CS364A: Algorithmic Game Theory Lecture #19: Pure Nash Equilibria and PLS-Completeness*

Tim Roughgarden†

December 2, 2013

1 The Big Picture

We now have an impressive list of tractability results — polynomial-time algorithms and quickly converging learning dynamics — for several equilibrium concepts in several classes of games. Such tractability results, especially via reasonably natural learning processes, lend credibility to the predictive power of these equilibrium concepts. See also Figure 1.

> [Lecture 17] In general games, no-(external)-regret dynamics converges quickly to an approximate coarse correlated equilibrium (CCE).

*©2013, Tim Roughgarden. These lecture notes are provided for personal use only. See my book *Twenty Lectures on Algorithmic Game Theory*, published by Cambridge University Press, for the latest version.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: `tim@cs.stanford.edu`.

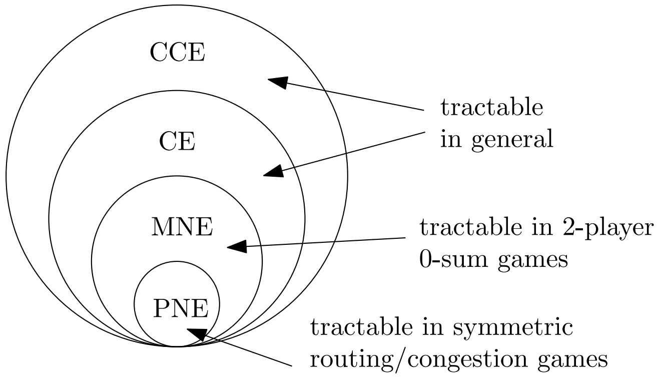
CCE
CE
MNE
PNE
tractable in general
tractable in 2-player 0-sum games
tractable in symmetric routing/congestion games

Figure 1: The hierarchy of solution concepts.

[Lecture 18] In general games, no-swap-regret dynamics converges quickly to an approximate correlated equilibrium (CE).

[Lecture 18] In two-player zero-sum games, no-(external)-regret dynamics converges quickly to an approximate mixed Nash equilibrium (MNE).

[Lecture 16] In atomic routing games that are symmetric — that is, all players share the same source and sink — ϵ-best-response dynamics converges quickly to an approximate pure Nash equilibrium (PNE).

Also, Problem 32 shows how to use linear programming to compute an exact CCE or CE of a general game, or an exact MNE of a two-player zero-sum game. Problem 22 shows how to use minimum-cost flow to compute an exact PNE of a symmetric atomic selfish routing game.

While the list above is not exhaustive, it does cover a significant chunk of the general results known for efficiently learning and computing equilibria. We'd like more, of course, such as tractability results for MNE in (non-zero-sum) two-player games and PNE for (non-symmetric) atomic routing games. No such results are known, despite non-trivial effort by many smart people. Do we merely need a new and clever idea, or are these problems intrinsically difficult? How might we prove limitations on what can be computed or learned efficiently?

These questions are in the wheelhouse of computational complexity theory. Why is it so easy to come up with computationally efficient algorithms for the minimum-spanning tree problem and so difficult to come up with one for the Travelling Salesman problem? Could it be that no efficient algorithm for the latter problem exists? If so, how can we prove it? If we can't prove it, how can we nevertheless amass evidence of intractability? These questions are, of course, addressed by the theory of NP-completeness. This lecture and the next describe analogs of NP-completeness for equilibrium computation problems. We address the technically simpler case of PNE in routing and congestion games first, and discuss MNE of bimatrix games next lecture.

2 Local Search Problems

When Johnson, Papadimitriou, and Yannakakis [2] initiated the complexity-theoretic study of local search problems, they did not have equilibrium computation in mind. However, the theory they developed is perfectly suited for reasoning about PNE computation in routing and congestion games, as we'll see in Section 3. To foreshadow the connection, computing a PNE of a congestion game is equivalent to computing a local minima of the Rosenthal potential function (Lecture 13). Hence, complexity theory for local search problems is immediately relevant to these equilibrium computation problems.

