Game Theory and Randomized Algorithms

Guy Aridor

Game theory is a set of tools that allow us to understand how decision-makers interact with each other. It has practical applications in economics, international relations, computer science, and many other disciplines. Though many game theoretic models utilize unrealistic assumptions at times, the intuition and results garnered from them usually give meaningful insights into the phenomena trying to be modelled. Different game theoretic models make different assumptions regarding the structure and knowledge of the decision makers in the game, but two core assumptions made in game theoretic models are that the decision-makers are rational and that they take into account the decisions made by the other decision-makers involved in the game (i.e. they are strategic). We say that a decision-maker is rational if he or she attempts to maximize their expected utility or payoff from the game.

In order to model the interactions between two decision-makers, we introduce the concept of a game. A game is a description of strategic interactions that the decision-makers (from here on, players) involved in the game can take as well as their preferences and interests. There are various different types of games, but on the most basic level there are two core types of games. The first, simultaneous games, are games where the players decide on the action they want to take (from here on, move) simultaneously. Due to this, the players cannot condition their move based on what the other player does, but rather what they believe that the other player will do. The second, sequential form games, are games where players move sequentially. In these types of games, players move one at a time (sequentially) and this means that players in the game can have different strategies conditioned on what they have observed that the other players have played. Sequential form games are typically represented as game trees where the branches in the tree represent strategies the user can play. This type of game provides the opportunity for varying degrees of information between players at different points in the game, unlike simultaneous games. When this happens, a player may not be sure what particular point in the game tree they are at when deciding what move to play. The concept used to describe and model these scenarios is an "information set" where, when modelling the game, we can group together the points of the game tree where a player will be unsure of which point in the game tree he or she is at. Using information sets we can actually represent simultaneous games using game trees rather easily. The standard representation of simultaneous form games, however, is in a table form called the normal form. This representation is usually much more convenient for solving for equilibrium states, so it is generally preferred. It is possible to represent sequential form games in normal form as well, but the size of the table will be exponential in the size of the game tree. This seemingly could cause problems when attempting to efficiently solve sequential form game equilibrium, but there are methods of dealing with these issues that I will not discuss here. For the rest of this essay, we will focus only on games in normal form.

In general, when we are given a set of player's potential strategies and preferences and we use this to construct our game, we want to be able to apply some techniques to understand the solution to this game. A solution to a game will give us a set of possible "steady state" moves that the players in the game will take given the potential strategies and preferences of the players. This is, of course, all assuming what we stated above as well as the fact that the preferences of the other players in the game are common knowledge. To make this more concrete, we will consider a classic example and from it derive a technique for looking for possible stable solutions to a game.

Suppose that there are two prisoners that have been charged with a crime and are in custody at a police station. They are put in separate rooms and are being interrogated. The interrogator in each room tells the prisoner in that room that they can confess and be free with no charge as long as they confess and serve as a witness against the other when they are released. If one of them confesses, then the prisoner that confessed will serve no time and the other prisoner will serve 4 years. If they both confess, they'll each face 3 years and if neither confesses they'll each serve only one year. Intuitively, one would think that the equilibrium behavior would be for both of them to not confess since we had assumed that our players are rational and therefore want to maximize their expected utility so it would make sense for both of them to each want to each only one year of prison. However, we have to remember that the players are strategic, so they need to consider the actions of other players in the game. Let us consider the game described above in normal form:

	Don't Confess	Confess
Don't Confess	(1, 1)	(4, 0)
Confess	(0, 4)	(3, 3)

