Algorithmic Game Theory

Abstract

When computers began their emergence some 50 years ago, they were merely standalone machines that could iterate basic computations a fixed number of times. Humanity began to tailor problems so that computers could compute answers, thus forming a language between the two. Algorithms that computers could understand began to be made and left computer scientists with the task of determining which algorithms were better than others with respect to running time and complexity. Independently, right around this time, game theory was beginning to take off with applications, most notably in economics. Game theory studies interactions between individuals, whether they are competing or cooperating. Who knew that in almost 50 years time these two seemingly independent entities would be forced together with the emergence of the Internet? The Internet was not simply made by one person but rather was the result of many people wanting to interact. Naturally, Game Theorists stepped in to understand the growing market of the Internet. Additionally, Computer Scientists wished to create new designs and algorithms with Internet applications. With this clash of disciplines, a hybrid subject was born, Algorithmic Game Theory (AGT). This paper will showcase the fundamental task of determining the complexity of finding Nash Equilibria, address algorithms that attempt to model how humans would interact with an uncertain environment and quantify inefficiency in equilibria when compared to some societal optimum.

Contents

1 Introduction

The study of Algorithmic Game Theory (AGT) lies within two seemingly different subjects: theoretical computer science and economics. The internet allowed for a new global market and a myriad of small markets where everything could be monitored, from clicks of a mouse to time spent on a webpage. Economists want to find where profit can be optimized, where the computer scientist aims to accomplish just that but would like to do it quickly. One important question within economics is finding Nash Equilibria of certain games or situations. Thinking of competing markets as a game with players, a Nash Equilibrium is a point in a game where there is no incentive for a player to switch strategy unilaterally when the other players stick to their own equilibrium strategy.

Definition 1 (Nash Equilibrium)**.** A Nash Equilibrium (NE) is any vector of n strategies $\boldsymbol{p^*} = (p_1^*, p_2^*, ..., p_n^*)$ where the payoff $u_j(\boldsymbol{p})$ for player j cannot be any greater when changing a strategy from one side against the other player's equilibrium strategies.

$$u_j(\boldsymbol{p^*}) = \max_{p_j} u_j(p_1^*, ..., , p_{j-1}^*, p_j, , p_{j+1}^* ..., p_n^*) \tag{1}$$

John Nash won the Nobel Prize in Economics in 1994 for his proof that any two person game with a finite number of pure strategies has at least one equilibrium pair. If the NE could always be found, why not develop a program that finds this point exactly and call it the solution of the game? The problem is that Nash used the Brouwer Fixed Point Theorem from Topology to prove this statement and it is known to be very difficult to find the Brouwer Fixed Point. Despite this difficult task, computer scientists were up for the challenge to find NE. If programs were able to find NE, it would mean that computers could model human interaction and find the optimal point for all players in a given game. Algorithms were made to find NE, like the Lemke-Howson (LH) Algorithm. Unfortunately the LH Algorithm uses the Simplex Method which is known to not run in polynomial time. To add to the difficulty, the LH algorithm only finds one NE from the zero point (all strategies represented but all being zero), but there may be several and one may give higher payoffs to some players without harming others.

It may be difficult to find NE without any intuition into the game, but players in a game typically converge rapidly to a NE. This begs the question, can algorithms be developed that simply model a typical person's natural game play to quickly arrive at the NE? This question is addressed in the minimizing regret and learning algorithms section.

However, letting players simply act selfishly may give rise to suboptimal solutions. The price of anarchy section will deal with finding ways to quantify what happens when players act selfishly versus when players are directed to some societal good. Specifically, routing games will be analyzed as a way to illuminate this subject by example.

In economics, Game Theory predicts an agent's equilibrium behavior with no regard to how that state is achieved. The way to arrive at NE is of prime interest to Computer Science and Complexity Theory. What algorithms can be made to model human behaviour? How hard is it to find NE? Is it really best to allow people to act selfishly? This area of research has exploded with interest in the past 10 years due mainly to the emergence of the Internet. This paper will highlight some of the main topics in this new field with some results as well.

2 Computational Complexity of Nash Equilibria

"If your laptop cannot find it, neither can the market" (Kamal Jain [8]).

Discovering how to computationally find NE may seem beside the point, since NE is a conceptual tool that describes how players in a game should rationally behave when put against competing rational players. So why try to model this human interaction? Setting algorithms to locate NE will provide sound credibility to predictions of individual's behavior. Within the market of the internet, if NE can be calculated efficiently, marketers can easily see where to focus attention and what prices to charge. Placing the rational behavior of humans in an algorithm would give rise to marketers simply placing the right parameters and variables in a program and seeing how players would react.

What would be the natural way for a rational player to play a game? Allowing a player to repeatedly make an improving move would perhaps be the most natural gambit [8]. In a mathematics formalism, let $\boldsymbol{p} = (p_j, i \in \{1, \cdots, n\})$ be a vector of responses or strategies, with j denoting a specific player out of n players and each p_j coming from a set of possible strategies for player j denoted P_j. Thus, the utility or payoff to player j with a specific strategy vector $\boldsymbol{p}$ is $u_j(\boldsymbol{p})$. When a player changes strategy and everyone else stays the same, it will be denoted as (p'_j, p_{-j}), where $p'_j \in P_j$.

Definition 2 (Best Response [8])**.** A strategy p'_j is a best response if

$$\arg\max_{p \in P_j} u_j(p, p_{-j}) = p'_j$$

For a pure NE, this would mean that one strategy is a best response. For a mixed NE, there may be a set of strategies that are best responses. Any mixed strategy that is a best response would then have all pure strategies that are included in the mixed strategy as best responses.

The intent of this section is to demonstrate that finding a NE is a computational problem where one must find the pure strategies that make up the set of best responses. If one can find the set of best responses, then the space where NE can occur is reduced and a NE would lie in the best response strategies. This task is known to be quite difficult to implement, but how hard is it?

2.1 Parity Argument

Before the complexity of finding NE can be calculated, perhaps it may be illuminating to see how Nash proved existence to see if this problem can be solved easily. Unfortunately, this effort is futile because Nash used the Brouwer Fixed Point Theorem and finding the Brouwer fixed point is known to be very difficult [7].

Because Nash's theorem guarantees a NE to every game where each player has a finite number of options to choose from, the problem of finding NE seems to be different than other problems in complexity theory. Recall that a problem is in $\mathscr{NP}$ if a solution to the problem can be verified in polynomial time. Further, a problem is in $\mathscr{NP}$-Complete if every problem in $\mathscr{NP}$ can be reduced to it. Thus, problems in the $\mathscr{NP}$-Complete class are the hardest form of problems and if one problem in this class can be solved in polynomial time, then the Millennium Prize Problem $\mathscr{NP} = \mathscr{P}$ is solved, where $\mathscr{P}$ is the class of problems that can be solved in polynomial time. Let the problem of NASH be the computational problem of finding a NE in a given game. Assigning this problem to $\mathscr{NP}$ would not be appropriate because it is not a decision problem to begin with. Further, the question, does a NE exist, is trivial by Nash's Theorem.

Moving away from the existence quality of NE, NASH can be simplified into several decision problems that can be shown to be $\mathscr{NP}$-Complete [8]. Are there two or more NE? Is the NE for a given player greater than a certain value? Does a NE exist with a certain strategy in the set of best responses, etc.? Thus it seems that in order to solve the main problem NASH, one must find a different class of problems that NASH is a part of.

Introduce the class of problems –FP and FNP– in which an output can be more than binary. These may be defined as the class of problems of finding a y with a given x such that the binary function $P(x, y) = 1$ [4]. Let x be the input of a game and $P(x, y)$ be one if y is a NE and zero otherwise. Thus the problem of NASH is to find y. Whether or not y is a NE can be checked in polynomial time (linear in the number of strategies of a given player), thus NASH lies in FNP. Further, a problem is in $TFNP$ (for Total Function) if and only if the problem is in FNP, and there is guaranteed to be at least one solution for it [4]. This makes NASH $\in TFNP$. It can also be shown that the problem of finding the Brouwer fixed point of a given function also lies in $TFNP$ [7]. It is obvious that $FP \subseteq TFNP \subseteq FNP$ (where FP is the function form of $\mathscr{P}$) but the relation cannot be reduced any more, as is the case with $\mathscr{NP}$ and $\mathscr{P}$.

However, the class $TFNP$ is typically used as a category of problems, because there is no known problem in which everything in $TFNP$ can be reduced to it. The complexity class that NASH falls into is an even more specialized class called $PPAD$ for Polynomial Parity Argument in a Directed Graph [4]. The parity argument states:

Any finite graph has an even number of odd degree nodes [7].

