A Theorist's Toolkit **(CMU 18-859T, Fall 2013)**

Lecture 16: Constraint Satisfaction Problems

10/30/2013

Lecturer: Ryan O'Donnell *Scribe: Neal Barcelo*

1 Max-Cut SDP Approximation

Recall the Max-Cut problem from last lecture. We are given a graph $G = (V, E)$ and our goal is to find a function $F : V \to \{-1, 1\}$ that maximizes $\mathbf{Pr}_{(i,j)\sim E}[F(v_i) \neq F(v_j)]$. As we saw before, this is NP-hard to do optimally, but we had the following SDP relaxation.

$$\begin{array}{rl} \max & \text{avg}_{(i,j)\in E}\{\frac{1}{2} - \frac{1}{2}y_{ij}\} \\ \text{s.t.} & y_{vv} = 1 \quad \forall v \\ & Y = (y_{ij}) \text{ is symmetric and PSD} \end{array}$$

Note that the second condition is a relaxation of the constraint "$\exists x_i$ s.t. $y_{ij} = x_i x_j$" which is the constraint needed to exactly model the max-cut problem. Further, recall that a matrix being Positive Semi Definitie (PSD) is equivalent to the following characterizations.

1. $Y \in \mathbb{R}^{n\times m}$ is PSD.
2. There exists a $U \in \mathbb{R}^{m\times n}$ such that $Y = U^T U$.
3. There exists joint random variables $\boldsymbol{X}_1, \ldots \boldsymbol{X}_n$ such that $y_{uv} = \mathbf{E}[\boldsymbol{X}_u \boldsymbol{X}_v]$.
4. There exists vectors $\vec{U_1}, \ldots, \vec{U_n}$ such that $y_{vw} = \langle \vec{U_v}, \vec{U_w} \rangle$.

Using the last characterization, we can rewrite our SDP relaxation as follows

$$\begin{array}{rl} \max & \text{avg}_{(i,j)\in E}\{\frac{1}{2} - \frac{1}{2}\langle \vec{v_i}, \vec{v_j} \rangle\} \\ \text{s.t.} & \|\vec{v_i}\|_2^2 = 1 \\ & \text{Vector } \vec{v_i} \text{ for each } v_i \in V \end{array}$$

First let's make sure this is actually a relaxation of our original problem. To see this, consider an optimal cut $F^* : V \to \{1, -1\}$. Then, if we let $\vec{v_i} = (F^*(v_i), 0, \ldots, 0)$, all of the constraints are satisfied and the objective value remains the same. So, any solution to the original problem is also a solution to this vector program and therefore we have a relaxation. We will use SDPOpt to denote the optimal value of the SDP relaxation above.

Now the question is how can we get a real integer cut whose solution value is close to the SDPOpt. As we will see the idea is to use a rounding technique developed by [GW95]. The outline of the algorithm as follows,

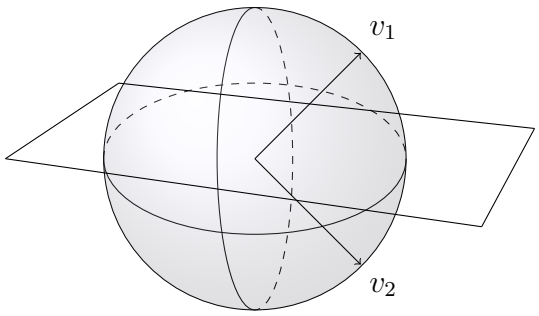

Figure 1: An example of rounding by choosing a random hyper plane. Here an edge between the vertex representing v_1 and v_2 would be cut since they are on different sides of the hyperplane.

1. Solve SDP, get vector $\vec{v_i}$ for each $v_i \in V$.
2. Pick a random hyperplane through the origin which partitions the vectors into two sets
3. Label one of these sets with $+1$ and the other with -1.

