = \frac{n^2}{n^2-1}[B - \frac{2}{n+1} \cdot \frac{(Bc)(Bc)^t}{c^t Bc}]$$
       $$x' = x - \frac{1}{n+1} \cdot \frac{Bc}{\sqrt{c^t Bc}}.$$
   (c) Let $B = B'$ and $x = x'$.
4. Output $x_{out}$ and $v_{out}$.
```

Agent's Program — $m_i^{\epsilon,n,r}(ellipsoid)$

```
1. Receive $x_i$.
2. Transmit $\nabla U_i(x_i)$ and $U_i(x_i)$.
3. Repeat.
```

It is apparent that the agents do no computation in this mechanism. Their only use is to provide function and gradient values. The center is constantly recomputing the current ellipsoid. This is done a total of $4(rn)^2 \log(rn/\epsilon)$ times, thus providing a impassable barrier to reducing the complexity significantly. Whereas we could have reduced the complexity somewhat by involving the agents in the computation of the new ellipsoid, there is no way to reduce the dependence of the complexity from n to $\log n$. Thus, even with these modifications, the ellipsoid mechanism is not efficient.

Theorem 1 *The Ellipsoid mechanism*[5] *has complexity*

$$T(m^{\epsilon,n,r}(ellipsoid)) = \mathcal{O}(r^3 n^2 \log(rn/\epsilon)[X(1 + \frac{\log 1/\epsilon}{rn}) + I + U])$$

Proof: The proof is ommitted for brevity. It is a straightforward application of the proof in [NY83] where the convergence of the ellipsoid method is shown. □

3.2 The Mirror Descent Mechanism

The mirror descent mechanism is based Nemirovsky and Yudin's method of mirror descent[6]. Mirror descent is the natural generalization of gradient descent to general metric spaces.

[5]Note that the above mechanism requires the computation of a square root which is not one of our elementary operations. This can be computed approximately by a binary search, and the update formula for M', m' slightly modified to accept this approximation. This is the reason for the $(1 + \frac{\log rn/\epsilon}{(rn)^2})$ term in the complexity. We have ommitted this as the modification is straightforward, but involved. The interested reader should see [PS82][pp.182-5].

[6]The method of mirror descent mechanism is described in [NY83]. The construction and analysis of the algorithm is quite sophisticated and intricate. We refer the interested reader to Nemirovsky and Yudin's book [NY83], and provide only a sketch here. First we describe the motivation behind the Mirror Descent algorithm, and then describe its specialization to resouce allocation.

Consider the convex program:

$$\min f(x) \text{ s.t. } x \in G$$

where $f(x)$ is a convex function and G is a convex region of the finite dimensional metric space (Readers unfamiliar with the general theory of metric spaces should consult [DS66].) $E = \Re^n_1$ with metric $||\cdot||_p$ where

$$||x||_p = (\sum_i |x_i|^p)^{1/p}$$

and note that the dual of E is $E^* = \Re^n_\infty$ with metric $||\cdot||_\infty$.

This is necessary for resource allocation as the complexity of gradient descent depends both on the size of the feasible region and the Lipschitz constant of the utility function both of which depend strongly on the chosen metric.

Gradient methods have been commonly used for resource allocation and optimization in general. The idea that convergence rates depend on the metric chosen is also well known. The use of variable metrics was used by Oren and Luenberger [OL74] to improve convergence for certain nonlinear programs.

For resource allocation we choose the metric

$$\|x\|_1 = \sum_{i,j} |x_i^j|$$

for which the diameter of the feasible region is contained in a ball of radius r and the

---

Now define a sublinear function on E^* by

$$V(\phi) = \begin{cases} \frac{1}{2}\|\phi\|_\tau^2 & \|\phi\|_\tau \leq 1 \\ \|\phi\|_\tau - \frac{1}{2} & \|\phi\|_\tau > 1 \end{cases}$$

where $\tau = (\log(n)/16e)$. and consider the continuous trajectory in the dual space defined by

$$\frac{d\phi(t)}{dt} = -\nabla U(V'(\phi(t)))$$

where $\phi(t) \in E^*$. Since $V(\phi)$ is a function on E^* then $V'(\phi)$ is an element of E. If we let

$$x(t) = V'(\phi(t)).$$

and choose

$$V_*(\phi) = V(\phi) - < \phi | x^* >$$

where x_* is the optimal solution and $< \phi | x >$ is the natural pairing of E and E^*. We can show that $V_*(\phi)$ acts as a Lyapunov function since

$$\frac{dV_*(\phi(t))}{dt} = < -\nabla U'(V'(\phi(t))) | V'(\phi(t)) - x^* > = < -\nabla U'(V'(\phi(t))) | x(t)) - x^* >$$

$$\leq U(x^*) - U(x(t)) \leq 0$$

and

$$\frac{dV_*(\phi(t))}{dt} = 0$$

when $x(t) = x^*$. Since $V_*(\phi)$ is bounded below on E^* the trajectory is guaranteed to converge to the optimal solution.

Now the Mirror Descent algorithm is the discretization the above argument which approximates the trajectory of the differential equation.

Lipschitz constant of U is less than 1. Then the main action of this algorithm occurs in the dual space E^*. At each iteration the agents provide $\nabla U_i(x_i)$ and, as this is an element of E^*, take a small step in this direction. The center's role is to compute the step size ρ. This computation only requires that the center compute the norms of different objects which are sent to him from the agents.

```
Center's Program — $m_c^{\epsilon,n,r}(descent)$

1. Set $\phi = 0$, $a = +\infty$, $b = 0$, and $\phi_i = 0$.
2. Continue
   (a) Compute $(\|\phi\|_\tau)^{\tau+1}$.
   (b) For all $i$: transmit $(\|\phi\|_\tau)^{\tau+1}$ to agent $i$.
   (c) For all $i$: receive $x_i'$ from agent $i$.
   (d) For all $i$: transmit $\|x^j\|_1$ for all $j$ to agent $i$.
   (e) For all $i$: receive $x_i$, $u_i = U_i(x)$, and $g_i = \nabla U_i(x_i)$ from agent $i$.
   (f) Compute $l = \|g\|_\infty$, $u = U(x)$, and $a = \min[a, u]$.
   (g) Compute $\|x\|_1$.
   (h) For all $i$: transmit $\|x\|_1$, and $l$ to agent $i$.
   (i) If $g = 0$ then goto 3.
   (j) Let $\hat{\delta} = \epsilon + (u - a)/l$.
   (k) For all $i$: receive $\hat{\zeta}_i$ from agent $i$.
   (l) Compute $\delta = \hat{\delta} / \|\hat{\zeta}\|_\infty$.
   (m) Compute $\rho = \delta/(2\tau)$ and $\gamma = \delta\rho/2$.
   (n) For all $i$: transmit $\rho$ to agent $i$.
   (o) Let $b = b + \gamma$.
   (p) For all $i$: receive $\phi_i$, $\phi_i^\tau$, and $\phi_i^{\tau-1}$ from agent $i$.
   (q) Compute $\hat{V} = V(\phi) - \|\phi\|_\infty$.
   (r) If $\hat{V} > -b$ then goto 3, otherwise increment $i$ and goto 2.
3. Output $a$.
```

Agent's Program — $m_i^{\epsilon,n,r}(\textit{descent})$

```
1. Set $\phi_i = 0$.
2. Repeat.
    (a) Receive $(\|\phi\|_r)^{r+1}$ from the center.
    (b) Compute $\hat{x}_i = (\|\phi\|_r)^{r+1}\phi_i$.
    (c) Transmit $\hat{x}_i$ to the center.
    (d) Receive $\|\hat{x}^j\|_1$ for all $j$ from the center.
    (e) Compute $x_i^r = \frac{\max(0,\hat{x}_i^j)}{\|\hat{x}^j\|_1}$ if $\|\hat{x}^j\|_1 > 1$ otherwise set $x_i^r = \max(0, \hat{x}_i^j)$.
    (f) Let $\mu_i^j = +1$ if $\hat{x}_i^j > x_i^j$, otherwise let $\mu_i^j = -1$.
    (g) Transmit $U_i(x_i)$ and $g_i = \nabla U_i(x_i)$.
    (h) Receive $l$.
    (i) Let $\xi_i = g_i/l$.
    (j) Let $\hat{\zeta}_i = \xi_i + \mu(x_i)$.
    (k) Transmit $\hat{\zeta}_i$.
    (l) Receive $\left\|\hat{\zeta}_i\right\|_\infty$.
    (m) Let $\zeta_i = \hat{\zeta}_i / \left\|\hat{\zeta}\right\|_\infty$.
    (n) Transmit $\zeta_i$.
    (o) Receive $\rho$.
    (p) Let $\phi_i = \phi_i - \rho\zeta_i$ and
    (q) Transmit $\phi_i$.
    (r) Go to 2.