The parity argument can be demonstrated with the following problem in Graph Theory. In preparation, let $G = (V, E)$ be a given graph and $|V| = n$ and s be a path called the stick which is a sequence of edges $s = e_1, e_2, \ldots, e_m$ where the endpoints of e_i are the vertices v_i and v_{i+1}. Further denote $d(v)$ to be the degree of a vertex, and $\eta(v)$ to be the number of edges connecting v and $v_i, i \in \{1, \ldots, m\}$, i.e. every vertex that is a part of the stick except the last vertex.

Theorem 1 (Smith's Theorem [12])**.** *The number of Hamiltonian Paths in a graph $G = (V, E)$ beginning with stick s and ending in a vertex of the set $W = \{w \in V : d(w) - \eta(w)$ is even$\}$ is even.*

Proof. Let $s = e_1, e_2, \ldots, e_m$ be a stick in the graph G. Let $h = e_1, e_2, \ldots, e_{n-1}$ be a Hamiltonian path that begins with the stick s. Introduce edge e_n which is an edge connecting the last vertex of h and a vertex v_k where $k \geq m + 1$. Let $\tilde{L} = \{e_1, e_2, \ldots, e_n\}$, which is known as a lollipop of the graph. This set of edges contains two Hamiltonian paths (h and h') that begin with s; thus, $h = e_1, e_2, \ldots, e_{n-1}$ and $h' = e_1, e_2, \ldots, e_{k-1}, e_n, e_{n-1}, \ldots, e_{k+1}$.

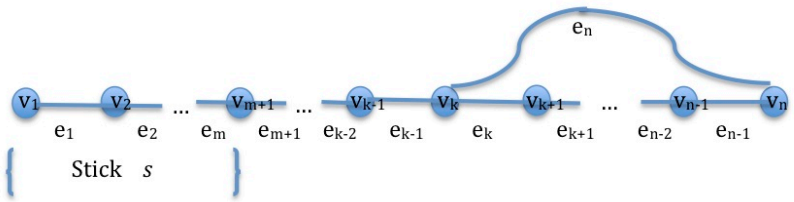

Figure 1: Two Hamiltonian paths beginning with the same stick s.

Construct the lollipop graph $\tilde{L}(G, s)$ which maps each Hamiltonian path h in G beginning with s to a vertex in $\tilde{L}(G, s)$. An edge exists between h_1 and h_2 if and only if they differ by a single edge in the original graph. To calculate the degree of a vertex h in $\tilde{L}(G, s)$ consider the degree of the last node $d(v_n)$ in a Hamiltonian path. The number of Hamiltonian paths that can be reached with the same initial stick s is then $d(v_n) - \eta(v_n)$. Thus subtracting one from this to eliminate the Hamiltonian path of h that makes the vertex in the lollipop graph, the result becomes $d(h) = d(v_n) - \eta(v_n) - 1$.

Each distinct Hamiltonian path is a new vertex in the lollipop graph. It can then be seen that every different vertex in the lollipop graph is a different lollipop of the original graph G. Thus in the set W in the hypothesis, this is exactly the set of vertices in $\tilde{L}(G, s)$ of odd degree. From the parity argument, the number of vertices in $\tilde{L}(G, s)$ of odd degree is even. It follows that the number of Hamiltonian paths is even [12]. □

2.2 The PPAD Complexity Class

With Smith's Theorem, a problem similar to NASH can be formed. Given a graph with odd degree and a Hamiltonian path of the graph, find another Hamiltonian path. This problem is known as SMITH and is similar to NASH in that it has a guaranteed solution. SMITH makes up a classic problem in the class known as PPA for Polynomial Parity Argument [7]. PPA is defined in terms of an algorithm which implicitly defines an exponentially large graph. At each step of a given algorithm, it starts from a vertex and calculates what vertex it has been to and what vertex it will travel to next in polynomial time. One does not need to take into account the difference between what vertices have already been visited and the vertex that will be visited in the next step. The desired solution is a graph that is symmetric, i.e. lacks direction, in the PPA case.

Remembering the Lemke-Howson (LH) Algorithm, a NE was found along a directed graph where the algorithm would calculate the next vertex in polynomial time and would finish when LH arrived at a vertex that had no successor. Consider the following statement which follows from the parity argument:

In any directed graph where each vertex has at most one edge leaving and one edge entering, if there is a source then there must be a sink [4].

Before a formal definition of $PPAD$ is attempted another problem must be defined. Given successor circuit S and predecessor circuit P each with the same number of input bits and output bits, say n, a directed graph can be constructed on the set of length n vectors, i.e. 2^n possible inputs. An edge exists from vertex x to y if and only if $S(x) = y$ and $P(y) = x$. Let the predecessor of the all zero vector be itself and the successor of it be something different, so that the algorithm moves away from zero. Consider the problem END-OF-LINE: find an input x such that $P(S(x)) \neq x$ or $S(P(x)) \neq x \neq$ all zero vector [4].

Definition 3 (PPAD [4])**.** $PPAD$ is the class of all search problems that can be polynomial-time reduced to END-OF-LINE.

Thus all problems in $PPAD$ involve finding the source or sink vertex of an exponentially large directed graph. It can be shown [8] that NASH can be reduced to END-OF-LINE and then END-OF-LINE reduces to it, so NASH is in $PPAD$-complete.

3 Learning Algorithms and Regret Minimization

Game theorists typically consider problems of making decisions repeatedly in an uncertain environment with uncontrollable or unforeseeable events. Because a player is allowed to continually play the game, one would want to learn from his/her past experiences and update information for how to play the game next time. This brings the idea of learning algorithms to the forefront, where players

learn from past events. This can then be placed in situations where many players compete and every player learns from past repetitions of game play. This may be the most natural way a rational person would play a game, and thus algorithms that can model this is of prime interest to the Algorithmic Game Theorist.

These algorithms can also be thought of as making repeated moves that aim to minimize regret [2]. An algorithm produces a regret when the difference between an algorithm's loss and the optimal decision's loss is positive. Regret analysis seeks to minimize this regret incurred by the algorithm [2]. Previous approaches for making adaptive algorithms are not suited for an environment in which malicious interactions are permissible. Learning algorithms have been known to exist in optimization theory, but they make the simplifying assumption that the feedback received is truthful and the environment is not updating with each implementation [2]. Introducing players and competition eliminates these simplifications and begs for a new area of research.

However, an introduction to preexisting algorithms would clarify the algorithms with game theoretical applications. Consider an algorithm H that updates at each time step t. From the list of m available actions $X = \{1, \ldots, m\}$, the algorithm chooses a probability distribution $\boldsymbol{p}^t = (p_i^t, i \in X)$ of the m actions at each t. Here, the full information model will be analyzed where the player knows the expected loss l_i^t at each time from the action he/she used [3]. These loss values can be thought of as percentages, or values restricted to $[0, 1]$. Thus, the expected loss at time step t from using algorithm H is $l_H^t = \sum_{i=1}^m p_i^t l_i^t$. The loss of using the i^{th} action during the first T time steps is given as $L_i^T = \sum_{t=1}^T l_i^t$ and the total loss of using algorithm H for the first T time steps is $L_H^T = \sum_{t=1}^T l_H^t$ [3].

When dealing with external regret, the algorithm designer compares the performance of the given algorithm with that of the best single action in hindsight. Let $\mathscr{G}$ be a class of algorithms, such that H is a part of it. The desired algorithm would minimize the regret of using any algorithm in $\mathscr{G}$. Let the regret of an algorithm H be $R_{\mathscr{G}} = L_H^T - \min_{g \in \mathscr{G}} L_g^T$. A popular choice for a class of algorithms is the set of using a single action, i.e. $\mathscr{G} = X$. Thus, with $L_{\min}^T = \min_{i \in X} L_i^T$, the regret can be written as $R^T = L_H^T - L_{\min}^T$ [3].

3.1 Greedy Algorithms

To simplify things a bit, let the losses be restricted to the values zero or one and not any real number between. In order to understand the goal of this section, it will be interesting to see how a naive first attempt at these algorithms may turn out. The `GREEDY` alogorithm selects the action that has lowest total loss at the previous time step, i.e. the action $x^t = \arg\min_{i \in X} L_i^{t-1}$ [3]. To break ties, let the action with lowest index be chosen if many actions have the same minimum loss [3].

`GREEDY` Algorithm:

Initiate: $x^1 = 1$

Increment: $L^t_{\min} = \min_{i \in X} L^{t-1}_i$
To prevent ties let $M^{t-1} = \{i : L^{t-1}_i = L^{t-1}_{\min}\}$
$x^t = \min M^{t-1}$

Example 1 (Greedy)**.** To see what the worst case is for this algorithm, consider the following example: Let $X = \{1, 2, 3, 4, 5\}$ and let each time step loss for each action be given as:

$$\begin{array}{c} \\ t=1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} x=1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

The `GREEDY` Algorithm begins at $x = 1$ and then incurs a loss of one and then moves to $x = 2$ because this is the lowest indexed value that has the minimal loss of zero at that stage. Continuing in this manner, `GREEDY` incurs a loss of six in six time steps. Restricting to the comparison class to be that which sticks with a single action at every iteration, the optimal loss would be one and can be achieved by using any $x = 2, 3, 4,$ or 5 at every iteration in this example.