The moves in the first row are the moves for player 1 and the moves in the first column are the moves for player 2. The first value in a payoff entry describes the payoff for player 1 and the second value in a payoff entry describes the payoff for player 2. For example, the entry in (Don't Confess, Confess) is (0, 4) which says that if Player 1 does not confess and Player 2 confesses then the payoff for player 1 is 0 (corresponding to 0 years in prison) and the payoff for player 2 is 4. It should now be clear that both players prefer to confess regardless of what the other player does. Remember that both players act simultaneously so neither player knows what the other player does. Since the payoffs are symmetric without loss of generality we'll consider only why player 1 strictly prefers confessing to not confessing. First we ask ourselves if it's better for player 1 to confess or not confess when player 2 doesn't confess (the same will be true for player 2 when considering the actions of player 1). We see that given that player 1 knows that player 2 is not going to confess, it is better for player 1 to confess since if he or she confesses then they will serve no time in prison, as opposed to one year. Likewise, if player 1 knows that player 2 is going to confess, if player 1 confesses as well then he or she will only serve 3 years in prison, as opposed to 4. As a result, regardless of what player 2 does, it is better for player 1 to confess and as a result player 1 will always confess. We call this type of strategy a dominant strategy because for every possible action of the other players, it is always better for this player to play a particular strategy. Likewise, we call the strategy that would never be used, a dominated strategy. This tool is useful in finding the equilibrium of games, especially when one considers the fact that we can iteratively delete dominated strategies until we reach a final result. However, it's clear that not every game we consider will be able to be solved this way. For instance, we can reason that it may be feasible that we have some sort of equilibrium even when there are no dominated or dominating strategies. A more general equilibrium which we will describe is the concept of a Nash Equilibrium. There are more general equilibrium concepts such as Nash Perfect Equilibrium, but we will not cover this here.

A Nash Equilibrium is a solution to a game that describes a profile of moves such that for each player i and a set of moves in the game, given that the rest of the players are playing their equilibrium moves, player i cannot play a different move and improve his or her payoff. In other words, given the actions of the other players in the game, no player can profitably deviate from his or her set of moves. Any arbitrary game is not guaranteed to have a Nash Equilibrium or even one unique Nash Equilibrium. An example of a game with no unique Nash Equilibrium is matching pennies, a game where one player picks heads or tails and wins if the other player doesn't correctly guess what he or she picked, but loses if the other player guesses what he or she picked. An example of a game with multiple Nash Equilibrium is the "Battle of the Sexes" where two players prefer to do something together but when picking between two possible things to do together, they have different preferences and cannot communicate with each other which results in multiple Nash Equilibrium. I will not go into detail about these games, but one can easily find out more information by looking them up as they are classic examples. We can, however, give the conditions necessary for a Nash Equilibrium to exist.

To give the conditions necessary for a Nash Equilibrium to exist, let us formalize what we have done previously a bit more. First, let us formalize the concept of a game. Each game consists of a finite set, N, representing the set of players, a non-empty set of possible actions A_i for each player (such as confess or not confess above), and a preference relation $\succeq_i$ for each player which is defined over the set of actions for that player. If all of the players have a finite set of actions, then the game is finite. These are all simply generalizations of what was discussed before. Let us also restate the definition of a Nash Equilibrium in terms of best-response functions. A best-response function gives the best possible response for a player given the strategies of all the other players in the game. Formally, $B_i(a_{-i}) = \{a_i \in A_i : (a_{-i}, a_i) \succeq_i (a_{-i}, a_i') \text{ for all } a_i' \in A_i\}$. The definition of Nash Equilibrium is then equivalent to $a_i^* \in B_i(a_{-i}^*)$ for all $i \in N$. This means that we find the best response function for each player and then find a profile of moves for which these moves are best responses to the equilibrium moves of all the other players. We will now use this definition to show some basic conditions under which we can state that a Nash Equilibrium exists.

We first will go over a few basic definitions. The first is what it means for a set to be compact. A compact set is a set where every subsequence converges to a point in the set. To understand this, consider the two following sets [0, 1] and [0, 1). The former is a compact set because every subsequence converges to some point in the set. This can shown using a theorem about compact sets in $\mathbf{R}^n$ that states that a set is compact in $\mathbf{R}^n$ if it is both bounded and closed.