```

Theorem 2 *The Mirror Descent mechanism*[7] *has complexity*

$$T(m^{\epsilon,n,r}(md)) = \mathcal{O}(r^3 \frac{\log(rn)}{\epsilon^2}[X(1 + \frac{\log 1/\epsilon}{rn}) + I + U])$$

Proof: The mechanism is a straightforward application of Nemirovsky and Yudin's method of mirror descent [NY83]. The complexity is computed by noting that the algorithm must terminate in $\mathcal{O}(r^2 \frac{\log(rn)}{\epsilon^2})$ iterations by the theorem in [NY83] and the following facts. The radius of F in the one norm is r and the Lipschitz constant of $U(\cdot)$ is less than 1 in the infinity norm as we assume that $U_i(\cdot)$ has Lipschitz constant less than 1. Noting that each iteration requires $\mathcal{O}(r)$ queries to the oracles, and a similar number of comunnication steps, completes the analysis. □

The number of steps is $\mathcal{O}(\frac{\log rn}{\epsilon^2})$ and this cannot be improved. Thus the poor convergence (sublinear) of this mechanism cannot be avoided[8].

[7]Note that as in the ellipsoid mechanism, we are required to take a root of a real number. This occurs in steps a and p. Once again we may use a binary search to compute this to enough accuracy that the algorithm will not be significantly effected.

[8]Nemirovsky and Yudin have computed lower bounds for the number of iterations required for optimizing general convex functions [NY83]. It is interesting to note that Mirror Descent is optimal (of least complexity) when n is large, and the ellipsoid method is nearly optimal for high accuracy computations, in their framework. Thus the only reason we are able to improve on these mechanisms is the special structure inherent in resource allocation problems.

4 Price Mechanisms

The idea of a price mechanism is based on the idea of a Walrasian tatonnement process. This process (described in detail in [Var78]) is based on the idea of a center announcing a sequence of prices, and agents picking a consumption bundle that maximizes $U_i(x_i) - p \cdot x_i$. This process continues until a feasible allocation is reached.

The importance of this process is demonstrated by Hurwicz [Hur86a]. He shows that this process defines an optimal mechanism, in the sense of having the minimal message space of any static mechanism which implements a pareto optimal allocation. However, this mechanism is not stable [Sca60]. A modification of it based on the 'Global Newton Method' [Sma76] is stable, but requires a larger message space. These mechanisms assume agents can provide their exact optimal consumption bundle for a given set of prices. Thus agents are able to instantly compute the optimum of a difficult optimization problem, while any such computation requires an arbitrarily large number of computational steps. The issue of accuracy and computation ability of agents in this situation has been neglected in the literature.

In this section we show how to construct finite 'computational' mechanisms based on the Walrasian mechanism. These mechanisms are very efficient and naturally distributed.

4.1 One Resource

In this section we present a price based algorithm for allocating a single resource. The algorithm we present is not the simplest possible, but it is the one most amenable to generalization. Most of the ideas in its construction are extendable to the multi-resource problem. Thus, our goal here is to provide a basis for constructing other price based mechanisms.

This mechanism operates in two stages. In the first stage the center announces a price and the agents compute a resource utilization that approximately maximizes $U_i(x_i) - px_i$ to a given accuracy. Using this information the center computes a new price and the process continues until the center has found a price $\hat{p}$ that approximately minimizes $U(x) - p\sum_i x_i$.

However, the current allocation may not be feasible for two reasons. First, even if the price was the correct price p^*, the resulting allocation might not be feasible if the agent's utility function was flat at this price, and his marginal utility constant, thus allowing for a wide choice of possible consumption choices all with the same net utility. (see figure 3.) Another problem is that the price $\hat{p}$ may be arbitrarily far from p^* due to inter-agent effects.

This problem is remedied in stage 2. Basically, the center asks the agents for their largest and smallest consumption choices that are reasonably good, given the price $\hat{p}$, which the agents compute approximately. By noting that a feasible allocation must occur for some

x with the consumption by each agent in the interval between these two points, we see that some convex combination of these points must give a feasible allocation. This combination $\hat{x}$ is then used as the allocation. While this allocation may be arbitrarily different from the optimal allocation ($\|\hat{x} - x^*\|$ may be quite large) it is still an accurate one in the sense of $|U(x^*) - U(\hat{x})|$. convex combination of two other good allocations and convex combinations preserve the accuracy of solutions.

Note the interplay between the agents and the center. Essentially the center is computing a good approximate allocation by asking question of the agents, who are computing approximate answers. Thus there are two levels of computation and two levels of approximation.

```
Center's Program — $m_c^{\epsilon,n,1}(price)$
Stage 1:
  1. Let $p_l = 0$, $p_h = 1$, and $\hat{\epsilon} = \epsilon/3n$.
  2. Repeat $N_c = \lceil \log(1/\hat{\epsilon}) \rceil$ times.
     (a) Let $p_m = (p_l + p_h)/2$
     (b) For all $i$: transmit $p_m + \hat{\epsilon}/2$ to agent $i$.
     (c) For all $i$: receive $l_i^+ = L_i(x_i^+, p_m + \hat{\epsilon}/2)$ from agent $i$.
     (d) For all $i$: transmit $p_m - \hat{\epsilon}/2$ to agent $i$.
     (e) For all $i$: receive $l_i^- = L_i(x_i^-, p_m + \hat{\epsilon}/2)$ from agent $i$.
     (f) Compute $l^\pm = \sum_i l_i^\pm$.
     (g) If $|l^+ - l^-| \leq \epsilon$ goto 'Stage 2'.
     (h) Else if $l^+ > l^-$ let $p_h = p_m$.
     (i) Else let $p_l = p_m$.
  3. Continue
  4. Let $\hat{p} = p_m$.
Stage 2:
  1. For all $i$: transmit $\hat{p}$ to agent $i$.
  2. For all $i$: receive $x_i^\pm$ from agent $i$.
```

```
3. Compute $s^{\pm} = \sum_i x_i^{\pm}$.
4. Let $\alpha = (s^+ - 1)/(s^+ - s^-)$.
5. For all $i$: transmit $\alpha$ to agent $i$.
```

Agent's Program — $m_i^{\epsilon,n,1}(price)$

Stage 1:

```
1. Receive $p$.
2. Let $x_l = 0$ and $x_h = 1$.
3. Let $\epsilon' = \hat{\epsilon}^2$.
4. Repeat $N_a = \lceil \log(1/\epsilon') \rceil$ times.
    (a) Let $x_m = (x_l + x_h)/2$
    (b) Let $v_m = U_i'(x_m)$.
    (c) If $v_m < p$ let $x_l = x_m$.
    (d) Else let $x_h = x_m$.
5. Continue
6. Transmit $L_i(x_m, p)$ to the center.
7. Go to 1.
```

Stage 2:

```
1. Receive $\hat{p}$ from center.
2. Do for $w = \pm 1$.
    (a) Let $x_l = 0$ and $x_h = 1$.
    (b) Set $\hat{l} = L_i(\hat{x}_i, \hat{p})$.
    (c) Repeat $\lceil \log(\hat{\epsilon}) \rceil$ times.
        i. Let $x_m = (x_l + x_h)/2$
        ii. Let $l_m = L_i(x_m, \hat{p})$ and $v_m = U_i'(x_m)$.
```

iii. If $|\hat{l}_m - l_m| < \hat{\epsilon}$ let $x_l = x_m$ if $w = +1$ otherwise let $x_h = x_m$.

iv. Else if $v_m < p$ then let $x_l = x_m$.

v. Else let $x_h = x_m$.

3. Continue

4. Transmit $x^{\pm}$ to center.

5. Receive α from center.

6. Compute $x_i = \alpha x_i^+ + (1-\alpha)x_i^-$. (This is the allocation to agent i).

Theorem 3 *The 1 resource price mechanism $m^{\epsilon,n,1}(price)$ has complexity*

$$T(m^{\epsilon,n,1}) = \mathcal{O}((\log \frac{n}{\epsilon})^2[X + I + U])$$

Proof: It is easy to see that stage 1 dominates stage 2. During stage 1 the center performs $\log(1/\hat{\epsilon})$ iterations, and each iteration requires the computation of a sum, a few simple computations, and two calls to the agents to compute their own optimization problem. During these calls each agent performs $\log(1/\epsilon')$ iterations, and each iteration requires one call to the oracle and several basic computations. This shows that the complexity is as stated.

To prove that the algorithm computes a correct solution consider the Lagrangian

$$L(x,p) = \sum U_i(x_i) - p \cdot (\sum x_i - 1)$$

and the function

$$F(p) = \max_x L(x,p).$$

By duality theory of convex programming [Roc70] we know that $F(p)$ is a convex function and if $p^* = \min_p F(p)$ then

$$x^* = \text{argmax}_x L(x, p^*).$$

Define $x(p) = \text{argmax}_x L(x,p)$ so $x^* = x(p^*)$. The Lagrangian is separable in x and can be written as $L = \sum_i L_i$ where

$$L_i(x_i, p) = U_i(x_i) + p(x_i - 1/n)$$

so $x_i(p) = \text{argmax}_{x_i} L(x_i, p)$.