A pattern can be seen in this simple example that gives an upper bound for the the resulting loss when using the `GREEDY` algorithm. At each time step, the worst `GREEDY` can do is gain a loss of one and a single strategy would not increase, or has loss of zero at the time step. The set M^t then decreases by at least one because the action chosen has increased from the minimum loss value. The algorithm can continue to gain a loss of one without the optimal loss increasing m times (the number of actions the player has available to him/her). As soon as the last action increases, the optimal loss increases. If the above example continues in the similar fashion at time step nine, action $x = 5$ will have not increased but the loss taken by `GREEDY` will have increased to nine. Beginning with a loss of zero, each action except one can then be used before the loss of the optimal single action increases by one. Thus, the following theorem can be stated.

Theorem 2. *The loss of the `GREEDY` algorithm for a model with m actions after T time iterations is bounded above by*

$$L^T_{\texttt{GREEDY}} \leq m \cdot L^T_{min} + (m-1) = \mathcal{O}(m) \text{[3].}$$

The order m loss mainly comes from the naive way the algorithm selects the next action by concentrating on a single action at each time step. Instead of choosing the action that has minimal loss with the lowest index, introduce a uniform distribution over all actions of minimal loss. This modification produces

a significant improvement to the GREEDY Algorithm. This algorithm will be termed RANDOMIZED GREEDY Algorithm [3].

RANDOMIZED GREEDY (RG) Algorithm	
Initialize:	$p_i^1 = 1/m$ for $i \in X$
Increment:	Similar to GREEDY, use $L_{\min}^{t-1}$ and M^{t-1} as defined above Let $p_i^t = 1/\lvert M^{t-1}\rvert$ for $i \in M^{t-1}$ and $p_i^t = 0$ otherwise.

In order to place a similar worst case bound on this algorithm, one must analyze the size of M^t at each time step because this effects the probability distribution that is used to choose the actions.

Theorem 3. *The loss for the RG algorithm after T time steps for a model with m actions is bounded above by*

$$L_{RG}^T \leq \log m + (1 + \log m) L_{min}^T = \mathcal{O}(\log m) \text{[3]}.$$

Proof. Consider the time steps between when $L_{\min}^T$ goes from c to $c+1$. Let t and $t+1$ be in this range. The size of M^t can decrease by an amount k in a single time step, say t to $t+1$. Let the $|M^t| = \mu$. Looking at the loss attributed to RG at t,

$$L_{\text{RG}}^t = \sum_{i=1}^{m} p_i^t l_i^t = \frac{1}{\mu} \sum_{i \in M^t} l_i^t = \frac{k}{\mu}$$

The last equality follows from the k actions increasing from the minimum loss, thus leaving M^t. This last term can be bounded by letting the set M^t decrease by one at k time increments

$$\frac{k}{\mu} \leq \frac{1}{\mu} + \frac{1}{\mu - 1} + \cdots + \frac{1}{\mu - k + 1}.$$

Now, how large can k be before the loss of the minimum increases? Worst case, k can be μ and μ can be as large as m (the number of actions available). Hence, within the time it takes for the minimum loss to increase by one, the increased loss of RG can be written as a truncated summation of $\log m$

$$\frac{1}{m} + \frac{1}{m-1} + \cdots + \frac{1}{1} \leq \log m + 1.$$

However, this is a bound on the loss of RG in a unit increase of the minimal loss. Thus, over the entire time the RG algorithm will increase at worst by $\log m$ before the minimal loss increases by one, but this can happen each time the minimal increases. Thus, the desired inequality is formed [3]. □

Note that this simple modification to the GREEDY algorithm has made a significant performance improvement.

3.2 Learning Algorithms Applied to Game Theory

The problem becomes, how do these regret minimizing algorithms interact with similar algorithms? To model games, consider a set of n competitors and player j has a set S_j of m actions for $j \in \{1, \ldots, n\}$. Consider a function that takes into account the action of each player and maps this set of actions to a single value in $[0, 1]$. More precisely, the loss function for player j is defined as $f_j : S_j \times S_{-j} \to [0, 1]$ where S_{-j} is the cartesian product of every player's action other than j [3]. At time t consider the vector of actions for each player $\boldsymbol{s}^t = (s_j^t, \text{ for } s_j^t \in S_j, j \in \{1, \ldots, n\})$. Thus, $f_j(\boldsymbol{s}^t) \in [0, 1]$.

The model must allow each player to randomize his/her actions according to a probability distribution. Let p_{ji}^t be the probability player j uses action i at time t [3]. Thus, player j chooses an action $s_j^t \in S_j$ according to the distribution $\boldsymbol{p}_j^t = (p_{ji}^t, i \in 1, \ldots, m)$ [3].

Without loss of generality, keep everything relative to player j. The loss to player j at t by using the algorithm H is $l_H^t = \mathbb{E}(f_j^t(\boldsymbol{s}^t)) = \sum_{s \in S_j} f_j(s, s_{-j}^t) p_{j,s}^t$ where each action is distributed over the probability distribution $\boldsymbol{p}_j^t$ [3]. Let l_k^t be the loss of using action k where $s_k^t \in S_j$ (could be the same as the one picked by H). Write the loss to player j for using action k at time t as $l_k^t = \mathbb{E}(f_j^t(s_k^t, s_{-j}^t))$ where the j^{th} player switched action to s_k^t [3]. The total loss incurred by player j after T time increments while using algorithm H is given as $L_H^T = \sum_{t=1}^T l_H^t$ and for an arbitrary action k, $L_k^T = \sum_{t=1}^T l_k^t$ [3]. It is now possible to define regret to player j in the game theoretical framework, $R_H = L_H^T - \min_{k \in S_j} L_k^T$ [3].

The game theory model, which takes into account many players competing and updating strategies, can then be placed into the framework of learning algorithms. The full information model was used in the previous scenario not dealing with game theory, but this is not the way it should be modeled here. A player typically only knows the actions that they have done up to a certain time, not the losses that would have been incurred if different actions were taken. This rules out the `GREEDY` algorithm from before, but there are other algorithms that are applicably to the partial information model. These algorithms attempt to model competing individuals, each trying to minimize regret against other players. Many sources on this subject look at the Mulit-Armed-Bandit problem that considers someone pulling many levers such that each gives a reward and the user would like to use the one that gives the highest reward [1]. This problem can then be adjusted to allow the levers to act adversarially against the user to place this problem in the realm of game theory [1]. Furthermore, this problem can be placed in the context of routing where each route a player can take across a network has costs and the user wants to minimize the cost to travel across it. It is these problems that will be addressed in detail in the next section.

4 Quantifying Inefficiency

However, if everyone is allowed to only think about themselves, what happens to the global good? What can go wrong when everyone is allowed to act selfishly? Might individuals do better personally if the global good is taken into account?

Example 2 (Prisoner's Dilemma)**.** Consider the prisoner's dilemma, which can be represented in the following matrix form:

$$\begin{array}{c} \\ \text{Confess} \\ \text{Don't} \end{array} \begin{array}{c} \begin{array}{cc} \text{Confess} & \text{Don't} \end{array} \\ \begin{pmatrix} (1,1) & (10,0) \\ (0,10) & (5,5) \end{pmatrix} \end{array}$$

It is easy to see by inspection that the NE to this game is (confess, confess) which gives a payoff of 1 to each player. However, this is pareto inefficient because both players can do better by coordinating and both not confessing. This would increase the payoffs to both players, but it is not a NE. It is thus concluded that the NE for the prisoner's dilemma is inefficient [11].

The problem of inefficiency is not the same as finding a NE. Rather, this is more of an optimization problem that has an objective function which naturally describes the quantity that is being maximized, i.e. the payoff. When the optimal and NE values are similar, selfish behavior does not have severe consequences and having an overseer directing the game would not give significant benefit to anyone. This brings up the question, when are NE guaranteed to optimize these optimization problems?

A natural choice in measuring the inefficiency of equilibria would be the ratio between the objective function value and the equilibrium. However, this brings up several problems, including the optimal value is typically difficult to compute and there may be several equilibria. Among the most popular measures of inefficiency is the price of anarchy (POA).

Definition 4 (Price of Anarchy [11])**.** Let u_{worst} be the worst equilibrium value and let $\hat{u}$ be the optimal outcome.

$$POA = \frac{\text{Worst equilibrium value of the game}}{\text{Optimal outcome}} = \frac{u_{\text{worst}}}{\hat{u}} \tag{2}$$

What if a game has many equilibria that are close to the optimal outcome and only one that is much higher? To differentiate between those that have many equilibria far from the optimum and those that have many equilibria close to it, the measure of price of stability (POS) will be introduced.

