CS 599: Algorithmic Game Theory	August 25, 2010

Lecture 1 - Overview of Algorithmic Game Theory

Prof. Shang-hua Teng *Scribes: Joseph Bebel and Henry Yuen*

1 Our starting point: game theory

This class is going to be about Algorithmic Game Theory (AGT). That requires us to understand game theory a bit. What are the characteristics of game theory?

In game theory, the fundamental objects of consideration are *games*, which involve a number of *players* and their *strategies*. In particular, we assume players to be *selfishly motivated*, or at least have an *objective* each player wishes to optimize. An important concept in game theory is the idea of *equilibrium*; intuitively, equilibrium in a game is a situation in which no player has an incentive to change her strategy. One of the most important results in game theory is what is called *Nash equilibrium*.

One of the main applications of game theory is to model economic situations, ranging from simple two player exchanges to huge online marketplaces. Of course, now this begs the question, what is *economics*? An instinctual definition of economics: it is the study of how we all use money.

But economics need not be about money. As we shall see later, we can study an idealized market model (albeit one that has been realized at one point in time) that does not use money whatsoever. A better definition, and more textbook-like: economics is the study of scarce resources, each of which have different uses.

Most people agree that game theory really took off with von Neumann's papers and book *Theory of Games and Economic Behavior* in the early 20th century. von Neumann, man of many talents, also played a seminal role in the development of computer algorithms. Game theory and computer science took off on their own paths, but the advent of the internet and electronic commerce has brought the two fields together in a meaningful way, giving rise to *algorithmic game theory*, which sits at the intersection of economics, theoretical computer science, operations management, and industrial systems engineering.

2 Computer science, and what it can do for you!

We all love computer science. We love its algorithms, its theory, and its widespread applicability. In particular, we all love its most important question: **P** vs **NP**. The question is about our burning desire to understand the intrinsic difficulty of computational problems. Answering this question will net you one million dollars - there's definitely some interesting economics in that!

Computer science is also heavily concerned with *constraint optimization* problems, which are formulated as follows:

Optimization problem: Maximize/minimize an *objective* function $f(x)$, subject to the constraints $x \in C$.

Examples of objective functions and constraints can vary wildly. For example, we may want to minimize the cost of building a toy factory, but subject to the constraint that we actually build at least 100,000 toys.

One of the big players in developing the computer science of optimization is AT&T labs, who wished to use this knowledge to solve a very practical problem: improving the reliability of its phone networks. Because of their efforts, as well as of computer scientists around the world, by the 1990's, lots and lots was known about single-objective optimization. Computer scientists then studied *multiobjective optimization*:

Multiobjective optimization problem: Optimize $f_1(x), f_2(x), \ldots, f_k(x)$ subject to the constraints $x \in C$.

By itself, multiobjective optimization already has many uses in economics. However, multiobjective optimization in itself is still too simple to capture the dazzling complexity in real world economic situations. For example, optimizing an objective function (or an array of objective functions) is doable and well understood, but what happens when you have many people all *competing* to optimize objective functions? One thing we can do is to view these situations as *games*.

Here's where everything comes together: traditional game theory can answer things about the *existence* of equilibria for these kinds of games. However, being the devoted computer scientists we are, we also are dying to know equally important things: how *complicated* is computing equilibrium? Is it possible to *reach* equilibrium? This is a starting point of algorithmic game theory: applying ideas and models from computer science to game theory. The tools of theoretical computer science are well-suited towards the study of games, mostly because of our deep understanding of optimization problems.

Nowadays, the confluence of algorithms and game theory has a very pragmatic application: the internet, e-commerce, and social networks have realized fantastically complex networks and games. Google AdWords is a prime example of algorithmic game theory in action. So answering questions in algorithmic game theory has a definite real-world impact.

Just as we marked John von Neumann as the father of game theory and computer algorithms, we can identify Christos Papadimitriou with writing the paper that put together the vision of AGT as we know it today: in 1991, he wrote *The Complexity of the Parity Argument and Other Inefficient proofs of Existence*.

3 A Brave New World

So what is the setup of our new paradigm? Whereas in both single- and multi-objective optimization, the problem is essentially about *one* person's interests, we have to consider multiple players now, and their interests, which may - and generally will - be conflicting. For simplicity, we will only consider a single objective function now: that is, every player seeks to optimize their own objective function, but only one of them.

So formally, let's say we have players $p_1, \ldots, p_n$, and they each have objective functions $u_1, \ldots, u_n$ they want to optimize, respectively.

$$\begin{array}{c|c} p_1 & u_1(x_1, \ldots, x_n) \\ \vdots & \vdots \\ p_n & u_n(x_1, \ldots, x_n) \end{array}$$

Player p_i controls variable x_i only (note that these variables themselves may represent vectors or some other kind of structure - not just single numbers). One can see where action comes from: Player 1 wishes to optimize his objective function u_1, but not all the variables are under his control! He can only choose to change x_1 in response to what others have chosen for $x_2, \ldots, x_n$. Of course, each of the x_i's have to satisfy constraints: x_i must belong to some constraint set C_i.