Note that the Lipschitz constant of $F(p)$ is

$$\Lambda(F) = |\frac{\partial L(x,p)}{\partial p} + \sum_i \frac{\partial L(x,p)}{\partial x_i}\frac{\partial x_i(p)}{\partial p}| = |(\sum_i x_i - 1) + 0| \leq n.$$

Also note that

$$\frac{dF(p)}{dp} = \frac{\partial L(x,p)}{\partial p}|_{x=x(p)}$$

by the envelope theorem.

Now we prove the theorem through two lemmas.

Lemma 1 *Stage 1 computes a $\hat{p}$ such that*

$$|F(\hat{p}) - F(p^*)| \leq \hat{\epsilon}$$

Proof: Notice that the agents simply perform a binary search for the x_i that satisfies $U_i'(x_i) = p$, which is the condition for optimality. Thus the x_i that the agent computes, which we denote $\hat{x}(p)$ is within ϵ' of the optimal $x_i(p)$ and this implies that $U_i(\hat{x}_i(p))$ is within ϵ' of $U_i(x_i^*(p))$.

The center is performing binary search on $F(p)$ using solutions that are guaranteed to be accurate to ϵ' in each coordinate and thus $n\epsilon'$-accurate in total using approximate gradients. Note that as we only need the sign of the derivative this inaccuracy is not important except when the computed derivative is close to zero. Thus, if the loop terminates normally then we know that this has not occurred and the solution must be accurate to $\hat{\epsilon}$. However, if at some iteration the center finds a derivative that is close to zero it immediately halts. This is correct since for this to occur the derivative at this point must be less than $\hat{\epsilon}$. This implies that either p_m is within $\hat{\epsilon}$ of p^* or that all three, p_m and $p^{\pm}$ are approximately equal. This implies that $F(p_m)$ is within $\hat{\epsilon}$ of the optimal solution no matter how far p_m is from p^*. For example assume that $p^* > p_m + \hat{\epsilon}$ then

$$F(p_m + \hat{\epsilon}) \leq (1 - \hat{\epsilon})F(p_m) + \hat{\epsilon}F(p^*)$$

by convexity . Therefore

$$F(p_m + \hat{\epsilon}) - F(p_m) \leq \hat{\epsilon}(F(p^*) - F(p_m))$$

and since $|F(p_m + \hat{\epsilon}) - F(p_m)| \leq \epsilon'$ we see that

$$0 \geq F(p) - F(p_m) \geq -\epsilon'/eh \geq -\epsilon$$

◇

Lemma 2 *Stage 2 computes an ϵ-accurate, feasible solution.*

Proof: In stage two the agents compute an approximation of x_i^+ and x_i^- which are solutions of

$$x_i^{\pm} = \text{argmax}_{x_i}(\pm x_i) \text{ s.t. } L_i(x_i, \hat{p}) > L_i(\hat{x}_i, \hat{p}) - \hat{\epsilon}$$

that is accurate to $\hat{\epsilon}$. Now if we can show that $s^+ \geq 1$ and $s^- \leq 1$ then $0 \leq \alpha \leq 1$ and the allocation x_i is a convex combination of $x_i^{\pm}$. Now as $|L(x^{\pm}, \hat{p}) - F(\hat{p})| \leq 2n\hat{\epsilon}$ we see that $|L(x, \hat{p}) - F(\hat{p})| \leq 2n\hat{\epsilon}$ as $L(x, p)$ is concave in x and thus $L(x, \hat{p}) \geq \min L(x^{\pm}, \hat{p})$ by concavity.

From this we immediately see that

$$|L(x, \hat{p}) - F(p^*)| = |L(x, \hat{p}) - L(x^*, p^*)| \leq 3n\hat{\epsilon}$$

and as both x and x^* are feasible this implies that

$$|U(x) - U(x^*)| \leq 3n\hat{\epsilon} \leq \epsilon$$

thus showing that the solution is sufficiently accurate.

Now we must show that $s^+ \geq 1$ and $s^- \leq 1$. If the center in stage one never saw a small derivative then $|\hat{p} - p^*| \leq \hat{\epsilon}$ this implies that

$$|L(x^*, p^*) - F(x^*, \hat{p})| \leq n\hat{\epsilon}$$

as the Lipschitz constant of $L(x, p)$ is less than n. Now as $L(\hat{x}, \hat{p}) \leq L(x(\hat{p}), \hat{p})$ then this implies that x_i^* is a $\hat{\epsilon}$ accurate solution to the agent's stage 2 problem. Thus $x_i^- \leq x_i^* \leq x_i^+$ implying that $s^+ \geq 1$ and $s^- \leq 1$.

However, if the center terminated stage 1 due to a small derivative then we know that $|\frac{\partial F(\hat{p})}{\partial p}| \leq \hat{\epsilon}$, but we also know that

$$\frac{\partial F(\hat{p})}{\partial p} = s(\hat{p}) - 1.$$

Thus we see that $|s(\hat{p})| \leq \hat{\epsilon}$ where $s(\hat{p}) = \sum_i x_i(\hat{p})$. Now if we consider $x' = x(\hat{p})/s(\hat{p})$ we see that $\sum_i x'_i = 1$ and $|x'_i - x_i(\hat{p})| \leq 2\hat{\epsilon}$. Thus $|L(x', \hat{p}) - L(\hat{x}, \hat{p})| \leq 2n\hat{\epsilon}$ for $\hat{\epsilon} \leq 1/2$, so $x_i^- \leq x'_i \leq x_i^+$, completing the proof of the lemma.$\diamond$

Combining the two lemmas completes the proof of the theorem. $\square$

4.2 Two or More Resources

The basic idea for the construction of a price mechanism for two or more resources follows the basic ideas of the mechanism for a single resource case. Stage 1 is the same as previously with two differences. In stage 1 the center computes the derivative using finite differences of two nearby prices. In higher dimensions this idea must be used to compute an approximation of the gradient. This is much more difficult and we must settle for a randomized method of computing an approximation that has only a small chance of failure [NY83]. The second difference is that we must replace the simple bisection algorithm with a ellipsoidal algorithm for convex programming. This is the same algorithm used for the ellipsoid mechanism, but in this case it operates on the price space, which is much smaller than the entire space which is necessary in that mechanism.

The construction of stage 2 is much more complicated than in the simple case. In one dimension it is straightforword to construct two solutions who's convex hull contain a feasible solution. In higher dimensions this is much more difficult. For the case of two resources we have constructed a method which solves this problem. We believe that a generalization of the method should work for an arbitrary number of resources.

In the following discussion we describe the construction of a price based mechanism for two or more resources. As the proofs for stage 1 are not any simpler for two resources than for the general case, we will give them for the case of an arbitrary number of resources. However for stage 2 we specialize to the two resources case and only briefly comment on a possible extension to cases with more resources at the end.

In stage 1 the mechanism computes a $\hat{p}$ that is $\hat{\epsilon}$ accurate for the minimum of $F(p)$ using $\hat{\epsilon}^3$-approximate solutions by the agents to evaluate $F(p)$.

Stage 2 iteratively computes a set of solutions to

$$\max c^t x \text{ s.t. } |L(x,\hat{p}) - L(\hat{x},\hat{p})| \leq \hat{\epsilon}$$

who's convex hull contain a feasible solution (approximately). This is done by repeatedly constructing simplices which either contain a feasible solution or restrict the solution by pushing a face of the simplex up towards the solution. (See figure 4.) This is continued until either a feasible solution is found (by being contained in the convex hull of a simplex) or a face of the simplex is close enough to a feasible solution that we can just 'round' the solution to feasibility with only a small loss in accuracy.

This idea seems to generalize to the problem with an arbitrary number of resources. Again in this case we can use simplices to confine a feasible solution to be 'trapped' by a face of the simplex. Unfortunately this does not gurantee that a straightforward rounding will work. However, we believe that by using several such simplices simultaneously we can successfully 'trap' the feasible solution, to allow for a rounding step to succeed.

As the mechanism for two or more resources is significantly more involved than that for one resource, we will describe the programs more descriptively and less formally than in the previous sections.

Center's Program — $m_c^{\epsilon,n,r}(price)$

Stage 1:

1. Construct the ellipsoid $x = (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ and $B = Diag(\frac{2}{\sqrt{r}}, \frac{2}{\sqrt{r}}, \ldots, \frac{2}{\sqrt{r}})$
2. Let $\hat{\epsilon} = \epsilon/4$.
3. Repeat $N_c = \lceil 4r^2 \log(1/\hat{\epsilon}) \rceil$ times.
   (a) Construct an approximate gradient at p by choosing $\tilde{p}$ at random from the box $K = \{p \mid \|p - \tilde{p}\|_1 \leq 3T\}$ where $T = \frac{\nu r}{2} \left\{ \frac{2n^2 r}{T_0} + \frac{2n\rho}{d} \right\}^{-1}$ and $\rho = \frac{\nu r}{2}$ Now let
   $$c = \sum_{i=1}^{r} \frac{f(\hat{x} + \tau e_i) - f(\hat{x})}{\tau} e_i$$
   where e_i is the unit vector in the i'th direction, $\tau = \frac{\alpha T T_0 \rho}{3n^2\sqrt{n}}$ and $\alpha = \delta/(r^2 \log(n/\hat{\epsilon}))$.