Definition 5 (Price Of Stability [11])**.** Let u_i be the i^{th} equilibrium value of the game with L equilibria.

$$POS = \frac{1}{L} \sum_{i=1}^{L} \frac{u_i}{\hat{u}} \tag{3}$$

Note that when there is a unique equilibria, $POS = POA$. In some circumstances, it may be better to quantify inefficiency with POA, where POS would be better to use in others. Routing games will be used to demonstrate these quantities.

Example 3 (Pigou's Example [11])**.** Consider the Pigou's Example where there is a source and destination node, s and d, respectively, and there are two links 1 and 2 that connect s and d. Denote the flow across each link to be $y_i, i = 1, 2$. The cost of using link 1 is constant, say $c_1(y_1) = 1$, and the cost of using link 2 is linear, say $c_2(y_2) = y_2$.

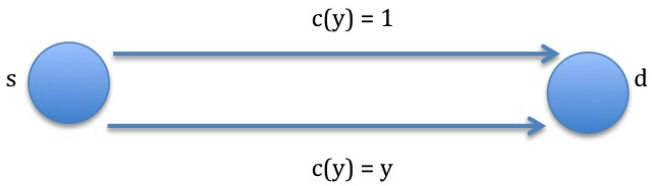

Figure 2: Pigou's Example

Suppose the problem is to push one unit of flow through this simple network with minimal cost. It is easy to see that an equilibrium to this network would be for everyone to take link 2, since if one deviates to the constant route, he/she does not improve the cost of 1. However, the cost on the link he/she previously occupied has decreased. Thus, the deviator would have incentive to move back to link 2.

The natural objective function can be written in the following optimization problem

$$\begin{aligned} \min_{y_1, y_2} \quad & \sum_{i=1}^{2} y_i c_i(y_i) = y_1 + y_2^2 \\ \text{subject to} \quad & y_1, y_2 > 0, \ \sum_{i=1}^{2} y_i = 1. \end{aligned}$$

It can easily be seen that the objective function is strictly convex and the feasible region is convex, so that Lagrangian methods can be used to find the optimal.

$$\begin{aligned} \mathscr{L}(\boldsymbol{y}, \lambda) &= y_1 + y_2^2 - \lambda(y_1 + y_2 - 1) \\ \frac{\partial \mathscr{L}}{\partial y_1} &= 1 - \hat{\lambda} = 0 \Rightarrow \hat{\lambda} = 1 \\ \frac{\partial \mathscr{L}}{\partial y_2} &= 2y_2 - \hat{\lambda} = 0 \Rightarrow \hat{\lambda} = 2y_2 \end{aligned}$$

To satisfy both the equations for $\hat{\lambda}$, it forces $y_2 = \frac{1}{2}$. Further, to have a feasible solution, y_1 must also be $\frac{1}{2}$. Hence, the optimal average flow is $\frac{3}{4}$. It turns out, as will be shown in the next section, that both the POS and $POA = \frac{1}{3/4} = \frac{4}{3}$.

4.1 Nonatomic Selfish Routing

Although the Pigou's example seems very simple, it allows one to see more general features of routing problems. To understand that the POS and POA are the same (i.e. the average cost for the traffic is the same in every equilibrium) in the previous example, nonatomic selfish routing will be introduced. The term nonatomic is used because there is a large number of players in the network and each player's contribution is small compared to the whole system [10]. To generalize the Pigou's example, consider the following notation.

To begin, a network is defined on a graph with vertices and edges $G = (V, E)$ where $|E| = m$. This graph has several source-destination pairs, where in the simple example there was only one pair. Label the set of source-destination pairs as $S = \{(s_1, d_1), (s_2, d_2), \cdots, (s_n, d_n)\}$. For simplicity let $(s_j, d_j) = j$ for $j = 1, \cdots, n$. Each player in the network is concerned with one of these source-destination pairs which can also be termed as commodities. Suppose that the set of all routes for commodity $j \in S$ is R_j and the set of every route for every commodity is $R = \cup_{j=1}^{n} R_j$.

As in the previous example, a cost function $\boldsymbol{c} = (c_i, i \in E)$ must be assigned to each edge. These will most likely depend on the flow across an edge. For mathematical simplicity assume $\boldsymbol{c}$ is nonegative, continuously differentiable, and nondecreasing with respect to a flow across the edge. Denote the flow along route r as x_r with $\boldsymbol{x} = (x_r, r \in R)$. In order to see how much cost is incurred by using a particular route, the amount of flow that is being used on each edge must be considered. Introduce the edge-route incidence matrix $A_{i,r}$ for $i \in E$ and $r \in R$ where $A_{i,r} = 1$ if i is in the route r and zero otherwise [6]. Further, denote the total flow on each edge as $\boldsymbol{y} = (y_i, i = 1, \ldots, m)$. Summarizing this in matrix notation yields

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \tag{4}$$

The cost attributed to moving a commodity along a route $r \in R$ is

$$c_r(x_r) = \sum_{i=1}^{m} c_i(y_i) A_{i,r}, \text{ for } r \in R \tag{5}$$

where the relationship between x_r and y_i is given in (4). Typically, the network will be known and commodities will be allowed to take certain routes, thus the commodity-route incidence matrix $\boldsymbol{H}$ can be constructed with entries $H_{j,r}$ being one when commodity j can use route r and zero otherwise [6]. Let the flows for each commodity be $\boldsymbol{f} = (f_j, j = 1, \ldots, n$. Summarizing this in matrix form gives

$$\boldsymbol{H}\boldsymbol{x} = \boldsymbol{f}. \tag{6}$$

An instance of a nonatomic selfish routing game is of the form $(G, \boldsymbol{f}, \boldsymbol{c})$ [10]. Thus to describe a nonatomic selfish routing game the following is needed:

the graph so that $\boldsymbol{H}$ and $\boldsymbol{A}$ can be constructed, the flows needed between every source destination pair and the cost function along every edge. With this information, the problem of finding the optimal flow for each commodity can be calculated.

Definition 6 (Wardrop Equilibrium [6, 10])**.** Let $\boldsymbol{x}$ be the feasible flow for a nonatomic instance $(G, \boldsymbol{f}, \boldsymbol{c})$. The flow $\boldsymbol{x}$ is a Wardrop equilibrium flow if for every commodity $j \in S$ and every route that has j as a commodity, say $r \in R_j$, then for $x_r > 0$

$$c_r(x_r) = \inf_{r' \in R_j} c_{r'}(x_r), \forall r \in R_j.$$

Thus, the flows are at equilibrium if the network cannot do any better by using a different route that connects the same source destination pair.

It can be shown that every nonatomic game has an equilibrium flow and every equilibrium flow gives the same cost along every edge. In order to prove this, the objective function method must be explained. Consider the societal cost function $zc_i(z)$. This function describes the cost that society must bear when z people use an edge that costs $c_i(z)$ for each person to use. Assuming $c_i(z)$ is convex and continuously differentiable, it is known that multiplying by z does not change these conditions. Denote $c_i^*(z) = \frac{d}{dz}\left(z \times c_i(z)\right)$ as the marginal cost function [10]. Let $\boldsymbol{x}$ be a feasible flow for the instance.

$$c_r^*(x_r) = \sum_{i \in r} c_i^*(y_i) = \sum_{i \in r} c_i(y_i) + \sum_{i \in r} y_i \frac{d}{dy_i}\left(c_i(y_i)\right)$$

$$= c_r(x_r) + \sum_{i \in r} y_i \frac{d}{dy_i}\left(c_i(y_i)\right)$$

It can be seen from the convexity of $\boldsymbol{c}$ that $\hat{\boldsymbol{x}}$ is an optimal flow for the instance if and only if for every commodity $j \in \{1, \ldots, n\}$ and every pair of routes with the same source destination pair $r, r' \in R_j$,

$$c_r^*(\hat{x}_r) \leq c_{r'}^*(\hat{x}_r)$$

With the instance $(G, \boldsymbol{f}, \boldsymbol{c})$ and the optimal function $y_i c_i(y_i)$ that is convex and continuously differentiable, $\hat{\boldsymbol{x}}$ is an optimal flow for the instance if and only if it is an equilibrium flow for $(G, \boldsymbol{f}, \boldsymbol{c^*})$, where $\boldsymbol{c^*}$ is the vector of marginal cost functions for each edge [10].