The first step can be done in polynomial time using the ellipsoid algorithm. For the second step, note that we can define a hyper plane by it's normal vector, so we just have to pick a random normal vector. More formally, let $g_1, \ldots g_n$ be n independent normal guassians. Then, we define $\vec{g} = (g_1, \ldots, g_n)$. Then, note that the side that v is on relative to this hyperplane is determined by the sign of it's dot product with $\vec{g}$. That is, we create a labeling function $F : V \to \{-1, 1\}$ with $F(v) = \text{sgn}(\langle \vec{v}, \vec{g} \rangle)$. An example of this rounding techinque can be seen in Figure 1. We first show the following lemma which tells us the probability two vectors will be cut by such a random hyper plane.

Lemma 1.1. *The probability that an edge (i, j) is cut is $\frac{1}{\pi}\angle(\vec{v_i}, \vec{v_j})$.*

Proof. Consider the plane defined by the span of $\vec{v_i}$ and $\vec{v_j}$. Let $\vec{g'}$ be the normalization of the projection of $\vec{g}$ onto this plane. The two crucial facts are the following. One is that since $\vec{g}$ is composed of gaussians which are rotationally symmetric, g' will be uniformly distributed on a unit circle on this plane. The other crucial fact is that, $\vec{v_i} \cdot \vec{g'} = v_i \cdot \vec{g}$, and so we can reduce our analysis to the plane. Now, consider Figure 1. Note that $v_j \cdot g' < 0$ if and only if g' is to the left of AB. Similarly, $v_i \cdot g' < 0$ if and only if g' is to the left of CD. Therefore, the signs of $v_i \cdot g'$ and $v_j \cdot g'$ are different (and hence the edge is cut) if and only if g' is between AB and CD. However, the angle between these is the same as the angle between $\vec{v_i}$ and $\vec{v_j}$. Since g' is uniformly distributed, the result follows. □

We will also need the following technical lemma which can be verified numerically.

Lemma 1.2. $\frac{\theta}{\pi} \geq .878(\frac{1}{2} - \frac{1}{2}\cos(\theta))$.

Theorem 1.3. *The [GW95] Algorithm is a .875 Approximation for Max-Cut.*

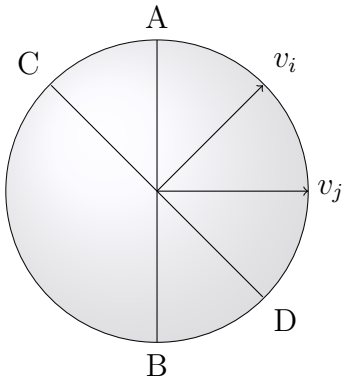

Figure 2: A figure representing the situation where the random plane will split v_i and v_j. If the projection of the normal vector, $\vec{g'}$, lies between AB and CD then v_i and v_j will be cut.

Proof. We first calculate the expected cost of the algorithm. Let F be the assignment returned by the algorithm. We have,

$$\begin{aligned}
\mathop{\mathbf{E}}_{\vec{g}}[\text{fraction of edges } F \text{ cuts}] &= \sum_{(i,j)\in E} \mathbf{Pr}[F \text{ cuts } v_i \text{ and } v_j] \\
&= \sum_{(i,j)\in E} \frac{1}{\pi}\angle(v_i, v_j) && (1) \\
&= \text{avg}_{(i,j)\in E}\left\{\frac{\angle(v_i, v_j)}{\pi}\right\} \\
&\geq .878 \cdot \text{avg}_{(i,j)\in E}\left\{\frac{1}{2} - \frac{1}{2}\cos(v_i \cdot v_j)\right\} && (2) \\
&= .878 \cdot \text{SDPOpt} && (3) \\
&\geq .878 \cdot \text{Opt} && (4)
\end{aligned}$$

Here (1) follows by lemma 1.1, (2) follows by lemma 1.2, (3) follows by the definition of the objective function in the SDP relaxation, and (4) follows since we showed that SDPOpt $\geq$ Opt. □

We end the section by noting that this approximation ratio is much better than the trivial .5 we receive from randomly placing each vertex in one of the two sides of the cut.

2 Constraint Satisfaction Problems

Constraint satisfaction problems represent a class of problems where variables are assigned to values from a domain of possible values subject to a set of constraints. As we will see

later in this section, this formulation captures many of the problems that we are interested in studying. First, consider the following formal definition. A *Constraint Satsifaction Problem* consists of a tuple $\langle V, \Omega, C \rangle$ each of which we will define below.

