CS 599: Algorithmic Game Theory Sept 22, 2010

Lecture 5 - Complexity of Games

Professor Shang-Hua Teng *Scribes: Darryl DeWeese and John Kim*

The Most Famous Complexity Class

Each complexity class characterizes a particular phenomenon in computer science. In the case of NP, this phenomenon is the discrepancy between finding a proof and verifying a proof. The NP versus P question asks: if you can verify a proof efficiently, can you find one efficiently?

The reason for NP's fame is that it contains many of the problems that arise naturally in our lives. Technically, only decision problems, in which we must answer "yes" or "no" according to whether our input is a member of a particular set, are eligible for membership in NP. If it's easy to verify that the answer to a particular decision problem is "yes", then the problem is indeed in NP. On the other hand, if it's easy to verify that the answer is "no", then the problem is in a class called co-NP. Some problems can be in both classes, such as the primality problem. Since the invention of the AKS algorithm, we can determine in polynomial time whether a number is prime, making "yes" instances of the problem easy to verify. But the "no" instances are easy to verify as well, if we are provided with one of the factors of the number in question.

Another common problem in computer science is optimization, in which we seek to minimize or maximize an objective function of some input. Intuitively, optimization problems are harder than decision problems. It turns out that many interesting optimization problems can be rephrased as decision problems, and thus the umbrella of NP includes many optimization problems as well.

Algorithmic Game Theory Rolls Differently

But in algorithmic game theory, we frequently encounter a problem which is neither of these: the problem of finding an equilibrium. This falls under the category of search problems. An example is the network routing game, where we are given a graph $G = (V, E)$ and the discrete unit flows $f_1, f_2, ..., f_n$, each of which selfishly travels from a start node to a destination node. We seek paths $p_1, p_2, ..., p_n$ which properly route each unit flow such that they are in equilibrium.

This example happens to be a search problem over a total function, meaning that there is always a solution—it's just a matter of finding it. The search space for 3-SAT, on the other hand, is not over a total function, because the instance may not be satisfiable. From analysis of potential games in previous lectures, we know that finding an equilibrium for a potential game is always a search over a total function, since an equilibrium always exists.

In a 1991 paper [1], Nimrod Megiddo, a researcher at IBM, along with Christos Papadimitriou, showed that if there exists a search problem over a total function that is NP-complete, then NP = co-NP. But this seems unlikely, suggesting that we need a class other than NP to characterize the difficulty of total search.

A Closer Look at Network Routing

To begin our study of the difficulty of total search, let's go back to the network routing game we mentioned earlier. This time we add the notion that the game is symmetric, meaning that the strategy space for each player is the same, i.e. they all have the same start node and end node. Our input is a graph $G = (V, E, s, t, k, c_e(\cdot))$, where s and t are the start node and end node, k is the number of selfish unit flows routing themselves through the network, and $c_e(n_e(P))$ specifies a cost function for each edge which increases monotonically with the number of players using the edge, $n_e(P)$. Just as before, the desired output is a vector of paths $P = \langle p_1, p_2, ..., p_n \rangle$ which properly routes each unit flow such that they are in equilibrium.

In an earlier lecture, we proved that when the value of the cost function for each edge is equal to the number of players using that edge, best response dynamics will converge in polynomial time. But what about the relaxed case when the cost functions are merely monotonically increasing? Best response dynamics could take exponential time to converge due to the large infinity of real numbers. Regardless of the slow convergence of best response dynamics, can we still compute an equilibrium state in polynomial time?

Consider our potential function for this game,

$$\Phi(P) = \sum_e \left[c_e(1) + c_e(2) + ... + c_e(n_e(P)) \right]$$

If we can find a vector of paths P^* which globally minimizes Φ, then P^* will be in equilibrium, because the value of Φ improves as each player selfishly improves his path, and hence, if a player could improve, then Φ would not be at a global minimum. As luck would have it, we can minimize Φ via a simple reduction to min-cost flow: for each edge $e \in E$, replace e with k copies of itself, of costs $c_e(1), c_e(2), ..., c_e(k)$, each of capacity 1. Now calculate the min-cost flow for the k units.

Note that this scheme only finds an equilibrium that results in a globally minimal value of Φ. There are many other equilibriums at locally minimal values of Φ which it will not find. And though we achieved a polynomial algorithm as we originally set out to do, our victory will be short-lived, for we have only solved a very special case of the congestion game. In the general case, this game is not nearly so easy. Now it's time to finally introduce the new class of problems we've been hinting at.