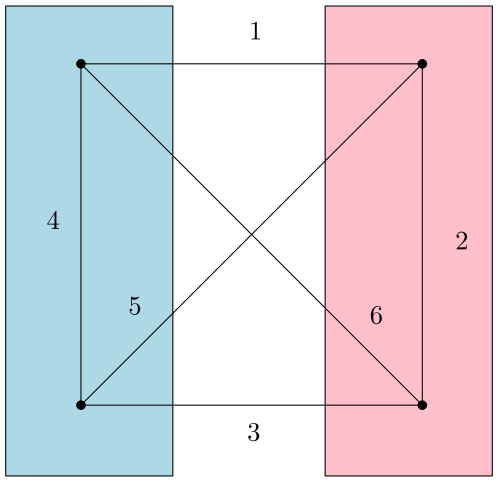

Figure 2: Max-cut instance and a local optimum that is not a global optimum.

2.1 Canonical Example: The Maximum Cut Problem

A canonical problem through which to study local search is the *maximum cut* problem. The input is an undirected graph $G = (V, E)$ with a nonnegative weight $w_e \geq 0$ for each edge. Feasible solutions correspond to cuts $(S, \bar{S})$, where $(S, \bar{S})$ is a partition of V into two non-empty sets. The objective function that we wish to maximize is the total weight of the cut edges — the edges with one endpoint in each of $S, \bar{S}$. Unlike the minimum cut problem, the maximum cut problem is NP-hard.

Local search is a natural heuristic that is useful for many NP-hard problems, including the maximum cut problem. The algorithm is very simple:

1. Start with an arbitrary cut $(S, \bar{S})$.
2. While there is an improving local move, make one.

By a *local move*, we mean moving a vertex v from one side of the cut to the other (keeping both sides non-empty). For example, when moving a vertex v from S to $\bar{S}$, the increase in objective function value is

$$\underbrace{\sum_{u \in S} w_{uv}}_{\text{newly cut}} - \underbrace{\sum_{u \in \bar{S}} w_{uv}}_{\text{newly uncut}} \; ;$$

if this difference is positive, then this is an improving local move. Local search terminates at a solution with no improving move, a *local optimum.* A local optimum need not be a global optimum; see Figure 2.

Since there are more local optima than global optima, they are generally easier to find. For example, consider the special case of maximum cut instances in which every edge has unit weight. Computing a global maximum remains an NP-hard problem, but computing a local maximum is easy. Because the objective function in this case can only take on the values $\{0, 1, 2, \ldots, |E|\}$, local search terminates (at a local maximum) in at most $|E|$ iterations.

There is no known polynomial-time algorithm (local search or otherwise) for computing local optima of maximum cut instances with general nonnegative weights. How might we amass evidence that no such algorithm exists?

The strongest type of negative result would be an unconditional one: a proof that there is no polynomial-time algorithm for the problem. No one knows how to prove unconditional results like this; in particular, such a result would separate P from NP. Instead, a natural response is to try to prove that finding local maxima of maximum cut instances is an NP-complete problem. In the next lecture we'll see why this is also too strong a negative result to shoot for. Instead, we'll develop an analog of NP-completeness tailored for local search problems.

2.2 PLS: Abstract Local Search Problems

This section and the next make precise the idea that the problem of computing a local optimum of a maximum cut instance is *as hard as any other local search problem*. This statement is in the spirit of an NP-completeness result, which establishes a problem to be as hard as any problem with efficiently verifiable solutions. For such "hardest" local search problems, we don't expect there to be any clever, problem-dependent algorithm that improves significantly over local search. This parallels the idea that for NP-complete problems, we don't expect there to be an algorithm that improves significantly over brute-force search. A byproduct of the theory developed in this and the next section is exponential lower bounds on the number of iterations required by local search to reach a local optimum.

What could we mean by "any other local search problem?" One interpretation of the definition of NP problems is that an efficient verifier of purported solutions is in some sense the minimal ingredient necessary to execute brute-force search through the set of candidate solutions. So what are the minimal ingredients necessary to run local search?

An abstract local search problem is specified by three polynomial-time algorithms.

1. The first algorithm takes as input an instance and outputs an arbitrary feasible solution. [In MaxCut, it outputs an arbitrary cut.]

2. The second algorithm takes as input an instance and a feasible solution, and returns the objective function value of the solution. [In MaxCut, it outputs the total weight of the edges crossing the provided cut.]