Definition 2.1 (Domain)**.** The domain of a CSP, Ω, is the set of values assignable to the variables $v \in V$.

Typically we will have $\Omega = \{0, 1, \ldots, q-1\}$ and often we will be in the case where $q = 2$.

Definition 2.2 (Constraint)**.** A constraint $c \in C$ consists of a function $\psi : \Omega^r \to \{0, 1\}$ and a tuple S or r variables from V.

Here we say that ψ has arity r and we let Ψ denote the set of all such allowable predicates.

Definition 2.3 (Instance)**.** An Instance $\mathcal{I}$ of a constraint satisfaction problem on an n variable set V is a list of constraints.

In order to get a feel for how we can model problems as constraint satisfaction problems, let's look at formulations of some familiar problems.

Example 2.4 (Max-Cut)**.** *For the Max-Cut problem, let* $\Omega = \{-1, 1\}$*,* V *be the set of vertices, and* $\Psi = \{\neq\}$*. The not equals predicates are of the form* $\psi : \{-1, 1\}^2 \to \{0, 1\}$ *returning* 1 *if the inputs are not equal and* 0 *otherwise.*

Example 2.5 (E3SAT)**.** *In the E3SAT, every clause must have exactly 3 literals. To model this as a CSP, let* $\Omega = \{0, 1\}$*, and* $\Psi = \{$*3-ary OR with literals*$\}$*. As an example one such* $\psi \in \Psi$ *would be* $\psi(x, y, z) = 1 - (1 - x)(y)(z)$ *which represents the clause* $x \vee \neg y \vee \neg z$*.*

Example 2.6 (NAE3-SAT)**.** *This stands for not all equal 3SAT, where we wish to assign true or false to each variable such that no clause has all of the literals equal. (So unlike 3SAT we disallow all true as well as all false). To model this as a CSP, let* $\Omega = \{0, 1\}$ *and let* $\Psi = \{$*NAE3 with literals*$\}$*. For example the* ψ *for the clause* $x \vee y \vee z$ *is* 1 *if not all of* x y *and* z *are the same.*

Example 2.7 (3-Cut (3-Color))**.** *Here we model the problem of 3-Coloring a graph as a CSP. Let* $\Omega = \{$*red,green,blue*$\}$ *and* $\Psi = \{\neq\}$*. Again the not equals predicate is similarly defined as in the Max-Cut problem where it takes as input two colors and returns* 1 *if they are not equal.*

Example 2.8 (Max-Bij(q)/Unique-Games(q))**.** *Let* $\Omega = \{0, 1, \ldots, q-1\}$*. We define* $\Psi = \{$*any "bijective" arity* 2 *predicate*$\}$*. By this we mean* $\psi : \Omega^2 \to \{0, 1\} \in \Psi$ *if and only if for all* $a \in \Omega$ *there exists a unique* $b \in \Omega$ *such that* $\Psi(a, b) = 1$*.*

Now that we have a feel for how many familiar problems can be formulated as CSP's, let's look at what it means to solve a CSP and how we can measure the goodness of solutions.

Definition 2.9 (Assignment)**.** Given a CSP instance $\mathcal{I}$, an *assignment* is a map $f : V \to \Omega$. We say that f satisfies a constraint $\psi = (C, S) \in \Psi$ if $\psi(f(S)) = 1$. Finally, let $\mathrm{Val}_{\mathcal{I}}(F)$ be the fraction of constraints that F satisfies.

Using this, we can now develop a notion of an optimal assignment, which will give us a way to compare assignments produced by our algorithms. This is done in the natural way as seen below.

Definition 2.10. The optimal assignment, denoted $\text{Opt}(\mathcal{I})$, is the assignment F that maximizes $\text{Val}_{\mathcal{I}}(F)$. More formally,

$$\text{Opt}(\mathcal{I}) := \max_{F:V\to\Omega}\{Val_{\mathcal{I}}\}$$

Note that $\text{Opt}(\mathcal{I}) \in [0,1]$. Finally, we say that $\mathcal{I}$ is satisfiable if $\text{Opt}(\mathcal{I}) = 1$.

