Lecture 13

Constraint Satisfaction Problems*

Now that we've seen all the theory, let's move on to the "applications", starting with CSPs. In fact, we begin with some examples of CSPs.

13.1 Examples

Max-3-SAT. In this problem, the *domain* is $\{0,1\}$; a collection of *constraint types* are disjunctions of the form $\{x_i \vee x_j \vee x_k, x_i \vee x_j \vee \overline{x_k}, x_i \vee \overline{x_j} \vee x_k, \ldots, \overline{x_i} \vee \overline{x_j} \vee \overline{x_k}, x_i \vee x_j, \ldots, \overline{x_i} \vee \overline{x_j}, x_i, \overline{x_i}\}$ resulting in total of 14 constraint types.

Max-Cut. Here, the domain is again $\{0,1\}$, the collection of constraint types is simply $\{v_i \neq v_j\}$. Intuitively, we have one such constraint for each edge in a given Max-Cut instance, and the goal is to label the vertices.

Max-E3-Lin2. The problem deals with clauses of 'E'xactly 3 variables mod '2'. As the name suggests, the domain is again $\{0,1\}$, and the constraint types are $\{x_i + x_j + x_k = 0 \mod 2, x_i + x_j + x_k = 1 \mod 2\}$.

Order(10). Here, the domain is the set $\{0,1,2,\ldots,9\}$, and the constraint types include $\{x_i < x_j\}$.

Now that we've seen some examples of CSPs, lets get down to some useful notation.

13.2 Notation

13.2.1 CSP Problem

A general Constraint Satisfaction Problem (CSP) is accompanied by a domain D of size q, typically $D = \{0,1,\ldots,q-1\}$ and a collection Γ of constraint types. Alternately, one could view Γ as a collection of relations $R : D^k \to \{0,1\}$, where k is the arity of the relation R. This relation evaluates to 1 if corresponding particular assignment satisfies the constraint.

*Lecturer: Ryan O'Donnell. Scribe: Ravishankar Krishnaswamy.

Together, the combination D and Γ form the CSP problem, typically denoted as $\mathsf{CSP}(\Gamma)$ (here the domain of the variables D is implicitly defined in Γ).

For example, in Max-Cut, $D = \{0, 1\}$ and $\Gamma \equiv \{\neq\}$, where $\neq (a, b) = 1 \iff a \neq b$.

13.2.2 CSP Instance

Now that we've seen how to describe problems, let's move on to describing specific instances of problems. A instance $\mathfrak{C} \in \mathsf{CSP}(\Gamma)$ over n variables is a *weighted list* of constraints $C = (R, S)$ where $R \in \Gamma$ is the constraint type, and S is the scope for R, i.e., it is an unordered k-tuple of distinct variables, where k is the arity of R.

Note. Since we will only be dealing with finite instances (and focus on *additive approximations*) in this lecture, let us WLOG assume that the weights $w_C \geq 0$, and they sum to 1. In particular, this lets us view the problem instance as coming with a probability distribution over clauses, and so we can write $C \sim \mathfrak{C}$ to denote that C is chosen with probability w_C. This view will substantially simplify the coming notation.

Now for a small example. Consider the Max-3-SAT instance: $1/4$ weight on $x_1 \vee x_3$, $1/2$ weight on $x_2 \vee \overline{x_3}$, and $1/4$ weight on $x_1 \vee x_3 \vee x_4$. In the form we have stated above, the first constraint can be written as $C = (R, S)$ where $R(a, b) = 1$ unless $a = 0, b = 1$, and the scope $S = (x_2, x_3)$ (ordered).

13.2.3 Algorithmic Assignment

An assignment $F : V \to D$ is said to satisfy a constraint $C \in \mathfrak{C}$ if $R(F(S)) = 1$, i.e., apply the labels suggested by F to the variables in the scope S and check if that assignment satisfies R. Given this, we can define the value of this assignment as

$$\mathsf{Val}_{\mathfrak{C}}(F) = \mathop{\mathbf{E}}_{C=(R,S)\sim\mathfrak{C}} [R(F(S))] \tag{13.1}$$

Finally, we also define $\mathrm{Opt}(\mathfrak{C}) = \max_F \mathsf{Val}_{\mathfrak{C}}(F)$. Using this definition, we say that an instance $\mathfrak{C}$ is satisfiable if $\mathrm{Opt}(\mathfrak{C}) = 1$.

Note. Since our view of satisfiability is of the form given in (13.1), we don't need to constrain R to have range $\{0, 1\}$. Indeed, we could allow range of $[0, 1]$, these are called "fuzzy predicates" and the CSP is called a vCSP, for value-CSP.

13.2.4 Approximation Algorithm

We view an algorithm with the following useful yardstick, albeit somewhat non-standard.