3. The third algorithm takes as input an instance and a feasible solution and either reports "locally optimal" or produces a better solution. [In MaxCut, it checks all $|V|$ local moves. If none are improving it outputs "locally optimal;" otherwise, it executes an improving local move and outputs the resulting cut.][1]

[1]There are some details we're glossing over. For example, all algorithms should check if the given instance is legitimate. There is also some canonical interpretation when an algorithm misbehaves, by running too long or outputting something invalid. For example, we can interpret the output of the third algorithm as "locally optimal" unless is outputs a feasible solution better than the previous one (as verified by the second algorithm) within a specified polynomial number of steps.

Every problem in PLS admits a local search algorithm: given an instance, use the first algorithm to obtain a start state, and iteratively apply the third algorithm until a locally optimal solution is reached. Since the objective function values of the candidate solutions strictly decrease until a locally optimal solution is found, and since there are only finitely many distinct candidate solutions, this procedure eventually terminates.[2] Because there can be an exponential number of feasible solutions (e.g., in a maximum cut instance), this local search procedure need not run in polynomial time.

The goal in an abstract local search problem is to compute a local optimum. This can be done by running the local search algorithm, but any correct algorithm for computing a local optimum is also allowed. The complexity class PLS is, by definition, the set of all such abstract local search problems [2]. Most if not all of the local search problems you've ever seen can be cast as PLS problems.

2.3 PLS-Completeness

Our goal is to prove that the problem of computing a local optimum of a maximum cut instance is as hard as any other local search problem. Having formalized "any other search problem," we now formalize the phrase "as hard as." This done using reductions, which are familiar from the theory of NP-completeness.

Formally, a *reduction* from a problem $L_1 \in PLS$ to a problem $L_2 \in PLS$ is two polynomial-time algorithms:

1. Algorithm A maps every instance $x \in L_1$ to an instance $A(x) \in L_2$.

2. Algorithm B maps every local optimum of $A(x)$ to a local optimum of x.

The definition of a reduction ensures that if we can solve the problem L_2 in polynomial time then, by combining it with algorithms A and B in the reduction in the natural way, we can also solve the problem L_1 in polynomial time.

Definition 2.1 ([2]) A problem $L \in PLS$ is *PLS-complete* if every problem of PLS reduces to it.

By definition, there is no polynomial-time algorithm for computing a local optimum of a PLS-complete problem, unless $PLS \subseteq P$. Most researchers believe that $PLS \not\subseteq P$, though confidence is not as strong as for the $P \neq NP$ conjecture.

A PLS-complete problem is a *single* local search problem that simultaneously encodes *every* local search problem. If we didn't already have the remarkable theory of NP-completeness to guide us, we might not believe that a PLS-complete problem could exist. As with NP-complete problems, though, PLS-complete problems *do* exist. Even more remarkably, many concrete problems that we care about are PLS-complete [2, 3]. In particular:

[2] Since each of the three algorithms runs in time polynomial in the input size, we are implicitly forcing feasible solutions to have polynomial description length. Hence, there are at most exponentially many feasible solutions.

Fact 2.2 ([3]) *Computing a local maximum of a maximum cut instance with general non-negative weights is a PLS-complete problem.*

The first step in developing a theory of PLS-completeness is to prove an analog of Cook's theorem, meaning an initial complete problem. Cook's Theorem states that 3SAT is an NP-complete problem; Johnson et al. [2] proved that a problem concerning Boolean circuits called "CircuitFlip" is PLS-complete. Once a first complete problem is identified, more can be obtained via reductions. Some such reductions were given in [2] and many more, including to the maximum cut problem, were given in [3].

We already mentioned the conditional result that if $PLS \not\subseteq P$, then there is no polynomial-time algorithm (local search or otherwise) for any PLS-complete problem. Even in the unlikely event that $PLS \subseteq P$, the specific algorithm of local search requires exponential worst-case time for all known PLS-complete problems. The reason is that Johnson et al. [2] gave an unconditional exponential lower bound for local search for the CircuitFlip algorithm, and all of the reductions that have been used to establish the PLS-completeness of other problems preserve this lower bound. In particular, local search can require an exponential number of iterations to converge to general maximum cut instances.