   (b) Compute a new p and B using a finite precision variant of the ellipsoid method.
4. Continue

Stage 2: (For $r = 2$.)

1. Let $c_h = (1,0)^\dagger$, $c_l = (-1,0)^\dagger$
2. For all i: transmit c_h and c_l to the agents.
3. For all i: receive s_h and s_l from the agents, where $s = \sum_{i=1}^{n} x_i \in \Re^2$.
4. Compute v such that $v^\dagger(s_l - s_h) = 0$ and $(0,1) \cdot v = \vec{1}$.
5. Let $v = v / \|v\|_2$.
6. If $v^\dagger s_l < \hat{\epsilon}$ then $c_m = (0,1)^\dagger$.
7. Else if $v^\dagger s_l > \hat{\epsilon}$ then $c_m = (0,-1)^\dagger$.

```
8. Else Goto ROUND.
9. Let $d = c_m$. (This will define the outward direction.)
10. Repeat $16\log(1/\hat{\epsilon})$ times.
    (a) For all $i$: transmit $c_m$ to all agents.
    (b) For all $i$: receive $s_m$ from all agents.
    (c) If $\vec{1}$ is in the convex hull of $s_l, s_m, s_h$ then goto FOUND.
    (d) Compute $v$ such that $v^\dagger(s_l - s_m) = 0$ and $d^\dagger v = 1$.
    (e) Let $v = v/\|v\|_2$.
    (f) If $-\hat{\epsilon} < v^\dagger(s_l - \vec{1}) < 0$ then set $s_h = s_m$ and goto ROUND.
    (g) Else if $v^\dagger(s_l - \vec{1}) < 0$ then $c_h = c_m$ and goto CONT.
    (h) Compute $v$ such that $v^\dagger(s_h - s_m) = 0$ and $d^\dagger v = 1$.
    (i) Let $v = v/\|v\|_2$.
    (j) If $-\hat{\epsilon} < v^\dagger(s_h - \vec{1}) < 0$ then set $s_l = s_m$ and goto ROUND.
    (k) Else let $c_l = c_m$.
    (l) CONT: Let $c_m = (c_l + c_h)/\|c_l + c_h\|_2$.
11. ROUND: Compute $0 \le \lambda \le 1$ such that $\|\lambda s_l + (1-\lambda)s_h\|_2 = v^\dagger s_l$.
12. Transmit $\lambda$, $v^\dagger s_l$, $c_l, c_h$ to all agents.
13. STOP.
14. FOUND: Compute $\lambda_l, \lambda_m, \lambda_h$ such that $\lambda_l s_l + \lambda_m s_m + \lambda_h s_h = \vec{1}$.
15. Transmit $\lambda_l, \lambda_m, \lambda_h$ and $c_l, c_m, c_h$ to all agents.
16. STOP.
```

Agent's Program — $m_i^{\epsilon,n,r}(price)$
Stage 1:

```
1. Receive $p$.
```

```
    2. Construct the ellipsoid $x_i = (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ and $B_i = Diag(\frac{2}{\sqrt{r}}, \frac{2}{\sqrt{r}}, \ldots, \frac{2}{\sqrt{r}})$
    3. Repeat $N_c = \lceil 4r^2 \log(n/\hat{\epsilon}^4) \rceil$ times.
        (a) Compute the gradient at $x_i$.
        (b) Compute a new $x_i$ and $B$ using the ellipsoid method for solving $L_i(x_i, p)$.
    4. Transmit $l_{max}$.
    5. Go to 1.
Stage 2:
    1. Receive $c$ from center.
    2. Construct the ellipsoid $x_i = (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ and $B_i = Diag(\frac{2}{\sqrt{r}}, \frac{2}{\sqrt{r}}, \ldots, \frac{2}{\sqrt{r}})$
    3. Repeat $N_c = \lceil 4r^2 \log(n/\hat{\epsilon}^3) \rceil$ times.
        (a) If $L_i(x_i, p)$ is sufficiently good then use $v = c$.
        (b) Else let $v = \nabla L_i(x_i, p)$.
        (c) Compute a new $x_i$ and $B$ using $v$.
    4. Transmit $x_i$.
    5. Go to 1.
    6. ROUND:
    7. Receive $\lambda$, $v^\dagger s_l$, $c_l, c_h$ from the center.
    8. Compute $x_i = (\lambda x_i(c_l) + (1-\lambda) x_i(c_h))/v^\dagger s_l$, where $x_i(c)$ is computed as in the above
       loop.
    9. STOP.
    10. FOUND:
    11. Receive $\lambda_l, \lambda_m, \lambda_h$ and $c_l, c_m, c_h$ from the center.
    12. Compute $x_i = \lambda_l x_i(c_l) + \lambda_m x_i(c_m) + \lambda_h x_i(c_h)$.
```

13. STOP.

Theorem 4 *For any δ the two resource price mechanism $m^{\epsilon,n,r}(price)$ has complexity*

$$T = \mathcal{O}((\log(n/\epsilon))[X(1 + \log(n/\epsilon)) + I + U\log(n/\epsilon))$$

with probality greater than $1 - \delta$.

Proof: The proof is given in the appendix.

For multiple resources the main ideas for the two resource case easily generalize. However, we have not completed the construction and proof for stage 2 and leave this for future work.

Conjecture 1 *The r resource price mechanism $m^{\epsilon,n,r}(price)$ has complexity*

$$T = \mathcal{O}((r^3\log(n/\epsilon))[X(1 + \frac{\log 1/\epsilon}{n} + \frac{r^2}{n} + r^4\log(n/\epsilon)) + Ir^2 + Ur^2\log(n/\epsilon))$$

Proof (partial): The complexity is computed assuming that stage 1 dominates stage 2. The correctness of stage 1 is shown in the proof for the two resource case. The construction of stage 2 is a yet incomplete, However, we believe that a method similar to that for two resources should work. $\diamond$

5 Resource Allocation with Externalities

In many instances the utility an agent derives from a certain bundle of resources is dependent on the resources used by the other agents. Two classic examples of this are pollution and congestion. These are both negative externalities, as the increased total use of resources reduces the agents derived utility.

Positive externalities are also possible. For example the usefulness of a communication network may depend on the number of people using it, assuming that it is operating below capacity. In this case if many people use the network then more people are reachable on the network thus increasing its value. (see e.g. [OS81].) However, in this paper we will only consider negative externalities, as they are the most common in resource allocation and distributed computing.

To account for externalities, we modify our formulation slightly, and assume negative externalities of a simple and common form. We allow the utility functions to depend on $\lambda = (\lambda^1, \ldots, \lambda^r)$ where $\lambda^j = \sum_{i=1}^n x_i^j$. So

$$U(x, \lambda) = \sum_{i=1}^{n} U_i(x_i, \lambda)$$

We require that $U(x)$ is concave and monotonically decreasing in λ. Since λ is a negative externality, we can be less restrictive about the definition of λ, by viewing it as just another variable subject to $\lambda^j \geq \sum_{i=1}^n x_i^j$, as the optimal solution will always occur when the constraint is binding.

We can modify both the ellipsoid and descent mechanisms quite simply, to solve the resource allocation with externalities. We note that the transformation $\hat{\lambda} = 1 - \lambda$ allows us to rewrite

$$F' = \{x \mid 0 \leq x,\ 0 \leq \hat{\lambda},\ \sum_{i=1}^{n} x_i^j + \hat{\lambda}^j \leq 1\}$$

and note that F' is equivalent to F with the addition of r new variables. Thus this new problem is almost identical structurally to the problem without externalities, and a slight modification of these mechanisms allows them to solve the new problem with negligible loss of efficiency.

The construction for an arbitrary (including Price) mechanisms is more complicated. Let $T(\epsilon, r, n)$ be the time required by an arbitrary efficient mechanism that solves the resource allocation problem without externalities. Note that this time is unchanged if we solve the generalized problem, but fix the externality effect to a certain level, say $\lambda_0 \leq 1$. Letting

$$V(\lambda) = \max_{x \in F} U(x, \lambda) \mid \lambda = \lambda_0$$

we can apply a modified version of the ellipsoid method to maximize $V(\lambda)$. (Note that $V(\cdot)$ is convex.) This allows us to show.

Theorem 5 *Let $m^{\epsilon,n,r}$ be an efficient mechanism for allocating resources without externalities. Then there exists a related mechanism $m_{ext}^{\epsilon,n,r}$ which solves the problem with externalities in*

$$T_{externalities}(\epsilon, r, n) = \mathcal{O}(r^3 \log(1/\epsilon)(T(\epsilon, r, n) + r)).$$

where $T(\epsilon, r, n)$ is the running time of $m^{\epsilon,n,r}$.