Noticing this, simply invert the previous result so that an equilibrium flow can be found for an optimal function whose derivative is the cost function $c_i(y_i)$ that is desired. Choose for the optimal function $\int_0^{y_i} c_i(u)du$ [10]. This implies that a flow $\hat{\boldsymbol{x}}$ is an optimal flow for the new objective function if and only if it is an equilibrium flow for $(G, \boldsymbol{f}, \boldsymbol{c})$. For this optimal function, it need only be required that c_i is nondecreasing and continuously differentiable since this makes $\frac{d}{dz}c_i(z) \geq 0$ which forces $\int_0^z c_i(u)du$ to be convex. Thus, minimizing a convex function over a convex set means that the optimal flow $\hat{\boldsymbol{x}}$ of this objective function is an equilibrium flow for the instance $(G, \boldsymbol{f}, \boldsymbol{c})$ which is the original

instance of the nonatomic selfish routing game. Knowing the desired objective function form, simply sum over every edge to get the network objective function

$$\phi(\boldsymbol{x}) = \sum_{i=1}^{m} \int_0^{y_i} c_i(u)du \tag{7}$$

Hence, a feasible flow for the instance $(G, \boldsymbol{f}, \boldsymbol{c})$ is an equilibrium flow if and only if it is at an absolute minimum of the objective function given in (7).

Theorem 4. *Let $(G, \boldsymbol{f}, \boldsymbol{c})$ be a nonatomic instance, then there is at least one equilibrium flow $\boldsymbol{x}$. Further, if there is more than one equilibrium flow for the instance then the cost on every edge is the same for every equilibrium flow [10].*

Proof. The proof technique here will follow from the objective function method outlined above where the equilibrium is made to be the optimal value of some objective function. It is known that the set of feasible flows of $(G, \boldsymbol{f}, \boldsymbol{c})$ make up a compact subset of a $|R|$ - dimensional Euclidean space where R is the set of all possible routes in the network. Further, the cost function is continuous, which makes the objective function in (7) a continuous function over this feasible set. Hence, ϕ is convex and achieves a minimum value on this feasible set and every minimum corresponds to an equilibrium flow of $(G, \boldsymbol{f}, \boldsymbol{c})$, as shown above.

Consider two different equilibria flows, say $\boldsymbol{x}$ and $\boldsymbol{x'}$ for the same instance. Therefore these two equilibria flows minimize ϕ. Consider convex combinations of our equilibria flows with $\lambda \in [0, 1]$.

$$\lambda^T \boldsymbol{x} + (1 - \lambda^T)\boldsymbol{x'} \tag{8}$$

With a convex feasible set, these vectors are also feasible flows. Knowing ϕ is convex, the following inequality can be made:

$$\phi(\boldsymbol{\lambda^T x + (1 - \lambda^T)x'}) \leq \lambda\phi(\boldsymbol{x}) + (1 - \lambda)\phi(\boldsymbol{x'}). \tag{9}$$

Because $\boldsymbol{x}$ and $\boldsymbol{x'}$ are global minima of ϕ, the inequality in (9) must hold with equality for every convex combination. Therefore, the result is a linear combination of optimal functions which means $\int_0^{y_i} c_i(u)du$ is linear between the two equilibria values, which makes every cost function constant between any two equilibria [10]. □

This shows that POS and POA are the same in nonatomic selfish routing games. In order to prove that equilibrium flows exist and give unique costs along edges for $(G, \boldsymbol{f}, \boldsymbol{c})$, a very unnatural choice of objective function given in (7) had to be introduced. This brings up questions such as, what is the natural choice of objective function, and how are the natural objective function and the objective function that gives equilibria different?

Example 4 (Braess's Paradox [6])**.** Consider a four node network with two disjoint routes from s to d given in Figure 3. Suppose there are six units of traffic that need to move across the network. The flow from s to d is $\boldsymbol{f} = (6)$,

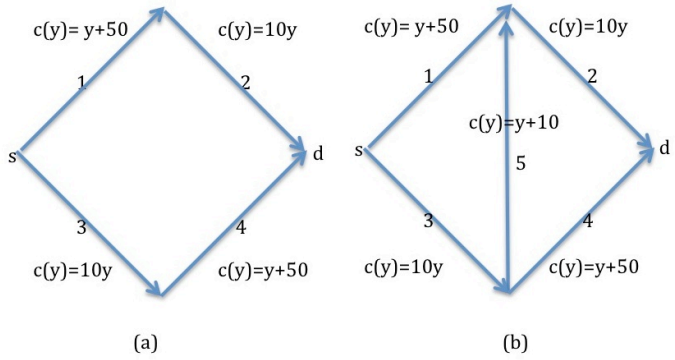

Figure 3: Braess's Paradox with (a) the original network and cost functions and (b) the adjusted network with an added edge and cost.

the commodity-route incidence matrix is $\boldsymbol{H} = (1, 1)$, the edge cost vector is $\boldsymbol{c}^T = (y_1 + 50, 10y_2, 10y_3, y_4 + 50)$, and the edge-route incidence matrix $\boldsymbol{A}$ is:

$$\boldsymbol{A} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cc} 1 \to 2 & 3 \to 4 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \end{array}$$

In the equilibrium flow, the traffic is split evenly between the two routes, so that there is no incentive to switch routes and both routes have the same cost of 83. Adding an edge of small cost to the network does something surprising. Denote the routes $r1$ = route using edges $1 \to 2$, $r2 = 3 \to 4$, and $r3 = 3 \to 5 \to 2$. After a little thought, it can be seen that the equilibrium flow $\boldsymbol{x} = (x_{r1} = 2, x_{r2} = 2, x_{r3} = 2)$. Note that if anyone deviates, the traffic flow decreases along the original route and so the person who changed would revert back. However, now the total cost is 92 and the same for any route.

Paradoxically, simply adding a road with small cost makes everyone's travel time greater. The optimal flow is the equilibrium flow of the original network, which gives POA = 92/83 = 1.1084. What goes wrong here? What else needs to be taken into account? Begin by looking at the natural choice of objective function (as used in the Pigou's Example)

$$C(\boldsymbol{x}) = \sum_{i=1}^{m} y_i c_i(y_i). \tag{10}$$

This leads to the following optimization problem:

$$\begin{array}{ll} \min & C(\boldsymbol{x}) \\ \text{subject to} & x_r \geq 0 \text{ for } r \in R, \boldsymbol{y} \in \mathbb{R}^m \\ & \boldsymbol{Ax} = \boldsymbol{y}, \boldsymbol{Hx} = \boldsymbol{f} \end{array}$$

With the objective function guaranteed to be convex and the feasible region being convex, Lagrangian techniques can be used to find the optimal.

$$\mathscr{L}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{i=1}^{m} y_i c_i(y_i) + \boldsymbol{\lambda}^T(\boldsymbol{f} - \boldsymbol{Hx}) - \boldsymbol{\mu}^T(\boldsymbol{y} - \boldsymbol{Ax}) \tag{11}$$

where $\boldsymbol{\lambda}$ is n-dimensional (same as the number of commodities), and $\boldsymbol{\mu}$ is m-dimensional (same as the number of edges). Further let j be the commodity that uses route r.

$$\frac{\partial \mathscr{L}}{\partial x_r} = -\lambda_j + \sum_{i=1}^{m} \mu_i A_{i,r}$$

$$\frac{\partial \mathscr{L}}{\partial y_i} = c_i(y_i) + y_i \frac{d}{dy_i} c_i(y_i) - \mu_i$$

Maximizing over all real values for $\boldsymbol{y}$, no boundary is encountered and a minimum can be found.

$$\hat{\mu}_i = c_i(y_i) + y_i \frac{d}{dy_i} c_i(y_i) \tag{12}$$

However, being limited to positive x_r, either the derivative is zero away from x_r being zero or it is increasing or zero on the boundary.

$$\hat{\lambda}_{r_j} \begin{cases} = & \sum_{i=1}^{m} \hat{\mu}_i A_{i,r} : x_r > 0 \\ \leq & \sum_{i=1}^{m} \hat{\mu}_i A_{i,r} : x_r = 0 \end{cases} \tag{13}$$

This is almost the desired values for the optimal but there is an extra $y_i \frac{d}{dy_i} c_i(y_i)$ term in (12). This term can be interpreted as the congestion toll that the users on edge or link i have to pay [6]. Therefore the total cost, which is the delay along the road and the toll paid to use the road, is being minimized. This toll charge introduces an overseer to the game such that if he/she is allowed to intervene by imposing tolls, people will take the optimal routes of minimal total cost.

4.2 Atomic Selfish Routing

The distinction between atomic and nonatomic instances is that in the former, each commodity represents a single player who needs to send a large amount of traffic on a single route, whereas the latter models each person's contribution as being insignificant to the whole [10]. With this modification, it will be shown that Theorem 4 does not hold in its entirety. The instance is of the same form in both atomic and nonatomic versions, $(G, \boldsymbol{f}, \boldsymbol{c})$. Atomic instances can be thought of as finite simultaneous-move games where each player is a commodity since

the flow of each commodity is routed through a single route.