Definition 13.1. An algorithm (α, β)-approximates a CSP problem $\mathsf{CSP}(\Gamma)$ if for all instances with $\mathrm{Opt} \geq \beta$, the algorithm finds an assignment with value at least α.

Here's the above notation in action for the Max-Cut problem.

Algorithm	Guarantee	Range
Greedy	$(\frac{1}{2}, \beta)$	for all β
Random	$(\frac{1}{2}, \beta)$	for all β
GW-Algorithm	$(0.878\beta, \beta)$	for all β
Greedy	$(1, 1)$	for $\beta = 1$ (bipartite graph)

Exercise. The above chart shows that greedy is good if Opt is 1, i.e., the graph is bipartite. Is there a graph where Opt $\geq 1 - \epsilon$, yet the greedy algorithm only gets value $0.5 + \epsilon$?

Now that we're familiar with this notation, let's look at 3SAT, and begin with some hardness results. First is the fundamental NP-completeness result cast in our framework.

Theorem 13.2 (Cook-Levin)**.** $(1, 1)$*-approximating 3-SAT is NP-Hard.*

Subsequently, Håstad showed in his seminal paper [Hås01] the following result.

Theorem 13.3 (Håstad)**.** $(7/8 + \epsilon, 1)$*-approximating 3-SAT is NP-Hard for every* $\epsilon > 0$*.*

Now let's see what we know from the positive side. Indeed, we would expect the trivial *random assignment* algorithm to get 7/8th fraction of all clauses, right? Unfortunately, while this intuition is right if all clauses had exactly 3 literals, it breaks down if there are *fewer*! In particular, this assignment gets only 1/2 fraction of the clauses on the trivial instance where there is only one clause x_1. So the random algorithm can be easily seen to be $(1/2, \beta)$-approximation for any β. However, if we restrict our attention to the Max-E3-SAT problem, the random assignment algorithm is a $7/8, \beta)$-approximation for all β.

While this result along with Håstad's show that there is essentially no gap between the hardness and the easiness for Max-E3-SAT instances, the same is not true of all CSPs, including general 3SAT instances. In the next two lectures, we will address the problem of using LPs and SDPs to design CSP algorithms. This will familiarize us with using a "hierarchy of tools" in this paradigm, each a stronger technique than the previous.

1. Trivial Random Assignment
2. Linear Programming Relaxations (*)
3. Semidefinite Programming Relaxations
4. LPs/SDPs with "lifted" hierarchy constraints [Sherali-Adams, Lasserre, etc. (see, e.g., [GLS88]]

In this lecture, however, we will focus on using LPs to attack (well, approximately solve) CSPs.

13.3 Canonical LP Relaxation

Suppose we are given an instance $\mathfrak{C} \in \mathsf{CSP}(\Gamma)$. Then our canonical LP has variables $\mu_v[\cdot]$ for each v and $\lambda_C[\cdot]$ for all constraints. μ_v denotes the probability distribution over labels for variable v, and λ_C denotes the probability distribution over "local assignments" for constraint C. Indeed, if $C = (R, S)$, λ_C is a probability distribution over $D^{|S|}$.

We then write the following linear constraints in our canonical LP.

1. "μ_v is a prob. distribution on D for all $v \in V$". We encode this by the constraint $\mu_v[0] + \mu_v[1] + \ldots + \mu_v[q-1] = 1$ and $\mu_v[i] \geq 0$.

2. "λ_C is a prob. distribution on $D^{|S|}$ (i.e., local assignments) for all $C \in \mathfrak{C}$". For example, if some constraint had a scope $S = (x_1, x_2)$, the LP would encode the constraint $\lambda_C[x_1 \to 0, x_2 \to 0] + \lambda_C[x_1 \to 0, x_2 \to 1] + \ldots + \lambda_C[x_1 \to 1, x_2 \to 1] = 1$ and $\lambda_C[\cdot] \geq 0$. It is easy to see that we can encode these constraints in a similar manner to the μ distribution linear constraint.

3. "Objective Function". Our objective is easy to describe: $\max \mathsf{LPVal}$ which is the "total fractional extent to which the constraints are satisfied". Mathematically, this expression is
$$\mathsf{LPVal} = \mathop{\mathbf{E}}_{C \in \mathfrak{C}} \left[\mathop{\mathbf{E}}_{L \sim \lambda_C} [R(L(S))] \right]$$
As is, however, there is a big issue with the LP. That is, regardless of the problem and regardless of the instance, $\mathrm{Opt}(LP) = 1$. Indeed, there is nothing tying the λ_C's and the μ_v's. The LP solution could pick any local assignment A which satisfies a constraint and set $\lambda_C[A] := 1$. To resolve this issue, we put in one more set of constraints.