Polynomial Local Search

To define the class of Polynomial Local Search problems, PLS, it helps to first define local optimization problems. And in order to define local optimization problems, it helps to define general optimization problems. So let an instance of a general optimization problem have a set F of feasible solutions and a cost function $c(f)$ for all $f \in F$. The goal is to find $f^* \in F$ with optimal cost $c(f^*)$. Given this definition, an instance of a local optimization problem is an instance of an optimization problem in which each $f \in F$ has a neighborhood $N(f) \subset F$. The goal here is to find a feasible solution which has cost no worse than its neighbors. Finally, an instance of a problem in PLS is an instance of a local optimization problem where, for all $f \in F$, $c(f)$ and $N(f)$ can be computed in polynomial time. A problem p in PLS is PLS-complete if there is a polynomial time reduction from all other problems in PLS to p, such that the local optima of p map to local optima of the reduced problem [2].

An example of a PLS-complete problem is Local Max Cut. The input of Local Max Cut is a graph $G = (V, W)$, where W is a set of weighted edges. For a given cut (A, B) on G, the neighborhood consists of cuts that can be achieved by moving one node from A to B or vice versa. Therefore each neighborhood is of size $n = |V|$. The goal is to find a cut with locally maximal weight.

Another PLS-complete problem is the Bisection Problem. The input is identical to Local Max Cut, except that n is even. Here, we desire a cut (A, B) with $|A| = |B|$. The neighborhood of such a cut is the set of cuts that can be obtained by swapping a node in A with one in B. Again, the goal is to maximize the value of the cut.

Yet another example of a PLS-complete problem is 3-SAT. Define the neighborhood of a variable assignment to be any assignment that can be obtained by flipping one variable. Then the goal is to find an assignment which locally maximizes the number of satisfied clauses.

Congestion Game Is PLS-complete

In order to prove this, we will show that an instance $G = (V, W)$ of Local Max Cut can be transformed into a congestion game in polynomial time, and that an equilibrium of this congestion game maps to a locally optimal max cut back on G. So for each edge $e \in W$, create two resources: r_e^A and r_e^B. Define a bijection between players and nodes, and define the strategy space of each player to contain two choices: use all the "A" resources provided by adjacent edges, or all the "B" resources provided by adjacent edges. Since each node has $n-1$ neighbors, this means that a player can choose $n-1$ "A" resources or $n-1$ "B" resources. Define the cost function as follows: if one player is using a resource, the cost is 0. If two players are using a resource, each pays a cost of w_e.

Consider a cut (A, B) on G and its relation to the choices of the players in the congestion game. There is an obvious bijection between cuts and choices: a node is in A if and only if the corresponding player chose to use the resources marked "A", mutatis mutandis for B. Now consider a player $u \in A$. He pays a total of $\sum_{v \in A} w(u, v)$, since he is sharing the "A" resources adjacent to him with the other players on the A side of the cut. Denote this cost $c_A(u)$. If he moved to the set B, he would pay $c_B(u) = \sum_{v \in B} w(u, v)$, which is the sum of the edges adjacent to him that are crossing the cut. In equilibrium, no player wants to switch sides, meaning for each player $u \in A$, $c_A(u) \leq c_B(u)$, mutatis mutandis for players in B. That is, a player switching sides would decrease the weight of the cut. Therefore on G, in the cut induced by this equilibrium, the weight of the cut is locally maximized.

Blogging Game

There are n bloggers, each of which has 1 unit of space he needs to fill with writing. The strategy space of a blogger is as follows: he can write something original, he can copy the writing of other bloggers, or some combination of these. Specifically, each blogger has a permutation of the indices 1 to n, which ranks how much he likes each blogger, himself included. If we denote the amount that blogger i will write as w_i, then $0 \leq w_i \leq 1$, and $1 - w_i$ is how much he will copy.

Suppose the best response dynamic is for each blogger to copy greedily from everyone on his ranked list, in order, until he reaches himself. If at this point, he hasn't totaled 1 unit of writing, he will write the difference himself. Then the bloggers are in equilibrium when no one wants to change his mind about how much he will write. Note that each blogger may only copy original writing, e.g. if blogger i writes 0.7 units and copies 0.3 units, then blogger j can only copy a maximum of 0.7 units from blogger i. Also note that the solution space, in general, is not convex.

The question is then: does this game always converge to equilibrium? If so, how fast?

(This type of game is called an ordinal game, in which player preferences are qualitative, as opposed to a cardinal game, in which preferences are quantitative.)

As an example, suppose there are four bloggers with the following permutations:

Player	Preference
1	2 3 4 1
2	3 4 1 2
3	4 1 2 3
4	1 2 3 4

In this scenario, each blogger prefers to copy every other blogger before writing anything himself. One of the more obvious equilibrium vectors is $\langle w_1, w_2, w_3, w_4 \rangle = \langle \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4} \rangle$. In general, a vector is an equilibrium in this example if and only if $w_1 + w_2 + w_3 + w_4 = 1$.

References

[1] N. Megiddo and C. H. Papadimitriou. A note on total functions, existence theorems and computational complexity. *Theoretical Computer Science* 81 (1991), pages 317-324.

[2] Nisan, Roughgarden, Tardos, and Vazirani, editors. *Algorithmic Game Theory.* Page 499. Cambridge University Press, 2007.