Here would be a good place to recall all the notation that was introduced last section. Recall that $i \in E$ the set of edges of the network, so $i \in \{1, \ldots, m\}$. Also, $j \in S$ the set of commodities, source-destination pairs, or sometimes referred to as players $j \in \{1, \ldots, n\}$. Further, there are the sets R which is the set of all routes that connect any source-destination pair and R_j which is the set of routes that player j can take. A particular route in R will be denoted as r and can be made more specific to be in R_j. Also, consider the edge-route incidence matrix $\boldsymbol{A}$ and the commodity-route incidence matrix $\boldsymbol{H}$.

$$\boldsymbol{y} = (y_i, i = 1, \ldots, m), \text{ the vector of flows through each edge}$$

$$\boldsymbol{x} = (x_r, r \in R), \text{ the vector of flows through every route}$$

$$\boldsymbol{c} = (c_i, i = 1, \ldots, m), \text{ the vector of edge cost functions}$$

$$C(\boldsymbol{x}) = \sum_{i=1}^{m} y_i c_i(y_i), \text{ total cost, i.e. what to minimize to find optimal flow}$$

$$c_r(x_r) = \sum_{i=1}^{m} c_i(y_i) A_{i,r}, \text{ cost incurred for using a route } r \in R$$

$$\phi(\boldsymbol{x}) = \sum_{i=1}^{m} \int_0^{y_i} c_i(u)du, \text{ the optimal function that gives equilibrium flows.}$$

Definition 7 (Atomic Equilibrium flow [10])**.** Let $\boldsymbol{x}$ be a feasible flow. The flow $\boldsymbol{x}$ is an equilibrium flow if every player $j \in \{1, \ldots, n\}$ and every pair $r, r' \in R_j$ of routes for player j with $x_r > 0$

$$c_r(x_r) \leq c_{r'}(x'_{r'})$$

where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are identical except $x'_r = 0$ and $x'_{r'} = f_j$. Thus, simply move all of the flow from one route to another that connects with the same source-destination pair as commodity j's.

Example 5 (AAE Example [10])**.** By demonstrating with an example, it can be shown that an atomic instance gives equilibria that do not have the same link costs, which makes the POS and POA not the same. Consider a network where players are trying to move a unit of commodities $j \in \{1, 2, 3, 4\}$ from their source s_j to their destination d_j, with costs given in Figure 4. There are two possible routes for each player, the direct route or the two edge route. This leads to two equilibria: the first where all traffic uses the direct route and the other where everyone takes the two edge route. Note that everyone must send all their traffic along one route in the atomic case. The first equilibrium incurs a total cost of 4. This is also the optimal value. The second equilibrium gives a cost of 3 to player 1 because player 3 shares one of his/her edges with linear cost. Similarly for 2 because 4 shares one of the max cost edges. Now players 2 and 4 each use an edge of zero weight, but have the shared edge with players 1 and 3, respectively. Thus the total cost in this equilibrium is 10. This gives a POA of $10/4 = 2.5$, which is much higher than in the nonatomic example.

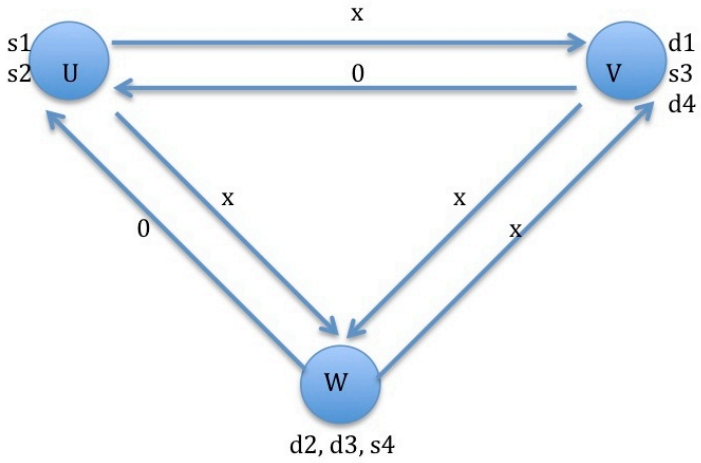

Figure 4: AAE example to demonstrate that different equilibrium flows have different costs.

With this simple example, two major differences were demonstrated between atomic and nonatomic routing. The first being that different equilibria give different costs, and the second being the POA can be huge in these games compared with their nonatomic counterparts. However, can equilibrium flows be shown to at least exist in the atomic instances? For this study, the simplifying assumption will be made that assumes all flows for every commodity is the same, i.e. $f_j = 1, \forall j \in \{1, \ldots, n\}$.

Theorem 5. *Let $(G, \boldsymbol{f}, \boldsymbol{c})$ be an atomic instance where the flow for each commodity is directed along a single route, $x_r = 1, \forall r \in R_j, \forall j \in \{1, \ldots, n\}$. Then an equilibrium flow $\boldsymbol{x}$ exists [10].*

Proof. Because atomic instances have a finite number of commodities and each has a finite number of strategies (sending whole traffic amounts down a single route) the optimal function in (7) must be discretized.

$$\tilde{\phi}(\boldsymbol{x}) = \sum_{i=1}^{m} \sum_{u=0}^{y_i} c_i(u) \tag{14}$$

Let $\boldsymbol{x}$ be a global minimum for $\tilde{\phi}(\boldsymbol{x})$. Assume $\boldsymbol{x}$ is not an equilibrium flow for the instance $(G, \boldsymbol{f}, \boldsymbol{c})$, so player j can decrease its cost by taking route r' instead of r for $r, r' \in R_j$ with new net flow $\boldsymbol{x'}$. For all edges in route r' that were in r (denoted by $r \cap r'$) the flow does not change, so the cost does not change along these edges. However, the flow changes along the edges that were not common, say $i \in r \backslash r'$ and $i \in r' \backslash r$ and thus the cost changes. In the objective function (14), adding a flow to the edges $i \in r' \backslash r$ yields the extra term $c_i(y_i + 1)$ with

an extra unit of flow, and for edges $i \in r \backslash r'$ the term $c_i(y_i)$ is lost.

$$0 > c_{r'}(x_{r'}) - c_r(x_r) = \sum_{i \in r' \backslash r} c_i(y_i + 1) - \sum_{i \in r \backslash r'} c_i(y_i) = \tilde{\phi}_(\boldsymbol{x'}) - \tilde{\phi}(\boldsymbol{x})$$

The last term has a contradiction because $\boldsymbol{x}$ was the minimum flow of the objective function. Thus, the equilibrium flow is the flow that minimizes the objective function given in (14). □

In order to prove this for differing flows for each player, the objective function needs to change, but the same proof technique is used. Despite the proof that equilibrium flows exist, the equilibrium flows do not give the same costs thus leading to $POS \neq POA$, as was seen by example.

4.3 Bounds on Price of Anarchy

The POA gives a measure of how far off the cost is between individuals acting selfishly in a network game and having an overseer directing traffic to get optimal flow. It would be of particular interest to know how far NE can be from the optimal. Again, the nonatomic and atomic cases will be treated seperately.

4.3.1 Nonatomic Bounds

Example 6 (Nonlinear Pigou [10])**.** Consider a variant to the Pigou example in Figure 5 where cost on the second edge is not linear but $c_2(y) = y^q$ for large q.

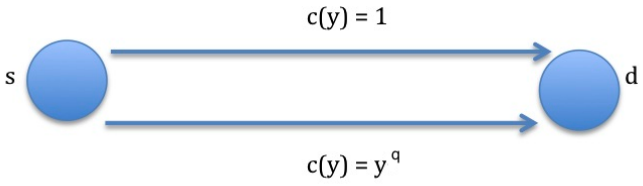

Figure 5: Pigou's Example with nonlinear cost function

The unique equilibria pushes the unit flow of traffic along the second edge, so that a unit total cost is incurred. However, the cost can be made smaller by sending a small fraction, say ϵ, on the constant cost edge and the rest on the nonlinear cost edge. This makes total cost $\sum_{i=1}^{2} y_i c_i(y_i) = \epsilon(1) + (1-\epsilon)(1-\epsilon)^q = \epsilon + (1-\epsilon)^{q+1}$. As q gets large and ϵ gets small, the cost can be made arbitrarily close to zero, which makes the POA grow without bound.

This example begs the question, can the POA be bounded in some cases and how big can it get? The relation between the objective function given in (7), which gives the equilibrium flows, and the true quantity that needs to be optimized (10), gives a particular bound on POA.

Theorem 6. *Let $(G, \boldsymbol{f}, \boldsymbol{c})$ be a nonatomic instance and suppose*

$$y_i c_i(y_i) \leq \gamma \cdot \int_0^{y_i} c_i(u)du, \forall i \in E \text{ and } y_i \geq 0.$$

Then the POA of the instance is no more than γ [10].