This is the basic setup of a game. It is ridiculously general, but now we have some language to work with. Before we continue, we will be entertained by a quiz:

Quiz 1 Suppose $G = (V, E)$ is a finite directed acyclic graph (DAG). Prove that there exists a vertex in V with outdegree is 0.

Proof. Take a vertex $v_1 \in V$. If it has outdegree 0, then we are done. Otherwise, arbitrarily select a neighbor v_2 that is connected to v_1 via an out-going edge $(v_1, v_2) \in E$. Continue collecting vertices $v_1, v_2, \ldots, v_k$ in this manner until we have found a vertex that has outdegree 0. We will not at any point find a repeat vertex, because G is promised to be acyclic. Thus, k is at most $|V|$, so we are guaranteed to find a vertex with outdegree 0. $\square$

4 Formal Model of Markets

Here, we discuss a formal model of a market system. This system is known as an **Exchange Market**. In this system, there are two kinds of objects: **Traders** and **Goods**. There is not any explicit notion of money.

To model an Exchange Market, we first assume that there are n different Traders, denoted as $t_1, \ldots, t_n$, and m different Goods, denoted $g_1, \ldots, g_m$.

At the beginning of the model, each Trader must have *something*. The entire marketplace is considered to have one unit of each Good. The amount of each Good that each Trader possesses at the beginning is called the **initial endowment**, which is represented by the following "E-matrix":

$$\begin{array}{c|ccc} & g_1 & \cdots & g_m \\ \hline t_1 & e_{1,1} & \cdots & e_{1,m} \\ \vdots & \vdots & \ddots & \\ t_n & e_{n,1} & & e_{n,m} \end{array}$$

satisfying, for all $j \in \{1, \ldots, m\}$:

$$\sum_{i=1}^{m} e_{i,j} = 1$$

That is, Trader t_i posesses Good g_j in the quantity $e_{i,j}$. In this model, Goods are considered completely divisible, so $0 \leq e_{i,j} \leq 1$ may be a real number.

In order to completely modelthe Exchange Market, we need to capture each Trader's self-interest. This is modeled by the **utility function**. The utility function $u : \mathbb{R}^m \to \mathbb{R}^n$ can be viewed as a collection of functions from $g_1, \ldots, g_m$ to a scalar representing the *value* of those goods to a particular Trader. The ith component of u, denoted u_i, denotes the utility function of the ith Trader.

Together, u and E form the Exchange Market.

Some important questions arise from the Exchange Market model:

- What is a reasonable exchange?
- Does there exist a reasonable exchange?
- Can you compute the reasonable exchange efficiently?

Using the Exchange Market model, we can attempt to compute the **market equilibrium** from the initial endowment and utility function, using an algorithm of Leon Walras.

Input: E, $(u_1, \ldots, u_n)$

In order to compute the market equilibrium, we use the concept of the *virtual market*. In the virtual market, we assign a virtual price to each Good, even though the market itself does not contain money. Therefore, the first step is to compute $(p_1, \ldots, p_m)$, the price for each Good.

Phase 1: Everyone sells their Goods for the designated price and the ith Trader now has b_i of virtual money:

$$b_i = \sum_{j=1}^{m} e_{i,j} p_j$$

which can also be written as:

$$\vec{b} = \vec{e_i}^T \vec{p}$$

Phase 2: Everyone buys Goods using their virtual money. The ith Trader buys $\vec{x_i}$ Goods:

$$\vec{x_i} = \arg\max\{u_i(\vec{x}) | \vec{x}^T \vec{p} \leq b_i\}$$

subject to the constraint that supply must equal demand, represented by:

$$\sum_{i=1}^{n} \vec{x_i} \leq \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

5 Quiz 2

Suppose $G = (V, E)$ is a finite directed graph in which the in-degree of each vertex is at most 1 and the out-degree of each vertex is at most 1. Prove that $\#\{\text{vertex with in-degree} = 0\} + \#\{\text{vertex with out-degree} = 0\}$ is an even number.

Proof. Observe that $\#\{\text{vertex with in-degree} = 0\} + \#\{\text{vertex with in-degree} = 1\} = |V|$ and $\#\{\text{vertex with out-degree} = 0\} + \#\{\text{vertex with out-degree} = 1\} = |V|$.

Therefore, $\#\{\text{vertex with in-degree} = 0\} + \#\{\text{vertex with in-degree} = 1\} + \#\{\text{vertex with out-degree} = 0\} + \#\{\text{vertex with out-degree} = 1\} = 2|V|$

Therefore, $\#\{\text{vertex with in-degree} = 0\} + \#\{\text{vertex with out-degree} = 0\} = 2|V| - \#\{\text{vertex with in-degree} = 1\} - \#\{\text{vertex with out-degree} = 1\}$

Because each edge has one in-vertex and one out-vertex, $\#\{\text{vertex with in-degree} = 1\} = \#\{\text{vertex with out-degree} = 1\} = |E|$.

Therefore, $\#\{\text{vertex with in-degree} = 0\} + \#\{\text{vertex with out-degree} = 0\} = 2|V| - 2|E|$. Since $2|V| - 2|E|$ is even, it follows that $\#\{\text{vertex with in-degree} = 0\} + \#\{\text{vertex with out-degree} = 0\}$ is even.

$\square$