Proof: There exists a modified version of the ellipsoid mechanism which can maximize $V(\lambda)$ over $\{\lambda \mid 0 \leq \lambda^j \leq 1\}$ using only $\mathcal{O}(r^2 \log 1/\epsilon)$ iterations and $2r + 1$ $\mathcal{O}(\epsilon^4)$-accurate function values of $V(\lambda)$ with only $\mathcal{O}(r^2)$ computations per iteration. This is shown in [NY83]. Allowing the center to use the algorithm shows that

$$T_{externalities}(m_{ext}^{\epsilon,n,r}) = r^3 \log(1/\epsilon)[T(\epsilon^3, r, n) + r]$$

and since the mechanism is assumed to be efficient it must be linearly convergent. Therefore the ϵ^4 in the complexity can be replaced by ϵ as this can only changes the running time by a multiplicative constant as $\log \epsilon^3 = 3 \log \epsilon$. □

Thus externalities do not increase the mechanism's dependence on n significantly or decrease its convergence rate.

6 Conclusions

We have described a general framework for constructing and comparing mechanisms for distributed resource allocation under bounded rationality. This has implications for both economics and optimization.

We have developed a strict computational model of resource allocation with realistic agents. This is in contrast to the large body of economic work where agents are given unbounded computational ability. Also our measure of complexity is much more relevant than the standard definitions of complexity as the size of message space. Using this model we have still been able to design efficient mechanisms for resource allocation and compute their efficiency.

Another major contribution of our work is explicitly considering mechanisms with a large number of agents. Many of the mechanisms in the literature either implicitly or explicitly apply to a very small group of agents, often two or three. In contrast, we are interested in mechanisms that apply to large or very large groups of agents. For resource allocation this is an important distinction, as allocating resources to a small group of agents is relatively easy, and the differences between different mechanisms is probably unimportant.

Thus our major contribution to economic theory comes by raising these two issues, computation and number of agents, and developing a theory that highlights them, which can also be used to develop mechanisms which solve the problem.???

For nonlinear programming, we have developed a rigorous theory of complexity for distributed continous optimization. This is to be contrasted with the standard asymptotic analysis used in nonlinear programming. Asymptotic analysis is only a very local concept and does not give any precise information about global convegence rates. Also, asymptotic analysis requires strong analyticity assumptions.

In contrast our complexity theory of 'global optimization' requires no assumptions other than convexity and smoothness, and we believe that the smoothness assumptions can be removed with a more detailed analysis. This new theory is motivated by the complexity theory of combinatorial optimization, studies by Nemirovsky and Yudin [NY83], work by Hochbaum and Shanthikumar [HS90], and global methods in optimiztion by Shub and Smale [SS92] based on the theory of computation over the real numbers [BSS88] and strong polynomiality.

Our results on price mechanisms in this paper and previous results [Fri92] can be seen as specific applications of this theory and preliminary results. These algorithms are probably not competetive with modern methods used in nonlinear programming, as yet. However, we believe that a global view of optimization may lead to a practically useful method of solving nonlinear optimization problems. We draw inspiration from the recent development of interior point algorithms for linear programming.

In summary, this paper can be interpreted as both descriptive and proscriptive. The descriptive aspect applies to the study of mechanisms and complexity in resource allocation. The proscriptive element applies to the theory of nonlinear programming, and is far more speculative.

A Proof of Theorem 4

In this appendix we give the proof of theorem 4. For stage 1 of the mechanism we give the proof for an arbitrary number of resources as this is notationally cleaner and requires no extra effort. For stage 2 we explicity consider the case of two resources. In this part of the proof the argument requires this restriction. However, we believe that this should generalize in a natural manner.

A.1 Stage 1

First we describe the basic concepts required by the proof and set some of the notation. Duality theory of convex programming [Lue84] allows us to rewrite the problem in the following manner. Define $s(x) = \sum_i x_i - 1$. Then let

$$L(x,p) = U(x) - p^{\dagger} s(x)$$

then define

$$F(p) = \max_x L(x,p)$$

which is a convex function of p and define $x(p)$ to be the allocation that achieves the maximum.

Now the basic theory of convex duality says that the minimum over p of $L(p)$ is equal to the maximum of the allocation problem. Also, if p^* is the price that achieves the maximum then $x(p^*) = x^*$ is the optimal allocation.

Using this we discover a natural distributed method for the allocation problem. Define

$$L_i(x_i,p) = U(x) - p^{\dagger}(x_i - 1/n)$$

so that $L(x,p) = \sum_{i=1}^{n} L_i(x_i,p)$. Define $F_i(p)$ similarly. Then given a p we can allow each agent to solve his own

$$\max_{x_i} L_i(x_i,p),$$

then a central processor can solve the low dimensional problem of finding the minimum of $F(p)$.

The first major difficulty which we must overcome is that of approximating gradients with only information about the functional value of $F(p)$. Deterministically this is very difficult as the function can have many 'kinks' where the value of the gradient can change abruply. However, if we choose the places to evaluate the function at random, then we can get a useful approximation of the gradient with only a small number of query points.

Lemma 3 *The procedure for approximating gradients is $\hat{\epsilon}$-accurate with probability greater than $1-\alpha$.*

Proof: See [NY83]. ◇

Lemma 4 *The ellipsoid method in stage one will fail with probability less than δ.*

Proof: The probability of error at each iteration is less than α. As there are $r^2 \log(1/\hat{\epsilon})$ iterations, the probability of having a failure at any stage during the entire process is less than $r^2 \log(1/\hat{\epsilon})\alpha = \delta$. ◇

Using the gradient computed by the method in the previous section, we can modify the ellipsoid algorithm slightly to still produce an $\hat{\epsilon}$-accurate answer. An elementary description of this can be found in [PS82].

Thus if we compute the values of $F(p)$ to accuracy $\hat{\epsilon}^3$ then the ellipsoid method described above computes an $\hat{\epsilon}$-accurate solution $\hat{p}, \hat{x}$ such that $|F(\hat{p}) - F(p^*)| \leq \hat{\epsilon}$ and $|L(x(\hat{p}), \hat{p}) - L(\hat{x}, \hat{p})| \leq \hat{\epsilon}$. Thus stage one can compute an $\hat{\epsilon}$-approximation $\hat{p}$ to the minimum of $F(p)$ and also an $\hat{\epsilon}$-approximation to the actual value of $F(\hat{p})$.

Thus we have shown:

Lemma 5 *For any δ, stage 1 produces a $\hat{p}$ and $\hat{x}$ such that*

$$|F(p) - F(p^*)| \leq \hat{\epsilon}$$

and

$$|L(\hat{x}, \hat{p}) - F(\hat{p})| \leq n\hat{\epsilon}^4$$

and therefore

$$|F(p^*) - L(\hat{x}, \hat{p})| \leq 2\hat{\epsilon}$$

for $\hat{\epsilon} \leq 1/2$, with probability greater than $1-\delta$.

Proof: The first statement follows from the argument above on the accuracy of the ellipsoid method with the randomized algorithm for computing gradients. The second from the accuracy of the agent's own computation of $x(\hat{p})$, and the third from combining the first two. ◇

A.2 Stage 2

In this section we construct a feasible solution as the convex combination of several $\hat{\epsilon}^2$-accurate solutions of

$$\max_x c^\dagger x \text{ s.t. } x \in G$$

where $G = G_1 \times \cdots \times G_n$ and

$$G_i = \{x \mid L(x_i, \hat{p}) - L(\hat{x}_i, \hat{p}) \leq 4\hat{\epsilon}\}$$

This is computed using the ellipsoid method by the individual agents as the problem is seperable into n independent pieces.

Theorem 6 *The computation of*

$$\max_x c^\dagger x_i \text{ s.t. } x_i \in G_i$$

to accuracy $\hat{\epsilon}^2$ can be accomplished in $r^2 \log(r/2\hat{\epsilon}^3)$ iterations of the ellipsoid method.

Proof: Note that G_i must contain the hypersphere of radius $\hat{\epsilon}$ as the Lipschitz constant of $L_i(x_i, \hat{p})$ is less than 1. This hypersphere must contain a cube of side $2\hat{\epsilon}/\sqrt{r}$ and thus has volume greater than $(2\hat{\epsilon}/\sqrt{r})^r$. The ellisoid method requires $r^2 \log(1/(V^{1/r}\epsilon)$ steps to compute an ϵ-accurate solution in a feasible convex region of volume V. $\diamond$

Now we show that there exists a feasible solution in G. This will follow from a sequence a lemmas.

Lemma 6 *Given p is an $\hat{\epsilon}^2$ accurate solution of $F(p)$, there exists a p' such that $\|p - p'\|_2 \leq \hat{\epsilon}$ and $\|\nabla F(p')\|_2 \leq \hat{\epsilon}$.*

Proof: Consider the path generated by

$$\frac{dp(t)}{dt} = \nabla F(p(t)) \quad \text{s.t.} \quad p(0) = p$$

and let $\gamma(x)$ represent the path of $p(t)$ measured in arclength, then

$$F(\hat{p}) - F(p^*) = \int_0^1 \nabla F(\gamma(x))^\dagger d\gamma = \int_0^1 \|\nabla F(\gamma(x))\|_2 \, d\gamma$$

as this is just the line integral of ∇F. By assumption $F(\hat{p}) - F(p^*) \leq \epsilon^2$. Thus the following must hold

$$\int_0^{\hat{\epsilon}} \|\nabla F(\gamma(x))\|_2 \, d\gamma \leq \hat{\epsilon}^2$$

as the integrand is non-negative.