2.1 Algorithmic Problems for CSP's

So now that we have developed a way of measuring the "goodness" of different assignments, there are several algorithm problems to consider when solving CSP's. We list these below and then examine what is known for each problem type.

1. Satisfiability: Given a CSP instance $\mathcal{I}$, decide if $\mathcal{I}$ is satisfiable. Recall that this is equivalent to determining if there is an assignment that satisfies all constraints.
2. Optimization: Find F with value as large as possible. This is equivalent to finding the assignment which satisfies the largest fraction of clauses.
3. Refutation: Given an instance $\mathcal{I}$, out a "proof" of $\text{Opt}(\mathcal{I}) \leq \beta''$for β as large as you can. We will see later exactly what we mean by a "proof".

2.1.1 Satisfiability

Let's look at task 1 first, determining whether a CSP instance $\mathcal{I}$ is decidable. If we look at the problems that we formulated before we'll see that every problem is either in P or $\mathsf{NP}-$Complete. In particular we have the following classifications.

Problems contained in P.

- Max-Cut
- Max-Bijection(q)
- 2-SAT
- Order(10) ($\Omega = \{0,\ldots,9\}$ constraints of the form $F(x) < F(y)$).

Problems that are $\mathsf{NP}-$complete.

- 3SAT
- NAE-SAT
- 1-out-of-3-SAT
- 3-Color

Let's take a closer look at some of the problems that are in P. While it may seem strange at first that Max-Cut is listed in P, recall that we are considering the satisfiability problem. So for Max-Cut, this question is equivalent to asking if the given graph is bipartite which is

clearly in P. Similarly, the satisfaction problem for Max-Bijection(q) is in P since for every assignment of values to a variable, it will uniquely force all other assignments. So we can simply try all possibilities for a single variable and see if the resulting forced assignments ever satisfy all constraints. One can also show that 2-SAT is in P but the argument is slightly more complicated than Max-Cut. Lastly, the Order(10), this can be solved using a topological sort which is in P.

Given the evidence above, one might wonder whether every CSP satisfiability problem is in P or NP−complete. Indeed this is the currently believed conjecture which we state below.

Conjecture 2.11 (Feder Vardi Dichotomy Conjecture, [Var98])**.** *Every CSP Satisfiability problem is in* P *or* NP−*complete.*

There is actually a stronger conjecture which gives an algorithmic way of determining whether a problem is in P or NP−complete.

Conjecture 2.12 (Algorithm Dichotomy Conjecture [BJK05])**.** *CSP*(Ψ) *satisfiability is in* P *if* Ψ *has a "Taylor Polymorphism" and in* NP−*complete otherwise.*

The second part of this conjecture has been proven but the first part is still open, and in particular they are missing algorithms for such problems that have "Taylor Polymorphisms". Schafer [Sch78] showed the first part for $|\Omega| = 2$ and Bolutov [Bul06] showed it for $|\Omega| = 3$.

2.1.2 Optimization and Refutation

So we saw that for satisfiability problems there is a nice conjecture about when problems are easy and hard. Unfortunately, almost all optimization problems for CSP's are NP−hard. Because of this, we will often be interested in approximating the optimal assignment.

Definition 2.13 (optimization (α, β)-approximation)**.** An efficient algorithm A is an (α, β)−approximation algorithm if for all $\mathcal{I}$ with $\text{Opt}(I) \geq \beta$, A finds an assignment F with $\text{Val}_{\mathcal{I}}(F) \geq \alpha$.

Note that one algorithm can be (α, β)−approximate simultaneously for several α, β. For example [GW95] is a $(.878\beta, \beta)$−approximation for all $\beta \in [0, 1]$. We can similarly define a notion of approximation for refutation algorithms since most refutation problems are NP−hard.

Definition 2.14 (refutation (α, β)-approximation)**.** For all $\mathcal{I}$ with $\text{Opt}(I) < \alpha$, A outputs a "proof" of "$\text{Opt}(\mathcal{I}) < \beta''$ or better.