Clearly, [0, 1] satisfies both and is in fact compact, whereas [0, 1) is not and therefore is not compact. We can see this because we can construct a subsequence that converges to 1, but 1 is not in the set so the set cannot be compact! Another important definition to go over is convexity of sets in Euclidean space. The definition of a convex set is that if we construct a line segment joining any pair of points of some set S, then if the line segment lies entirely in S, S is a convex set. We will also need to describe certain properties about the preference relation operator, $\succeq$. We will say that a preference relation $\succeq$ on $\mathbf{R}^n$ is quasi-concave if for every b $\in \mathbf{R}^n$ the set $\{a \in \mathbf{R}^n : a \succeq b\}$ is convex. In addition, a preference relation $\succeq$ on A is continuous if a $\succeq$ b whenever there are sequences $(a^k)_k$ and $(b^k)_k$ in A that converge to a and respectively for which $a^k \succeq b^k$ for all k. This means that if we have some sequence of bundles of goods (a^k) and each element of this sequence is at least as good as a bundle b then if this sequence converges to a then a is at least as good as b. Finally, we will need to understand fixed point theorems and in particular Kakutani's fixed point theorem (stated without proof) which we will utilize in our proof. Fixed point theorems in general say that under certain conditions there exists a point where f(x) = x. In our proof we will need a fixed point theorem so that we can show that there exists a value $a^* \in B(a^*)$ where B is a set-valued function $B(a) = \times_{i \in N} B_i(a_{-i})$. Kakutani's fixed point theorem goes as follows: Let X be a compact, convex subset of $\mathbf{R}^n$ and let $f : X \mapsto X$ be a set-valued function for which for all $x \in X$ the set f(x) is nonempty and convex and such that the graph of f is closed (this means that for all sequences $\{x_n\}\{y_n\}$ such that $y_n \in f(x_n)$ for all n, $x_n \mapsto x$, and $y_n \mapsto y$, we have $y \in f(x)$). If these conditions hold then there exists $x^* \in X$ such that $x^* \in f(x^*)$.

Using these we state the conditions necessary for a Nash Equilibrium to hold. A game has a Nash equilibrium if for all i $\in N$ the set A_i of actions of player is a nonempty, compact, and convex subset of a Euclidean space and the preference relation $\succeq_i$ is continuous and quasi-concave on A_i. The proof for this goes as follows. Suppose we define B as above where $B : A \mapsto A$ by $B(a) = \times_{i \in N} B_i(a_{-i})$. For every $i \in N$ the set $B_i(a_{-i})$ is nonempty since $\succeq_i$ is continuous. A_i is therefore compact and since $\succeq_i$ is quasi-concave on A_i, A_i is convex by the definition of quasi-concave. Based on our definition of continuous given above, B must have a closed graph since each $\succeq_i$ is continuous. Therefore, the conditions for Kakutani's theorem hold and B has a fixed point. From before, this must mean that there is an $a^* \in B(a^*)$ so this means that there exists a Nash Equilibrium!

However, there are some weaknesses to using this solution concept. Namely, we have to specify conditions for an equilibrium to exist and it will not always be the case that these conditions will hold. Ideally, we'd like to have a solution concept that will guarantee us a solution for a large variety of practical games. The solution concept that we defined and considered before required that players always play one particular move. What if we added some element of randomness to the moves by the player? In other words, what if our solution concept gave us a probability distribution across the strategies available to a player at a particular point in the game? In this case, we could view a player as randomly picking strategies based on the probability weights assigned to them by the distribution. In game theoretic terminology, this strategy is usually called a mixed strategy or a randomized strategy. There is some debate in the game theory community about what the actual meaning of a mixed strategy equilibrium is. At first glance, one would interpret the mixed strategy equilibrium as we did before where players randomly select a strategy because they are indifferent between all the "pure" strategies, but there are other possible interpretations. For instance, a classic example is the taxpayer audit example where the tax collector (say, the IRS) has to decide whether or not to audit taxpayers and taxpayers have to decide whether to pay or evade taxes. The classic analysis of this game shows that both parties have a mixed strategy equilibrium, but the reasoning behind mixed strategy equilibrium is supposed to be that both parties are indifferent between their pure strategies. However, we could argue that the IRS decides to only audit randomly because it cannot feasibly audit everyone but it wants to audit at least some people and wants to make sure that the taxpayers know that they are auditing so that everyone does not evade. This would not fit with the description given above. Nevertheless, adding the ability for players to utilize randomness in their strategy (regardless of how we interpret it), allows for us to be able to guarantee that for most games of practical interest, we can find an equilibrium.