Now assume that the theorem is false. This implies that for all $x \leq \hat{\epsilon}$ we must have that $\|\nabla F(\gamma(x))\|_2 > \hat{\epsilon}$ as the euclidian distance $\|\gamma(0) - \gamma(\hat{\epsilon})\|_2$ is less than the distance allong γ. However, this implies that

$$\int_0^{\hat{\epsilon}} \|\nabla F(\gamma(x))\|_2 \, d\gamma > \hat{\epsilon}^2$$

as

$$\int_0^{\hat{\epsilon}} \|\nabla F(\gamma(x))\|_2 \, d\gamma < \hat{\epsilon} \min_{0 \leq x \leq 1} F(\gamma(x))$$

thus providing a contradiction and proving the theorem. $\diamond$

However, the gradient of F is related to feasibility.

Lemma 7 *The feasibility,* $s(p) - \vec{1} = \nabla F(p)$.

Proof: This follows from the well know envelope theorem,

$$\frac{dF(p)}{dp} = \frac{\partial F(p)}{\partial p} = s(p) - \vec{1}$$

$\diamond$

Using this we show that a feasible solution exists for the above optimization problem.

Lemma 8 *Given p is an* $\hat{\epsilon}^2$ *accurate solution of* $F(p)$, *there exists an* x_i *such that* $|L(\hat{x}_i, \hat{p}) - L(x_i, \hat{p})| \leq 4\hat{\epsilon}$ *and* x_i *is feasible.*

Proof: From the previous lemma there exists a p' such that $\left\|s(p') - \vec{1}\right\|_2 \leq \epsilon$ then letting $x^j = x^j(p')/s^j(p')$ gives a x that is feasible and Noting that

$$\|x - x(p')\|_2 \leq 2\hat{\epsilon}$$

implies that

$$|L(x_i(p'), p') - L(x_i, \hat{p})| \leq 2\hat{\epsilon}$$

by the Lipschitz constant of L_i. Now by the Lipschitz constant of F_i for changing p we know that

$$|F_i(\hat{p}) - F_i(p')| \leq \hat{\epsilon}$$

Combining these inequalities produces the required result. $\diamond$

Now that we have shown that a G contains at least one feasible solution we will describe how to construct on of these.

A.3 Constructing a Feasible Solution

The easiest way to understand the stage 2 algorithm is to consider the map

$$s : S^1 \to \Re^r$$

defined by

$$s_i(\theta) = \mathrm{argmax}_{x_i}\ (\cos\theta, \sin\theta)^\dagger x_i \ \text{ s.t. } x \in G$$

and $s(\theta) = \sum_i s_i(\theta)$. Note that in the mechanism we work directly with $c = (\cos\theta, \sin\theta)$ instead of θ. For the purposes of the proof it is much clearer to work in θ directly.

The mechanism works in the following manner. The feasible point $\vec{1}$ must lie on one side of the line between s_l and s_h. We choose θ_m such that s_m is on the same side of the line that $\vec{1}$ is. Now these three points create three possible regions in which $\vec{1}$ can lie. (See figure 4.) If the point lies in the convex hull of s_l, s_h, s_m (region I) then we stop. Also if the point lies close to one of the lines $\overline{s_l s_m}$ or $\overline{s_h s_m}$ then we also stop. However if the point lies in one of the remaining regions (regions II or III) then we choose the two points that define that region to be our new θ_l and θ_h and continue the process.

In order to prove that the mechanism constructs a feasible solution we note that Stage 2 actually performs a binary search in θ, and the only ways it can halt are:

1. A feasible solution x is found as the convex combination of 3 points x_l, x_m, x_h.
2. A solution x with $\left\|s(x) - \vec{1}\right\|_2$ is the convex combination of two points x_l, x_h.
3. $\theta_l - \theta_h \leq 2\pi/\hat{\epsilon}$.

In the first case the following lemma shows that the convex combination of x_l, x_m, x_h that gives a feasible solution that is $\hat{\epsilon}$-accurate.

Lemma 9 *Let u, v, w all satisfy $|L(u, \hat{p}) - F(\hat{p})| \leq 4\hat{\epsilon}$ (resp. v, w). Assume that $x = \lambda_u u + \lambda_v v + \lambda_w w$ with $0 \leq \lambda_u, \lambda_v, \lambda_w$ and $\lambda_u + \lambda_v + \lambda_w = \vec{1}$. Then $|L(x, \hat{p}) - F(\hat{p})| \leq 4\hat{\epsilon}$.*

Proof: By convexity of L we have that $L(x, \hat{p}) \leq \max[L(u, \hat{p}), L(v, \hat{p}), L(w, \hat{p})]$. However by definition $F(\hat{p}) > L(x, \hat{p})$. $\diamond$

From the previous lemma we see that for case 2 we can simply round the solution to feasibility.

Lemma 10 *In case two the allocation defined by $x^j = x^j / s^j(x)$ satisfies $|L(x, \hat{p}) - F(\hat{p})| \leq 2\hat{\epsilon}$.*

Proof: Note that x_j is feasible. Also $\|x' - x\|_2 \leq \epsilon$ as $\|s(x) - 1\|_2 \leq \hat{\epsilon}$. Thus by the Lipschitz continuity of L we get the desired result. $\diamond$

Now we will show that case 3 is actually equivalent to case 2 as $|\theta_l - \theta_h| \leq 2\pi\hat{\epsilon}$ implies that there exists a solution x which is the convex combination of $x(\theta_l)$ and $x(\theta_h)$ and $\|s(x) - 1\|_2 \leq \hat{\epsilon}$.

Lemma 11 *Assume that the center's stage 2 algorithm received the exact solutions $s(\theta)$ at each iteration. Then if $|\theta_l - \theta_h| \leq \hat{\epsilon}$ then 1 must be within $\hat{\epsilon}$ of the line $\overline{s_l s_h}$.*

Proof: This follows from elementary geometry and the fact that the feasible solution 1 must be contained in the region bounded by $\overline{s_l s_h}$ by construction. Also we know that $c(\theta_l)^\dagger s_l \geq c(\theta_l)^\dagger 1$ and $c(\theta_m)^\dagger s_m \geq c(\theta_m)^\dagger 1$ as these (s_l, s_h) are the respective maxima of their respective c's. This region forms a triangle with angle $\alpha = \pi - \hat{\epsilon}$ as $\hat{\epsilon} = |\theta_h - \theta_l|$. (See figure 5.)

Since we know that $\|s_l - s_h\|_2 \leq 2$ the height of the triangle must be less than $2 \sin \hat{\epsilon}/2$ which is less than $\hat{\epsilon}$. Thus as the point 1 is contained within this triangle, we see that the distance from 1 to $\overline{s_l s_h}$ must be less than $\hat{\epsilon}$. $\diamond$

However, the center does not recieve exact solutions from the agents. The solutions from the agents are only accurate to $\hat{\epsilon}^2$. In fact the solutions $\hat{s}(\theta)$ can be arbitrarily far from the true solutions $s(\theta)$. However, as the next lemma shows, these solutions are actually close to the real solution for a slightly different θ. Thus we can imagine that the values of θ are uncertain. However, this does not effect the binary search if these errors in θ are less than $\hat{\epsilon}$.

Lemma 12 *Assume that $|c(\theta)^\dagger(s - s(\theta))| \leq \hat{\epsilon}^2$. Then there exists a θ' such that $|\theta - \theta'| \leq 2\pi\hat{\epsilon}$ and $\|s - s(\theta')\|_2 \leq \hat{\epsilon}$.*

Proof: As $|c(\theta)^\dagger(s - s(\theta))| \leq \hat{\epsilon}^2$ we see that there exists a $\hat{\theta}$ such that $\left\|s - s(\hat{\theta})\right\|_2 \leq \hat{\epsilon}$. Let $\gamma(x)$ be the boundary of the feasible region with $\gamma(0) = s(\theta)$ and $\gamma(\hat{\theta}) = s(\hat{\theta})$ and assume that γ is parametrized by arc length.

We will now show that there exists a θ' such that $|\theta' - \theta| \leq \hat{\epsilon}$ while $\left\|s(\theta') - s(\hat{\theta})\right\|_2 \leq \hat{\epsilon}$. Combining these inequalities produces the theorem.

First note that at $s(\theta)$ the normal to γ must point in the direction defined by θ. Note that

$$c(\theta)^\dagger(s(\theta) - s(\beta)) = \int_0^{x(\beta)} \tan(\theta(u) - \theta)du \leq \int_0^{x(\beta)} 2/\pi \; (\theta(u) - \theta)du$$

where $x(\beta)$ is the smallest x such that $\theta(x) \leq \beta$.