Remark 2.15. We remark that refutation (α, β)-approximation is strictly weaker than optimization (α, β)-approximation. To see why this is true, say A is an (α, β) approximation algorithm. If it doesn't output a value of at least α, then by the definition of optimization approximations we know that $\text{Opt}(\mathcal{I}) < \beta$. However this constitutes a proof that $\text{Opt}(\mathcal{I}) < \beta$ and therefore we have a refutation approximation.

While we do not have nice dichotomy conjectures like we do for satisfiability problems, there are several specific problems for which we know can classify the hardness based on the desired approximation. Let's take a few examples.

Max - E3SAT Here we are given a E3SAT instance and we want to satisfy as many clauses as possible. The following results are known

- $(1,1)-$approximation: $\mathsf{NP}-$hard.
- $(1-10^{-1},1)-$approximation: $\mathsf{NP}-$hard. ("PCP Theorem")
- $(\frac{7}{8},1)-$approximation: In P. (Just use a random setting and in expectation you meet $7/8$ of the clauses).
- $(\frac{7}{8}+\epsilon,1)-$approximation, for all $\epsilon>0$: $\mathsf{NP}-$hard [Hås01]

So as you can see for Max-E3SAT, the book is closed on the hardness of approximating. One might wonder what changes if we remove the exactly 3 portion of the problem and just consider Max-3SAT. Using a complicated SDP approach, [KZ97] were able to show that $(\frac{7}{8},1)$ is still in P. Note that you can no longer just choose randomly since now a clause may consist of a single literal.

Max-Cut We have seen the Max-cut problem from the beginning of this lecture, so let's look at what is known for the hardness of approximating this.

- $(1,1)-$approximation: In P. We saw before this is just asking if the graph is bipartite.
- $(\frac{3}{4},\frac{3}{4})-$approximation: $\mathsf{NP}-$hard.
- $(\frac{3}{4},\frac{4}{5}-\epsilon)-$approximation: $\mathsf{NP}-$hard. [Hås01, TSSW96]
- $(\frac{3}{4},\frac{1}{2}+\frac{1}{2\sqrt{2}})-$approximation: In P. This is the algorithm we covered in section 1.
- $(\frac{3}{4},\frac{5}{6})-$approximation: ?? Still don't know the answer to this question.

2.1.3 Unique Games

The following material was added on piazza. Recall that Unique-Games(q) is the CSP where the domain is $[q]$ and any "bijective" binary constraint is allowed. A special case of this is called Max-2Lin(q), in which the domain is $\mathbb{Z}_q$ and all the constraints are of the form $v_i - v_j = c \pmod q$ for variables v_i, v_j and constants c. When $q = 2$, a further special case is Max-Cut.

As we discussed, the satisfiability problem, $(1,1)$-approximating UG(q) is in P, easily solved by "propagation". On the other hand, it's easy to show that $(1-\epsilon,1-\epsilon)$-approximating UG(q) is NP-hard for any $\epsilon > 0$; in fact, this is true even for Max-Cut.

What about $(1/2, 1-\epsilon)$-approximating Unique-Games(q)? For $q = 2$ this is in P. However, for $q \geq 3$ it is unknown if it's in P or if it's NP-complete. This is even true in the special case of Max-2Lin(q).

The original "Unique Games Conjecture" of Khot (2002) is, "for all $\epsilon > 0$ there exists q such that $(\epsilon, 1-\epsilon)$-approximating UG(q) is NP-hard".

It is known to be equivalent to the following: "for all $\epsilon > 0$ there exists q such that $(1/2, 1-\epsilon)$-approximating Max-2Lin(q) is NP-hard". Also, that $1/2$ can be any fixed constant between 0 and 1.

References

[BJK05] Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:2005, 2005.

[Bul06] Andrei Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, January 2006.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

[KZ97] Howard Karloff and Uri Zwick. A 7/8-approximation algorithm for max 3sat? In *In Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 406–415, 1997.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC '78, pages 216–226, New York, NY, USA, 1978. ACM.

[TSSW96] Luca Trevisan, Gregory Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 617–626, 1996.

[Var98] Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput*, 28:57–104, 1998.