4. "Consistency Constraints". We write down the following constraint for every $C \in \mathfrak{C}$, for every $v \in V$, and for every label $l \in D$:
$$\mathop{\mathbf{Pr}}_{L \sim \lambda_C} [L(v) = l] = \mathop{\mathbf{Pr}}_{L \sim \mu_v} [L = l]$$
For example, if C had scope (x_1, x_2), then the above constraint would look like
$$\lambda_C[x_1 \to 3, x_2 \to 0] + \lambda_C[x_1 \to 3, x_2 \to 1] + \ldots \lambda_C[x_1 \to 3, x_2 \to q-1] = \mu_{x_1}[3]$$
for the choice of C, x_1 and label 3.

Note. In the above formulation, two constraints could have the same scope but the LP can assign different distributions over local assignments. We could put in constraints which enforce that if the scope is the same, then the local distribution is also the same, but these it is somewhat difficult using them in a meaningful way when we

are rounding the fractional solution. Another way of strengthening the LP could be to ensure that pairwise marginals are preserved instead of just μ_v. In fact, it is these kinds of constraints (albeit more methodically) which form the basis of LP/SDP hierarchies.

Theorem 13.4. *The above LP is a relaxation for the problem $CSP(\Gamma)$*

Proof. It is easy to see that Opt, the optimal assignment induces a feasible LP solution of value Opt, and so the LP is a relaxation. In particular we have $\mathrm{Opt}(\mathfrak{C}) \leq \mathsf{LPOpt}(\mathfrak{C}) \leq 1$. □

Having formulated the LP, the immediate next question is if the LP is any good? That is, can we extract a good integral solution of approximate value of LPOpt? This is our focus in the coming section.

13.4 Randomized Rounding for Max-k-SAT

Suppose we are given an instance $\mathfrak{C}$, with the optimal LP solution $Ł^* = (\lambda_C^*, \mu_v^*)$. The main question is how do we "round" this fractional solution to obtain an integer solution? The answer, (at least for this problem) is *randomized rounding*.

That is, for each variable v, define

$$F(v) = \begin{cases} 0 & \text{with probability } \mu_v^*[0] \\ 1 & \text{with probability } \mu_v^*[1] \end{cases}$$

Let us now study the expected value of this randomized solution, i.e., $\mathbf{E}_F[\mathsf{Val}_{\mathfrak{C}}[F]]$. To this end, notice that

$$\begin{aligned} \mathop{\mathbf{E}}_F[\mathsf{Val}_{\mathfrak{C}}[F]] &= \mathop{\mathbf{E}}_F\left[\mathop{\mathbf{Pr}}_{C\in\mathfrak{C}}[F \text{ satisfies } C]\right] && (13.2) \\ &= \mathop{\mathbf{E}}_{C\in\mathfrak{C}}\left[\mathop{\mathbf{Pr}}_F[F \text{ satisfies } C]\right] && (13.3) \end{aligned}$$

In the equality above, we have switched the order of taking expectation, which simply amounts to changing the order of summation. This now lets us focus on the algorithm's performance *term-by-term*. Indeed, let us fix some $C \in \mathfrak{C}$ and look at $\mathbf{Pr}_F[F \text{ satisfies } C]$?

For example, suppose $C = x_1 \vee x_3 \vee \overline{x_{10}}$, say. Then the probability that F *does not satisfy* C is precisely the probability that $x_1 = 0$ *and* $x_3 = 0$ *and* $x_{10} = 1$. Since these choices are independent, we get

$$\mathbf{Pr}[F \text{ satisfies } C] = 1 - \mu_1^*[0]\mu_3^*[0]\mu_{10}^*[1]$$

The function b_C. And in general, suppose the constraint C has scope S. Let $b_C : S \to \{0,1\}$ denote the **bad assignment** for S w.r.t C, i.e., $b_C(v)$ is the unique assignment to variable v

which makes C unsatisfied. In the above example, $b_C(x_1) = 0$, $b_C(x_3) = 0$ and $b_C(x_{10}) = 1$. In this notation, we have

$$\mathbf{Pr}\left[F \text{ satisfies } C\right] = 1 - \prod_{v \in S} \mu_v^* \left[b_C(v)\right] \tag{13.4}$$

Now, it remains to compare this expected value with that of the LP. Can the LP fetch much profit more from each clause? To answer this, let us consider the contribution of λ_C^* to the LP value.