Now assume the lemma is false and all $\theta(x)$ for $|x - x(\hat{\theta})|$ satisfy $|\theta(x) - \theta| > \hat{\epsilon}$, therefore

$$c(\theta)^{\dagger}(s(\theta(x)) - s(\theta)) > \hat{\epsilon}^2$$

by the previous equation. However this contradicts the fact that $\hat{s}$ is $\hat{\epsilon}^2$-accurate proving the lemma. $\diamond$

Thus the algorithm works even with the inexact replies given by the agents. $\square$

B List of Figures

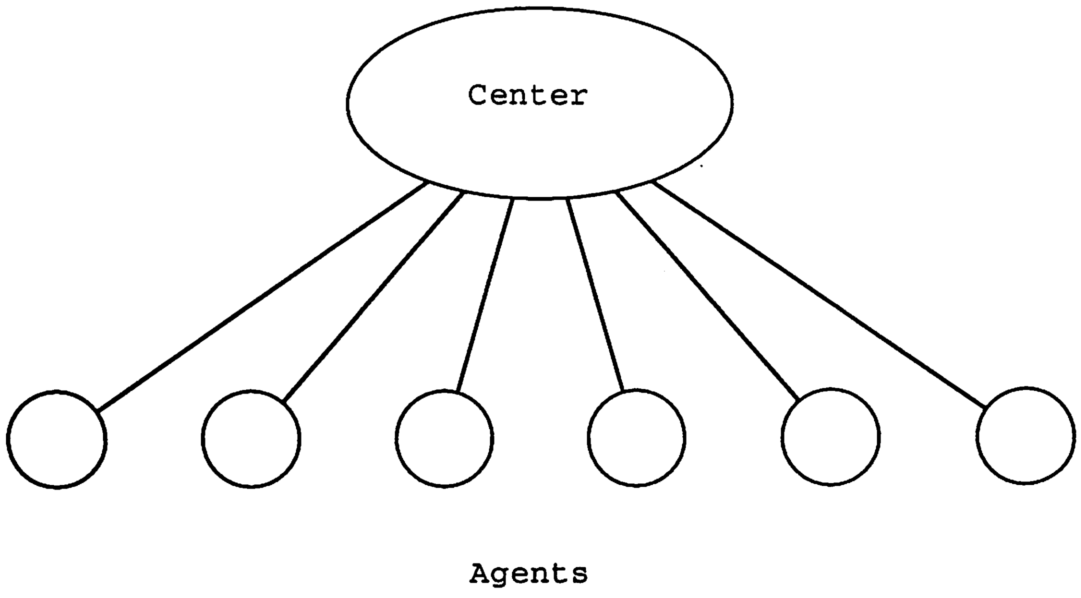
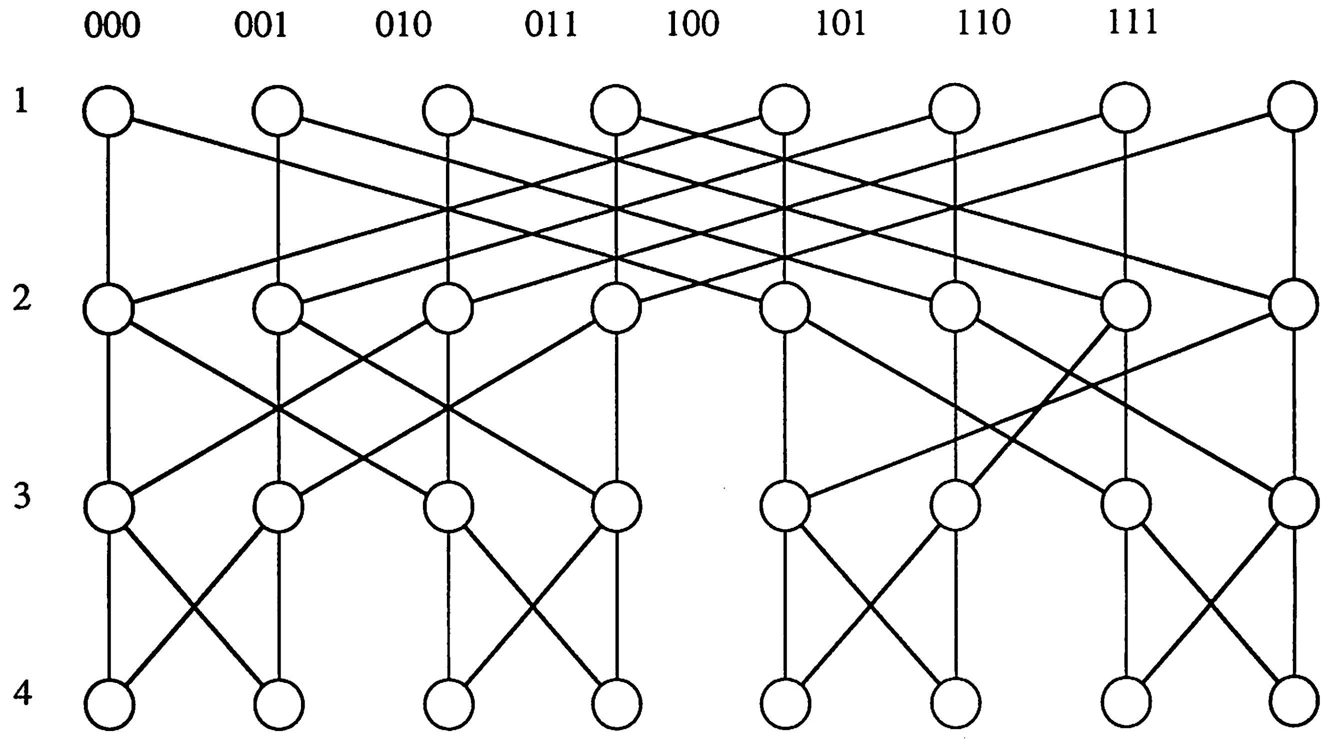


Figure 1 : A Central mechanism.



Agents

Figure 2 : Agents organized on a Butterfly network.

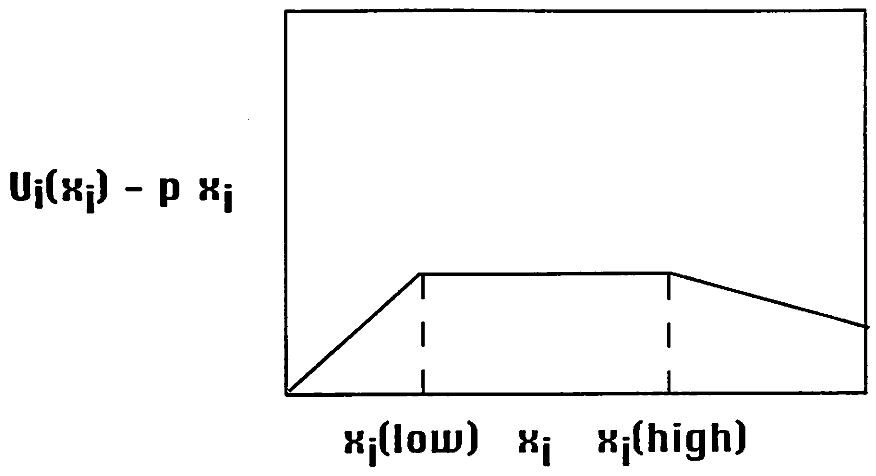


Figure 3: Multiple allocations can result from a single price. Note any allocation between x(low) and x(high) is of the same value to consumer i.