Now that we have introduced the concept of mixed strategies we can introduce the following theorem: Every finite simultaneous game has a mixed strategy equilibrium. Suppose there are m possible strategies for player i where m is finite (by assumption). We can represent the set of mixed strategies for player i with a vector $(p_1, \ldots, p_m)$ where each $p_k \geq 0$ for all k and p_k represents the probability that player i uses the k^{th} strategy. Since this is a probability distribution, $\sum p_k = 1$. Since the set of actions is finite this means that the set of mixed strategies is nonempty, convex, and compact. In order to show that the preference relations must be continuous and quasi-concave we need to define the expected payoff under a mixed strategy game. We can define the support of the set of strategies A to be the strategies that will be used in the mixed strategy with non-zero probability. Given that the randomization of the strategy is done independently between the players, we can define the probability that of a certain action profile is simply the multiplication of the probabilities that a particular strategy will be used across all players and moves in the game. Therefore we can define the expected payoff under a mixed strategy as $U_i(b) = \sum_{a\in A} u_i(a)(\prod_{j\in N}(a_j)(b_j))$ where b is a particular mixed strategy, A is the set of possible actions for player i, and u is the utility function of player i. Note that U_i is multilinear (not to be proven here). Due to this fact, each player's expected payoff function is linear in the probabilities and therefore each player's preference relation is quasi-concave in his own strategy and is continuous. Therefore the conditions defined above are held and since nothing was assumed about the game besides that the set of actions for each player was finite, every finite simultaneous game must have a mixed strategy equilibrium. Note that if each player's payoff function is also quasi-concave in his own action then the game has a pure strategy Nash equilibrium! Essentially, adding mixed strategies allows us to transform the action set and the preference relations to satisfy the conditions noted before for a Nash Equilibrium to exist regardless of what was each player's action set and preference relation.

We have now seen how adding randomization to finite strategic games leads to us always having an equilibrium result. Next, we will see how we can use game theory to evaluate complexity bounds on randomized algorithms. In fact, doing so is simply a corollary to the theorem we just proved. Before showing this, we will first define a certain type of game, a zero-sum game. A zero-sum game is simply a game where the sum of the payoffs across all the players at the end of the game is 0. From this, it is clear to see that in these particular types of games, when we have two players their preferences will be diametrically opposed due to the fact that if some player has a positive payoff, the nature of the game requires that the other player (or players) have a negative payoff (a good example of this is the matching pennies game described earlier). This encourages a "maximin" strategy from both of the players. In equilibrium, both players use this "maximin" strategy where player i will choose the action that is best for him on the assumption that whatever he does, player j will choose her action to hurt him as much as possible.

In order to state and prove the corollary needed, let us briefly redefine our results from before. We can say that the expected payoff for player i from a mixed strategy game is $V_i(x, y) = \sum_{i=1..m} \sum_{j=1...n} x_i a_{i,j} y_j$ where $a_{i,j}$ represents an entry in the payoff matrix and x and y are mixed strategies. We will denote this simply by $V(x, y)$ because of the fact that our result holds for two-player zero-sum games so $V_1(x, y) = -V_2(x, y)$ and therefore we can simply have V(x, y) denote the "value" of the game since the absolute value of the expected payoffs will be the same. We can define an equilibrium point in a two-player zero-sum game as a point where $V(x^*, y^*) \geq V(x, y^*)$ for all $x \in X_m$ and $V(x^*, y) \geq V(x^*, y^*)$ for all $y \in Y_n$. This is equivalent to $max_{x \in X_m} V(x, y^*) = V(x^*, y^*) = min_{y \in Y_n} V(x^*, y)$. Suppose we use the result from the previous section that showed that an equilibrium mixed strategy pair must exist. This must mean that $v_b = min_{y \in Y_n} max_{x \in X_m} V(x, y) \leq max_{x \in X_m} V(x, y^*) = V(x^*, y^*) = min_{y \in Y_n} V(x^*, y) \leq max_{x \in X_m} min_{y \in Y_n} V(x, y) = v_a$ so $v_b \leq v_a$ but we know $v_a \leq v_b$ which means that $v_a = v_b$! Therefore, for a two-player zero-sum game we know the following is true: $max_{x \in X_m} min_{y \in Y_n} V(x, y) = min_{y \in Y_n} max_{x \in X_m} V(x, y)$. We will now be able to apply this to finding lower-bounds of complexity on randomized algorithms using Yao's principle.