Proof. Allow $\boldsymbol{x}$ to be an equilibrium flow and let $\hat{\boldsymbol{x}}$ be the optimal flow for the instance $(G, \boldsymbol{f}, \boldsymbol{c})$. Take the hypothesis of the theorem and sum over all edges in the network.

$$C(\boldsymbol{x}) \leq \gamma \cdot \phi(\boldsymbol{x}) \leq \gamma \cdot \phi(\hat{\boldsymbol{x}}) \leq \gamma \cdot C(\hat{\boldsymbol{x}})$$

The second inequality holds because the equilibrium flow $\boldsymbol{x}$ minimizes the objective function ϕ and the last inequality holds because with any nondecreasing function, the following inequality can be made [10]

$$\int_0^{y_i} c_i(u)du \leq y_i c_i(y_i).$$

This gives the upper bound to POA

$$\gamma \geq \frac{C(\boldsymbol{x})}{C(\boldsymbol{x^*})} = POA. \tag{15}$$

□

Consider having polynomial cost functions where none of them have greater order than q. Note that $yc_i(y)$ increases the order of the polynomial by one and so does $\int_0^y c_i(u)du$ with a constant difference, $q+1$. Hence the POA can be no bigger than $q+1$ [10].

Theorem 7. *Let $\boldsymbol{x}$ be a feasible flow for the nonatomic instance $(G, \boldsymbol{f}, \boldsymbol{c})$. The flow $\boldsymbol{x}$ is an equilibrium flow if and only if for any other arbitrary feasible flow $\boldsymbol{x'}$ for the instance, the following holds [10]*

$$\sum_{i=1}^{m} c_i(y_i)y_i \leq \sum_{i=1}^{m} c_i(y_i)y_i'$$

where $\boldsymbol{Ax} = \boldsymbol{y}$ and $\boldsymbol{Ax'} = \boldsymbol{y'}$

Proof. Let there be commodities $j = 1, \ldots, n$ and edges $i = 1, \ldots, m$. Define a new function that is the same as the right hand side of the inequality in the theorem

$$B_{\boldsymbol{x}}(\boldsymbol{x'}) = \sum_{i=1}^{m} c_i(y_i)y_i'.$$

Note, that the definition for $B_{\boldsymbol{x}}$ can be written as another sum.

$$\sum_{j=1}^{n} \sum_{r:r\in R_j} c_r(x_r)x_r' = \sum_{r\in R} c_r(x_r)x_r' = \sum_{i=1}^{m} \sum_{r\in R: i\in r} c_i(y_i)x_r', \text{ by switching the sum}$$

$$= \sum_{i=1}^{m} c_i(y_i) \sum_{r \in R} A_{i,r} x'_r = \sum_{i=1}^{m} c_i(y_i) y'_i = B_{\boldsymbol{x}}(\boldsymbol{x}')$$

Because the equilibrium flow for a nonatomic instance has equal cost on every edge, the intuitive interpretation of $B_{\boldsymbol{x}}(\boldsymbol{x}')$ is the cost of the flow $\boldsymbol{x}'$ after the costs along every edge has been changed to be constant [10]. By the first definition of $B_{\boldsymbol{x}}$, the theorem can be restated as $\boldsymbol{x}$ is an equilibrium flow if and only if it minimizes $B_{\boldsymbol{x}}$ over all feasible flows.

Now, turn to the second definition of $B_{\boldsymbol{x}}$. In order to find the flow $\boldsymbol{x}'$ that minimizes $B_{\boldsymbol{x}}$, take the derivative.

$$\frac{\partial B_{\boldsymbol{x}}(\boldsymbol{x}')}{\partial x'_r} = c_r(x_r), \forall r \in R_j$$

Thus, it is clear to see that for any feasible flow $\boldsymbol{x}' > 0$, $c_r(x_r)$ needs to be as close to zero as possible, i.e. only the flows that minimize $c_r(x_r)$ for all $r \in R$. By definition of equilibrium flow, this condition is satisfied if and only if $\boldsymbol{x}$ is an equilibrium flow. Therefore, the flow $\boldsymbol{x}$ minimizes $B_{\boldsymbol{x}}$ if and only if it is an equilibrium flow, which satisfies the condition in the theorem. □

Although an upper bound has been found for POA, this depends on the costs being the same for every equilibrium along every edge. Thus, another upper bound to the POA in general multicommodity flow networks will be considered.

Definition 8 (Pigou Bound [10])**.** Let $\mathscr{C}$ by a set of cost functions. The Pigou Bound $\alpha(\mathscr{C})$ is given as

$$\alpha(\mathscr{C}) = \sup_{c \in \mathscr{C}} \sup_{z,p \geq 0} \frac{pc(p)}{zc(z) + (p-z)c(p)}$$

and by definition $\frac{0}{0} = 1$.

Theorem 8. *Consider a known set of cost functions $\mathscr{C}$ and the Pigou bound $\alpha(\mathscr{C})$ for $\mathscr{C}$. If $(G, \boldsymbol{f}, \boldsymbol{c})$ is a nonatomic instance with cost functions $c_i \in \mathscr{C}$ for $i \in E$ then the POA of the instance is bounded above by $\alpha(\mathscr{C})$ [10].*

Proof. Begin this proof as has been done with many of the others by assuming an equilibrium flow $\boldsymbol{x}$ and an optimal flow $\hat{\boldsymbol{x}}$ for a nonatomic instance. With cost functions in $\mathscr{C}$,

$$C(\boldsymbol{x}') = \sum_{i=1}^{m} c_i(y'_i) y'_i = \sum_{i=1}^{m} c_i(y_i) y_i \left(\frac{c_i(y'_i) y'_i + (y_i - y'_i) c(y_i) + (y'_i - y_i) c(y_i)}{c_i(y_i) y_i} \right)$$

$$= \sum_{i=1}^{m} c_i(y_i) y_i \left(\frac{c_i(y'_i) y'_i + (y_i - y'_i) c_i(y'_i))}{c_i(y_i) y_i} \right) + \sum_{i=1}^{m} (y_i - y'_i) c_i(y_i)$$

$$\geq \frac{1}{\alpha(\mathscr{C})} \sum_{i=1}^{m} c_i(y_i) y_i + \sum_{i=1}^{m} (y_i - y'_i) c_i(y_i)$$

This inequality follows from the definition of the Pigou Bound with $z = y_i'$ and $p = y_i$. From the last theorem, the last term in the last inequality is nonegative.

$$C(\boldsymbol{x'}) \geq \frac{C(\boldsymbol{x})}{\alpha(\mathscr{C})}$$

$$\Rightarrow \alpha(\mathscr{C}) \geq \frac{C(\boldsymbol{x})}{C\boldsymbol{x'})} = POA$$

□

Now what happens if more roads are added with nondecreasing, nonnegative affine costs to each route in the Braess's Paradox example? Can the POA get worse? Calculate the Pigou Bound for $c(z) = az + b$ for $a, b \geq 0$ [10].

$$\alpha(\mathscr{C}) = \sup_{z,p \geq 0} \frac{p(ap+b)}{z(az+b) + (p-z)(ap+b)}$$

Notice that the denominator is a quadratic in z, which can be minimized with respect to z.

$$\frac{\partial(z(az+b) + (p-z)(ap+b))}{\partial z} = 0 \Rightarrow z = p/2$$

Plugging this value for z into the expression above yields

$$\frac{ap+b}{(3/4)ap+b} \to \frac{4}{3} \text{ as } p \to \infty$$

Thus, the most the POA can be for nonatomic instances is far less than the POA found in the atomic case (5/2).

4.3.2 Atomic Bounds

The main difference when dealing with Atomic Bounds is that the objective function upper bound theorem cannot be used because costs are not the same for any equilibrium flow. The bound γ only works for the equilibrium flow that minimizes the objective function ϕ. This bound may be more useful in bounding the POS. In nonatomic instance POS and POA are the same, but in atomic instances they are usually different. Thus, to bound POS in the atomic case, the bounding technique given in Theorem 7 will be used.

The focus here will be on cost functions that are nondecreasing, nonnegative, and affine functions. Recall that in an atomic instance, when a player (or commodity) j changes routes, then the flow along the route that he/she was on becomes zero and the route that he/she switches to gets all the flow f_j.

Theorem 9. *Let $(G, \boldsymbol{f}, \boldsymbol{c})$ be an atomic instance where every cost function is affine $c_i(z) = az + b$, with $a, b \geq 0$. Let $\boldsymbol{x}$ be the equilibrium flow and $\hat{\boldsymbol{x}}$ be the optimal flow. Let player j use route r in $\boldsymbol{x}$ and r' in $\hat{\boldsymbol{x}}$ for $r, r' \in R_j$. Then,*

$$\sum_{i \in r}(a_i y_i + b_i) \leq \sum_{i \in r'}(a_i(y_i + f_j) + b_i) \text{ [10].} \tag{16}$$

Proof. The proof follows immediately from the definition of atomic equilibrium flows where the flow changes from $\boldsymbol{x}$ to $\hat{\boldsymbol{x}}$ by amount f_j along routes that were changed.

$$c_r(\boldsymbol{x}) \leq c_{r'}(\hat{\boldsymbol{x}}) \Rightarrow \sum_{i \in r}(a_i y_i + b_i) \leq \sum_{i \in r'}(a_i(y_i + f_j) + b_i)$$

□

This gives a way to relate the equilibrium flow with the optimal flow.