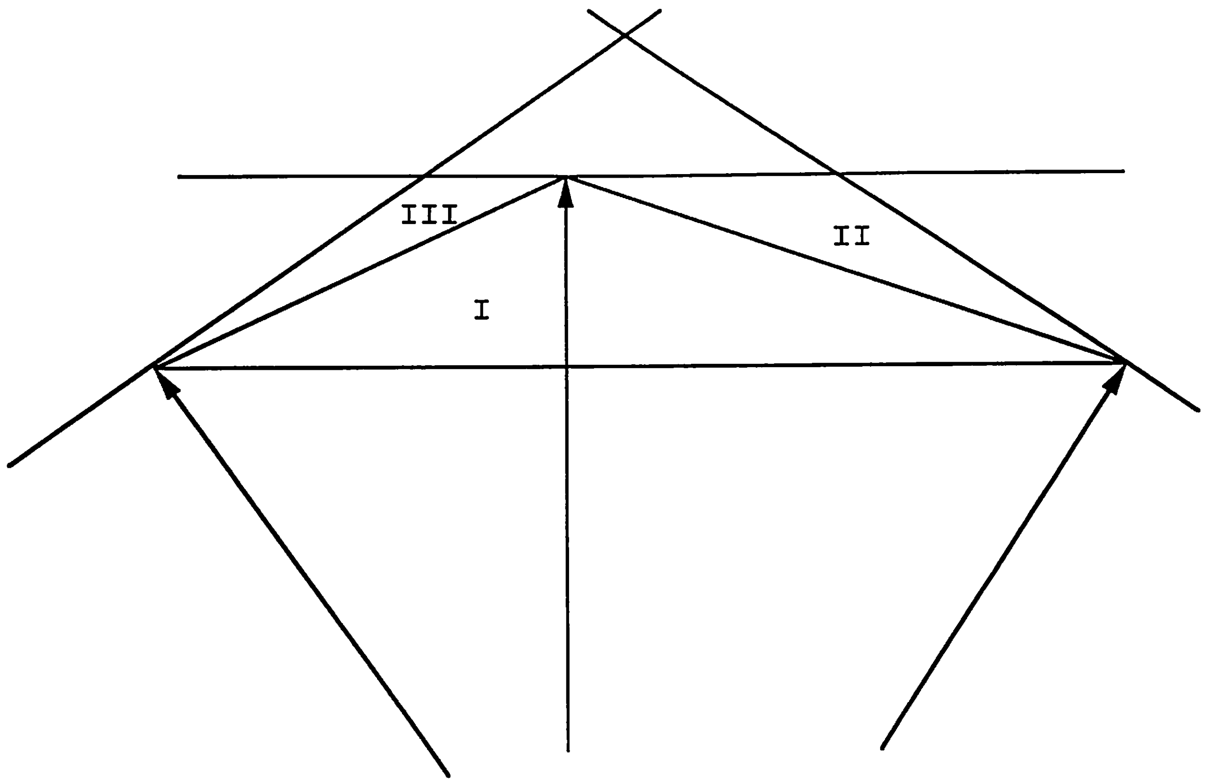


Figure 4: Geometry of Stage 2 Algorithm

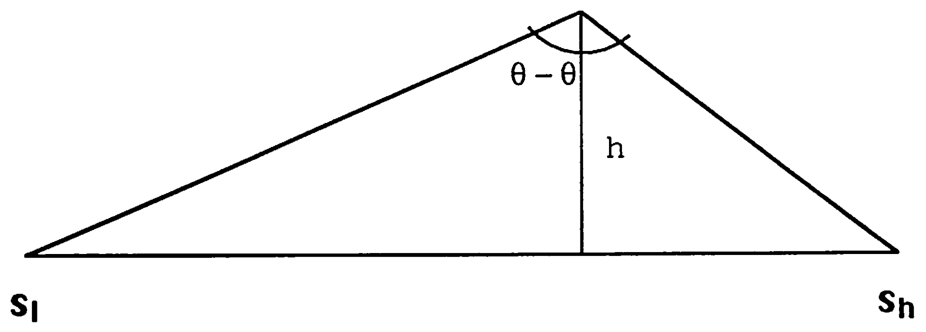


Figure 5: Accuracy of stage 2.

References

[AH60] K. Arrow and L. Hurwicz. Decentralization and comoputation in resource allocation. In R. Pfouts, editor, *Essays in Economics and Econometrics.* University of North Carolina Press, Chapel Hill, 1960.

[Blu89] L. Blum. Lectures on a theory of computation and complexity over the reals (or an arbitrary ring). Technical Report TR–89–065, International Computer Science Institute, Berkeley, CA, December 1989.

[BSS88] L. Blum, M. Shub, and S. Smale. On a theory of computation over the real numbers: *NP*-completeness, recursive functions, and universal machines. *Bulletin of the ACM*, 21(1):1–46, 1988.

[DS66] N. Dunford and D. Schwarts. *Linear Operators Part 1: General Theory.* Interscience Publishers, New York, 1966.

[EK83] J.G. Ecker and J. Kupfershmid. The ellipsoid algorithm for nonlinear programming. *Mathematical Programming*, 27:83–106, 1983.

[FNY88] D. W. Ferguson, C. Nikolau, , and Y. Yemini. Microeconomic algorithms for load balancing in distributed computer systems. *Proceedings of the 8th International Conference on Distributed Computing Systems*, 1988.

[Fri92] E. J. Friedman. A strongly polynomial algorithm for convex programs with combinatorial constraints and resource allocation. Technical Report UCB/ERL/IGCT M92/6, Interdisciplinary Group on Coordination Theory, Electronics Research Laboratory, University of California, Berkeley, CA 94720, January 1992.

[HM85] L. Hurwicz and T. Marschak. Discrete allocation mechanisms: Dimensional requirements for resource-allocation resource-allocation mechanisms when the desired outcomes are unbounded. *Journal of Complexity*, 1:264–303, 1985.

[Hoc] D. S. Hochbaum. Lower and upper bounds for the allocation problem and other nonlinear optimization problems. Manuscript, School of Business Administration and Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720. September 1989, Revised December 1990.

[HS90] D. S. Hochbaum and J. G. Shanthikumar. Convex separable optimization is not much harder than linear optimization. *Journal of the ACM*, 37(4):843–862, October 1990.

[Hub88] B. Huberman. *The Ecology of Computation.* North Holland, Amsterdam, 1988.

[Hur73] L. Hurwicz. The design of mechanisms for resource allocation. *The American Economic Review*, pages 1–30, May 1973.

[Hur77] L. Hurwicz. On the dimensional requirements of informationally decentralized pareto satisfactory adjustment processes. In K.J. Arrow and L. Hurwicz, editors, *Studies in Resource Allocation Processes.* Cambridge University Press, Cambridge and New York, 1977.

[Hur86a] L. Hurwicz. On informational decentralization and efficiency in resource allocation mechanism. In S. Reiter, editor, *Studies in Mathematical Economics.* Mathematical Association of America, Washington D.C., 1986.

[Hur86b] L. Hurwicz. On informationally decentralized systems. In C.B. McGuire and R. Radner, editors, *Decision and Organization.* University of Minnesota Press, Minneapolis, 1986.

[IK88] T. Ibaraki and N. Katoh. *Resource Allocation Problems: Algorithmic Approaches.* MIT Press, 1988.

[Kha79] L.G. Khachian. A polynomial algorithm for linear programming. *Soviet Math. Doklady*, 20:191–4, 1979.

[KS89] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed compute systems. *IEEE Transactions on Computers*, 38:705–16, 1989.

[Lei91] F. T. Leighton. *Introduction to parallel algorithms and architectures.* Morgan-Kaufman, 1991.

[Lev65] A. Yu Levin. On an algorithm for minimizing convex functions. *Soviet Maths*, 1:286–90, 1965.

[Lue84] David G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, Reading, Massachusetts, second edition, 1984.

[Mar86] T. A. Marschak. Organizational design. In K. J. Arrow and M.D. Intrilligator, editors, *Handbook of Mathematical Economics.* North–Holland, New York, 1986.

[Meg83] N. Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM J. Comp.*, 12(2):347–59, 1983.

[MR74] K. Mount and S. Reiter. The informational size of message spaces. *Journal of Economic Theory*, 9, 1974.

[MR87] K. Mount and S. Reiter. The existence of a locally stable dynamic process with a statically minimal message space. In Theodore Groves, Roy Radner, and Stanley Reiter, editors, *Information, Incentives, and Economic Mechanisms: Essays in Honor of Leonid Hurwicz.* University of Minnesota Press, Minneapolis, 1987.

[MR90] K. Mount and S. Reiter. A model of computing with human agents. Discussion Paper No. 890. Center for Mathamatical Studies, J.L. Kellogg Graduate School of Management, Northwestern University, Evanston, Illinois 60208, 1990.

[NY83] A.S. Nemirovsky and D.B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* Wiley, New York, 1983.

[OL74] S.S. Oren and D.G. Luenberger. Self-scaling variable metric algorithms I: criteria and sufficient conditions for scaling a class of algorithms. *Management Science*, 20:845-862, 1974.

[OS81] S.S. Oren and S. Smith. Critical mass and tariffs in electronics communication markets. *The Bell Journal of Economics*, 12(2):467–487, Autumn 1981.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[Roc70] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[RW91] S. Russell and E. Wefald. *Do the Right Thing: Studies in Limited Rationality.* MIT Press, Boston, 1991.

[San85] B. Sanders. A private good/public good for optimal flow control of an m/m/1 queue. *IEEE Transactions on Automatic Control*, 30:1143–5, 1985.

[San88] B. Sanders. An incentive compatible flow control algorithm for rate allocation in computer networks. *IEEE Transactions on Computers*, 37, 1988.

[Sca60] H. Scarf. Some examples of global instability of the competetive equilibria. *International Economic Review*, 1:157–72, 1960.

[She90] S. Shenker. Efficient network allocations with selfish users. Preliminary Draft, 1990.

[Sim86] H. A. Simon. Theories of bounded rationality. In C.B. McGuire and R. Radner, editors, *Decision and Organization.* University of Minnesota Press, Minneapolis, 1986.

[Sma76] S. Smale. A convergent process of price adjustment and global newton methods. *Journal of Math. Econ.*, 3:107–20, 1976.

[Son85] G. Sonnevend. An "analytical centre" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. In A. Prekopa, editor, *Lecture Notes in Control and Information Science: 84.* Springer-Verlag, Berlin, 1985.

[SS92] M. Shub and S. Smale. The complexity of bezout's theorem. parts I, II, and III. Unpublished Manuscript, 1992.

[Var78] H. R. Varian. *Microeconomic Analysis.* W.W. Norton, New York, 1978.