Suppose that we have a problem P with a finite set X of inputs as well as a finite set of deterministic algorithms A for solving P. For each $a \in A$ and $x \in X$ we can define cost(a, x) as the cost incurred by the algorithm, which could be its space complexity or time complexity or any sort of measure of complexity. Suppose now we were to consider some randomized algorithm R to solve P. This randomized algorithm, in one particular run-through, would be equivalent to running one of the deterministic algorithms in A. However, the reason the algorithm is randomized is because it is a probability distribution over the set of deterministic algorithms. Therefore the randomized algorithm's expected cost is cost(R, x) $= \sum_{a \in A} Pr(a) * Cost(a, x)$. The worst case cost occurs when we choose the worst possible input (i.e. the one that maximizes the cost given the randomized algorithm), so the randomized complexity is defined to be $min_R max_{x \in X} cost(R, x)$ where min_R is the best possible randomized algorithm. We can also define the complexity with respect to the input distribution. For this, we can define the expected complexity cost of a deterministic algorithm d to be cost(d,D) $= \sum_{x \in X} Pr(x) * Cost(x, d)$. Under the distribution D, the best any deterministic algorithm can do is $min_{a \in A} cost(a, D)$ so we define the distributional complexity to be $max_D min_{a\in A} cost(a, D)$. To cast the problem in game theoretic terms, we can model this as a two-player zero-sum game where one player picks a deterministic algorithm and the other player picks the input and the resulting payoff is the cost of running the deterministic algorithm on the chosen input. To solve this game, we can use the theory we proved in the previous section. We know that an equilibrium must exist in this game and therefore this means that $max_D min_{a\in A} cost(a, D) = min_R max_{x\in X} cost(R, x)$. Dropping the leftmost terms on both sides gives us $min_{a\in A} cost(a, D) \leq max_{x\in X} cost(R, x)$ which gives us a lower-bound on randomized complexity! Now we can pick a distribution of the inputs and if we can prove that every deterministic algorithm incurs at least cost C then this means that the randomized complexity is at least C. Note, however, that Yao's principle is only valid for Las Vegas algorithms, not Monte Carlo algorithms.

Let us see how to apply this to give a lower bound on game tree evaluation. We define a binary game tree of height 2k where nodes in the tree at an even distance from the root are labeled AND and nodes at an odd distance from the root are labeled OR. The goal of the problem is to compute the value returned by the root. A deterministic algorithm for doing this is, in the worst case, at best n where is the total number of leaf nodes which, in this case, is $\Omega(4^k)$ since our tree is of height 2k. Consider an AND node whose children are leaf nodes. We can construct a tree such that the first child always considered by the deterministic algorithm returns a one so it always has to consider the second child. Extending this logic to every level of the tree, we can see how in the worst case the best we can do is to access all the leaf nodes, so our algorithm runs in time $\Omega(n)$.

Now, let's apply Yao's principle to the problem and see if we can give a lower bound on the expected cost of the algorithm. Remember that Yao's principle only applies to Las Vegas algorithms, so we will be giving a lower bound on the complexity for Las Vegas algorithms for solving this problem. First, we simply convert the AND-OR game trees to NOR trees. should give the same result (not proven here). Every node in the new NOR tree returns 1 if and only if both children return 0. This will give the same result as the AND-OR tree we considered before (not proven here). According to Yao's principle we need to describe a probability distribution on the inputs that leads to a high expected cost for any deterministic algorithm. A good probability distribution for this is setting each leaf of the tree to 1 with probability $r = (3-\sqrt{5})/2$. The probability that NOR returns a 1 is $(1-r)^2 =$

r. The expected cost C(h), where h is the height of the tree (or subtree), for evaluating the result at a node is minimal if we figure out the result of one part of the subtree before going to the other. In this case, if we evaluated the left subtree of a node and it evaluated to 0 then we would know that the final result would be 0 regardless of the value of the right subtree. The probability that both need to be evaluated is then at least (1-r) from our definition before. Due to this $C(h) \geq C(h-1) + (1-r)C(h-1)$. Simplified, $C(h) \geq (2-r)^h$. Given that the height of our tree is log (h) and substituting, we can end up with the lower bound of $n^{.694}$. Therefore, by Yao's Principle, any randomized Las Vegas algorithm can at best achieve an expected cost of $\Omega(n^{.694})$.

In this paper, we have developed some of the basic ideas of game theory as well as shown how utilizing randomness when attempting to find solutions to our games helps us always find a solution and that we can actually use the results shown here to give lower-bounds on any sort of randomized algorithms. While these results are powerful, they only cover a small subset of game theory and its applications to computer science.

References

[1] Martin Osborne and Ariel Rubinstein, *A Course in Game Theory*. Cambridge, Massachusetts, 1994.

[2] http://www.math.udel.edu/ angell/minimax.pdf

[3] http://www-i1.informatik.rwth-aachen.de/Lehre/SS07/VRA/Material/yao.pdf

[4] http://sublinear-course.wikispaces.com/file/view/lecture5.pdf