Theorem 10. *Using the same notation and assumptions as in the previous theorem,*

$$C(\boldsymbol{x}) \leq C(\boldsymbol{x'}) + \sum_{i=1}^{m} a_i y_i y_i' \text{ [10].} \tag{17}$$

Proof.

$$C(\boldsymbol{x}) = \sum_{r \in R} c_r(x_r)x_r = \sum_{j=1}^{n} \sum_{r \in R_j} c_r(x_r)x_r = \sum_{j=1}^{n} \sum_{r \in R_j} x_r \sum_{i \in r} c_i(y_i)$$

Recall that in the atomic case, the flow sent down any route is f_j and zero along any other route for player j.

$$= \sum_{j=1}^{n} f_j \sum_{i \in r : r \in R_j} c_i(y_i) = \sum_{j=1}^{n} f_j \sum_{i \in r : r \in R_j} (a_i y_i + b_i)$$

$$\leq \sum_{j=1}^{n} f_j \left(\sum_{i \in r' : r' \in R_j} a_i(y_i + f_j) + b_i \right)$$

This last inequality follows from the last theorem where all edges in the new route r' can be summed over with the new flow f_j added. Notice that the flow along a route f_j is no more than summing every edge's flow on the route (because other routes may be sharing edges).

$$\leq \sum_{j=1}^{n} f_j \left(\sum_{i \in r' : r' \in R_j} a_i(y_i + y_i') + b_i \right) = \sum_{i=1}^{m} \left((a_i(y_i + y_i') + b_i) \sum_{j : i \in r' : r' \in R_j} f_j \right)$$

$$= \sum_{i=1}^{m} ((a_i(y_i + y_i') + b_i)y_i') = \sum_{i=1}^{m} y_i'(a_i y_i' + b_i) + \sum_{i=1}^{m} a_i y_i y_i' = C(\boldsymbol{x'}) + \sum_{i=1}^{m} a_i y_i y_i'$$

□

With these results, the maximum value that the POA can take in atomic instances can be calculated. In order to prove the following theorem, recall the well known Cauchy-Schwarz inequality.

$$\left(\sum_u q_u z_u \right)^2 \leq \sum_u q_u^2 \sum_v z_v^2 \tag{18}$$

Theorem 11. *With an atomic instance* $(G, \boldsymbol{f}, \boldsymbol{c})$ *and affine cost functions, the POA is at most* $(3+\sqrt{5})/2$ *[10].*

Proof. Once again, let $\boldsymbol{x}$ denote the equilibrium flow and $\hat{\boldsymbol{x}}$ be the optimal flow for the atomic instance. Let the affine cost function be given as $c_i(z) = a_i z + b_i$ for $a_i, b_i \geq 0$. Apply the Cauchy-Schwarz inequality to the vectors $(\sqrt{a_i} y_i, i \in E)$ and $(\sqrt{a_i} y_i', i \in E)$ to put a bound on the last term in (17).

$$\sum_{i=1}^{m} a_i y_i y_i' \leq \sqrt{\sum_{i=1}^{m} a_i y_i^2 \sum_{k=1}^{m} a_k {y_k'}^2} \leq \sqrt{C(\boldsymbol{x})}\sqrt{C(\hat{\boldsymbol{x}})}$$

Where the last inequality follows because adding positive b_i to every term increases it. Hence, from (17)

$$C(\boldsymbol{x}) \leq C(\boldsymbol{x'}) + \sqrt{C(\boldsymbol{x})}\sqrt{C(\hat{\boldsymbol{x}})} \Rightarrow \frac{C(\boldsymbol{x})}{C(\hat{\boldsymbol{x}})} \leq 1 + \sqrt{\frac{C(\boldsymbol{x})}{C(\hat{\boldsymbol{x}})}}$$

Solving the equation $z - 1 \leq \sqrt{z}$, the quadratic $z^2 - 3z + 1 \leq 0$ can be formed. Solve for z which is the same as POA

$$POA = \frac{C(\boldsymbol{x})}{C(\hat{\boldsymbol{x}})} \leq \frac{3+\sqrt{5}}{2} \approx 2.618$$

□

5 Conclusion

This paper has shown some of the major problems and results in Algorithmic Game Theory: assigning the conceptual problem of finding equilibria to a complexity class, demonstrating and designing algorithms that may mimic players in a game, and quantifying how allowing players to act selfishly may be worse than the optimal behavior. The class of PPAD was defined and the parity argument was given. The problem of Nash equilibria turned out to be equivalent to the problem of finding the end node in an exponentially large graph. Some basic algorithms were shown to see how a program may model human behavior. With some already well known algorithms from optimization theory, it was demonstrated that these could be placed in a Game Theoretical context. Pigou's example and Braess's Paradox were introduced to show how anarchy in routing games can lead to everyone suffering. Bounds were then placed on this anarchy in nonatomic and atomic cases.

It was shown above that the reason the optimal and the equilibrium flows in a routing problem were not the same was because they came from different objective functions. The exact objective function introduced a new term that could be thought of as the toll to players imposed by some external player. When this is taken into account, every player can benefit. This can be seen in

real life situations where everyone wants to visit a certain place and because everyone acts selfishly, there is a large bottlenecking at the destination and people become frustrated. With an overseer that directs the traffic in different directions, everyone would benefit because the flow would be reduced along a single route. This notion of social welfare versus selfish behavior can also be generalized to auctions. Consider the second price auction where the bidder is incentivized to bid truthfully. Thus the player acting selfishly and bidding his/her truthful bid coincides with the social good that everyone tells the truth and the highest bid wins. Hence the optimal and the NE are the same in this case.

Current topics being addressed in AGT include trying to find NE exactly [9]. Classical algorithms require to look for a solution in a finite space and this may not have the exact NE of the game. Rather than limiting the solution space, the theory of computability should be extended. Thus the smallest extension of computation would be desired that would make finding NE sufficient [9]. The proof outline used to prove that NASH is the same as the problem of END-OF-LINE involves showing NASH is equivalent to finding a Brouwer fixed point, then discretizing to find fixed points on a grid and showing that small perturbations around a point that sum to zero implies that a fixed point is near. Because this approximate for the fixed point can be arbitrarily small, in two player games, finding this "ϵ-approximate NE" is the same as finding a real NE. However, in more than two player games, this approximate does not necessarily imply a NE. Hence, the research on finding NE exactly in multiplayer games is becoming ever more important.

AGT has also developed some interest from the social sciences. The underlying assumption to all of game theory is that the players are rational. However, it is typical for people to act irrationally, but how can this be modeled? This has brought up the idea of Bounded Rationality where perhaps people do not act rationally because they are not aware of the rational choice [5]. Perhaps the ignorant person does the most rational behavior in a proper subset of all his/her choices. Knowing this, learning algorithms could limit what is known to the player and the player would pick the best strategy from that set.

AGT has provided a fruitful area of research that breaches into economics, computer science, and mathematics. It will continue to grow in theories and results because the applications are so important in growing markets. The internet and computers are only becoming faster and more ubiquitous. Algorithms will become more sophisticated and come closer to finding ways to simulate, or perhaps emulate, human interaction. This subject truly aims to breach the reality and science fiction barrier.

References

[1] P. Auer. The non-stochastic multi-armed bandit problem. *Annual Symposium on Foundations of Computer Science*, 36(1):322–331, 1995.

[2] I. Avramopoulos, J. Rexford, and R. Schapire. From optimization to regret minimization and back again. *Proceedings of the Third Conference on Tackling Computer Systems Problems with Machine Learning Techniques*, 2008.

[3] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 79–101. Cambridge University Press, 2007.

[4] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. *Journal of Computer and System Sciences*, 2008.

[5] B. D. Jones. Bounded rationality. *Annual Review Political Science*, 2:297–331, 1995.

[6] F. Kelly. Stochastic networks part iii course, 2011.

[7] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

[8] C.H. Papadimitriou. The complexity of finding nash equilibria. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 29–51. Cambridge University Press, 2007.

[9] A. Pauly. How incomputable is finding nash equilibria. *Journal of Universal Computer Science*, 16(18):2686–2710, 2010.

[10] T. Roughgarden. Routing games. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 461–486. Cambridge University Press, 2007.

[11] T. Roughgarden and E. Tardos. Introduction to the inefficiency of equilibria. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 443–459. Cambridge University Press, 2007.

[12] A. G. Thomason. How incomputable is finding nash equilibria. *Annals of Discrete Mathematics*, 3:259–268, 1978.