3 Congestion Games

We mentioned *congestion games* in passing in Lecture 13 as a natural generalization of atomic selfish routing games in which strategies are abstract subsets of a ground set, rather than paths in a graph. That is, a congestion game is described by a set E of resources (previously edges), an explicitly described strategy set $\mathcal{S}_i \subseteq 2^E$ for each player $i =, 1, 2, \ldots, k$ (previously s_i-t_i paths), and a cost $c_e(i)$ for each resource $e \in E$ and possible load $i \in \{1, 2, \ldots, k\}$. All the results we proved for atomic selfish routing games — the price of anarchy bound of $\frac{5}{2}$ for affine cost functions, the existence of PNE with arbitrary cost functions, and the convergence of ϵ-best-response dynamics to an approximate PNE in symmetric games with α-bounded jump cost functions — hold more generally, with exactly the same proofs, in congestion games.

We claim that the problem of computing a PNE of a congestion game — any PNE, if there are many — is a PLS problem. Essentially, the claim follows from the correspondence between best-response dynamics in a congestion game and local search with respect to the Rosenthal potential function (Lecture 13). Proving the claim formally involves describing the three algorithms that define a PLS problem. The first algorithm takes as input a congestion game, as described above, and returns an arbitrary strategy profile — for example, where each player takes its first strategy. The second algorithm takes a congestion game and a strategy profile $\mathbf{s}$, and returns the value of the Rosenthal potential function

$$\Phi(\mathbf{s}) = \sum_{e \in E} \sum_{i=1}^{n_e(\mathbf{s})} c_e(i), \tag{1}$$

where $n_e(\mathbf{s})$ is the number of players in the given profile $\mathbf{s}$ that use a strategy that includes resource e, and c_e is the given cost function of e. The third algorithm checks whether or not the given strategy profile is a PNE; if so, it reports "locally optimal;" if not, it executes an iteration of best-response dynamics and returns the resulting outcome (which has smaller potential function value). This can be done in time polynomial in the description of the given congestion game.

More interesting is that computing a PNE of a congestion game is as hard as every other local search problem.

Theorem 3.1 ([1]) *The problem of computing a PNE of a congestion game is PLS-complete.*

Of course, problems that require computing a PNE with additional properties, like the best or worst PNE, can only be harder.

Proof: We give a reduction from the maximum cut problem, which is PLS-complete [2]. We are given a graph $G = (V, E)$ with nonnegative edge weights $\{w_e\}_{e \in E}$. The first algorithm A of the reduction constructs a congestion game as follows:

1. Players correspond to the vertices V.
2. There are two resources for each edge $e \in E$, r_e and $\bar{r}_e$.
3. Player v has two strategies, each comprising $|\delta(v)|$ resources, where $\delta(v)$ is the set of edges incident to v in G: $\{r_e\}_{e \in \delta(v)}$ and $\{\bar{r}_e\}_{e \in \delta(v)}$.
4. The cost of a resource r_e or $\bar{r}_e$ is 0 if one player uses it, and w_e if two players use it.

The two strategies of a player can be regarded as choosing which side of a cut, S or $\bar{S}$, the player is on. Note that for an edge $e = (u, v)$ of G, there are only two players that have a strategy containing r_e or $\bar{r}_e$: the players u and v. In every strategy profile, the combined load on the twin resources r_e and $\bar{r}_e$ is exactly two: either both players u, v use the same resource (corresponding to choosing the same side of a cut) or exactly one player u, v uses each of $r_e, \bar{r}_e$ (corresponding to choosing different sides of a cut). This is why we specify cost functions only for one or two players.

There is a bijection between strategy profiles of this congestion game and cuts of the given graph G (allowing also cuts that are empty on one side), with cut $(S, \bar{S})$ corresponding to the profile in which every player corresponding to $v \in S$ (respectively, $v \in \bar{S}$) chooses its strategy that contains resources of the form r_e (respectively, $\bar{r}_e$).

This bijection maps cuts of G with weight $w(S, \bar{S})$ to strategy profiles with Rosenthal potential value (1) equal to $\sum_{e \in E} w_e - w(S, \bar{S})$. To see this, fix a cut $(S, \bar{S})$. For an edge e cut by $(S, \bar{S})$, the load on the corresponding resources r_e and $\bar{r}_e$ is 0, so these resources contribute nothing to the Rosenthal potential of the strategy profile. For an edge e not cut by $(S, \bar{S})$, the load on one of the resources $r_e, \bar{r}_e$ will be 2, and the other's load will be 0. This pair of resources contributes w_e to the potential function (1). Thus, the potential function value of the corresponding strategy profile is the total weight of edges uncut by $(S, \bar{S})$, or $\sum_{e \in E} w_e - w(S, \bar{S})$.