To this end, define

$$\begin{aligned}
p_C &:= \mathop{\mathbf{Pr}}_{L \sim \lambda_C^*}\left[L \text{ satisfies } C\right] && (13.5)\\
&= \mathop{\mathbf{Pr}}_{L \sim \lambda_C^*}\left[\bigcup_{v \in S} \{L(v) \neq b_C(v)\}\right] && (13.6)\\
&\leq \sum_{v \in S} \mathop{\mathbf{Pr}}_{L \sim \lambda_C^*}\left[\{L(v) \neq b_C(v)\}\right] && (13.7)\\
&= \sum_{v \in S} \left(1 - \mu_v^*\left[b_C(v)\right]\right) && (13.8)
\end{aligned}$$

Here, the final step used the "consistent marginals" constraint of the LP for the constraint C, variable v and label $b_C(v)$ inside each term of the summation. So it remains to compare the values in (13.4) and (13.8). Looking at the two forms, it suggests an application of the AM-GM inequality. With this in mind, let us express (13.4) as

$$\begin{aligned}
\mathbf{Pr}\left[F \text{ satisfies } C\right] &= 1 - \prod_{v \in S} \mu_v^*\left[b_C(v)\right] && (13.9)\\
&= 1 - GM_{v \in S}\{\mu_v^*(b_c(v))\}^{|S|} && (13.10)\\
&\geq 1 - AM_{v \in S}\{\mu_v^*(b_c(v))\}^{|S|} && (13.11)\\
&= 1 - (1 - AM_{v \in S}\{1 - \mu_v^*(b_c(v))\})^{|S|} && (13.12)\\
&\geq 1 - \left(1 - \frac{p_C}{|S|}\right)^{|S|} && (13.13)
\end{aligned}$$

Now for a plot (which compares the expression in (13.13) with the value p_C) to complete the story.

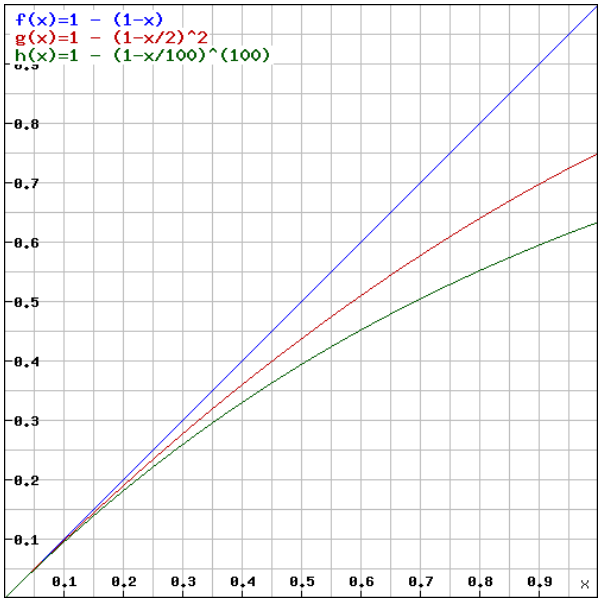

Figure 13.1: Comparing p_C (X axis) with $\mathbf{Pr}_F[F \text{ satisfies } C]$ (Y axis)

So as you can see, the worst case limit happens when $|S| \to \infty$, in which case the ratio reaches a limit of $0.63 = 1 - 1/e$. Using this worst case (proof by picture) we get

$$\begin{aligned}
\mathop{\mathbf{E}}_{F}\left[\mathsf{Val}_{\mathfrak{C}}\left[F\right]\right] &= \mathop{\mathbf{E}}_{F}\left[\Pr_{C\in\mathfrak{C}}\left[F \text{ satisfies } C\right]\right] && (13.14)\\
&= \mathop{\mathbf{E}}_{C\in\mathfrak{C}}\left[\Pr_{F}\left[F \text{ satisfies } C\right]\right] && (13.15)\\
&\geq \left(1-\frac{1}{e}\right) E_{C\sim\mathfrak{C}}\left[p_C\right] && (13.16)\\
&= \left(1-\frac{1}{e}\right) \mathsf{LPVal}(\mathcal{L}^*) && (13.17)\\
&= \left(1-\frac{1}{e}\right) \mathsf{LPOpt}(\mathfrak{C}) && (13.18)\\
&\geq \left(1-\frac{1}{e}\right) \mathrm{Opt}(\mathfrak{C}) && (13.19)
\end{aligned}$$

So we get

Theorem 13.5. *Randomized rounding is a $((1-1/e)\beta, \beta)$-approximation, for Max-SAT, for any β.*

We note that a slightly smarter rounding can give a 3/4 approximation [GW94], and also that SDPs do even better. But that's a story for another day.

Bibliography

[GLS88] Martin Grötschel, Lásló Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988. 4

[GW94] Michel X. Goemans and David P. Williamson. New 3/4-approximation algorithms for max sat. 1994. 13.4

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, July 2001. 13.2.4