Cuts of G with larger weight thus correspond to strategy profiles with smaller Rosenthal potential. Local maxima of G correspond to local minima of the potential function — that is, to PNE of the congestion game. Mapping PNE of the congestion game back to local optima of G is thus trivial to do, and the reduction is complete. ■

The proof of Theorem 3.1 also provides a good example of how reductions between PLS problems tend to preserve unconditional lower bounds on the running time of local search. Specifically, it establishes a bijection between local search in maximum cut instances and best-response dynamics in congestion games. Since the former process can require an exponential number of iterations to converge, so can the latter.

Remark 3.2 The theory of PLS-completeness is useful even if you care only about concrete lower bounds on specific dynamics like best-response dynamics, and not about computational complexity per se. It can be very difficult to prove such lower bounds from scratch, and is potentially much easier to simply reduce a PLS-complete problem to the equilibrium computation problem of interest.

We can build on the proof of Theorem 3.1 to extend PLS-completeness to the special case of *symmetric* congestion games.

Theorem 3.3 ([1]) *The problem of computing a PNE of a symmetric congestion game is PLS-complete.*

At first blush, Theorem 3.3 might seem to contradict positive results that we've already proved. First, recall the polynomial-time convergence result from Lecture 16 for ϵ-best-response dynamics. We proved that convergence result only for atomic selfish routing games with a common source and sink, but the same result and proof extend without change to symmetric congestion games. Thus, Theorem 3.3 implies that there is a big difference between exact and approximate PNE, and between exact and approximate best-response dynamics in symmetric congestion games.[3] Approximate PNE are tractable and approximate best-response dynamics converge quickly, while exact PNE are intractable and exact best-response dynamics can require an exponential number of iterations to converge.

Second, recall from Problem 22 that a PNE of a symmetric atomic selfish routing game can be computed in polynomial time using minimum-cost flow. Theorem 3.3 implies that this positive result cannot be extended to abstract symmetric congestion games (unless $PLS \subseteq P$).[4]

Proof of Theorem 3.3: We reduce the problem of computing a PNE of an asymmetric congestion game — PLS-complete by Theorem 3.1 — to that of computing a PNE of a symmetric

[3]Our proof of Theorem 3.3 will also violate the α-bounded jump assumption we made in Lecture 16, but the reduction can modified to respect this condition.

[4]In the asymmetric case, the complexity of computing a PNE is PLS-complete in both atomic selfish routing games and more general abstract congestion games [1]. Also, in the asymmetric case, computing even an approximate PNE is PLS-complete, and ϵ-best-response dynamics can require an exponential number of iterations to converge [4].

congestion game. Given an asymmetric congestion game with k players and arbitrary strategy sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$, construct a "symmetrized version" as follows. The player set remains the same. The old resource set is augmented by k additional resources $r_1, \ldots, r_k$. The cost function of each of these is defined to be zero if used by only one player, and extremely large if used by two or more. Each strategy of $\mathcal{S}_i$ is supplemented by the resource r_i, and any player can adopt any one of these augmented strategies. The key insight is that at a pure Nash equilibrium of the constructed symmetric game, each player adopts the identity of exactly one player from the original (asymmetric) game. This follows from the large penalty incurred by two players that choose strategies that share one of the new resources. Such a pure Nash equilibrium is then easily mapped to one of the original asymmetric game. ■

As an exercise, why doesn't the reduction in Theorem 3.3 work for atomic routing games? The obvious way to symmetrize an asymmetric routing game with sources $s_1, \ldots, s_k$ and sinks $t_1, \ldots, t_k$ is to add new source and sink vertices s and t, and new arcs $(s, s_1), \ldots, (s, s_k)$ and $(t_1, t), \ldots, (t_k, t)$, each with a cost function that is zero with one player and extremely large with two or more players.

References

[1] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 604–612, 2004.

[2] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

[3] A. A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

[4] Alexander Skopalik and Berthold Vcking. Inapproximability of pure nash equilibria. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 355–364, 2008.