Modeling General Constraint Satisfaction Problems

July 20, 2016

In this note, we will discuss a general technique in modeling constraint satisfaction problems (CSPs) as a convex program (in particular, as LP or SDP). We will also see how to design an approximation algorithm for these problems (since they are NP-hard in general, we will be looking for near-optimal solution).

1 General CSPs

A general *constraint satisfaction problem (CSP)* is described by the following ingredients:

- Domain: $D = \{0, 1, \ldots, q-1\}$.
- Constraint type: $\Gamma = \{R : D^k \to \{0,1\}\}$ where k is the arity of the relation R. The relation evaluates to 1 if the constraint is satisfied.

The CSP described by these ingredients are specified by $\mathsf{CSP}(\Gamma)$. The parameters k and D are implicitly defined in Γ.

CSP instances: Let $X = \{x_1, \ldots, x_n\}$ be variables. Then the $\mathsf{CSP}(\Gamma)$ instance is described by a collection of clauses $C_i = (R_i, S_i)$ where $R_i \in \Gamma$ and S_i is an ordered k-tuple. The clause C_i is satisfied by an assignment $f : X \to D$ if $R_i(f(S_i)) = 1$.

Optimizing CSP: The goal of the optimization problem is to find the best assignment. Let $\mathcal{D}$ be an instance (or distribution) of $\mathsf{CSP}(\Gamma)$. Given an assignment $\phi : X \to D$, we define its value as $val(\phi) = \sum_{i=1}^{m} R_i(\phi(S_i))$.

1.1 Many examples

1. *Max-3-SAT:* In this case, Γ represents the OR-predicate.
2. *Max-Cut:* We can model this as a CSP where the vertices that get assigned to one side get 0 value and the other gets 1; so the domain set is $D = \{0,1\}$. The set Γ contains only $\neq$ predicate.
3. *Max-E3-LIN2:* The domain is $\{0,1\}$. There are two types of precidates in Γ, i.e. $x_i + x_j + x_k = 0$ and $x_i + x_j = x_k = 1$ (under modolu two.)
4. *Max Acyclic Subgraph:* An edge is satisfied if it goes from lower to higher number. The domain is $[n]$. The predicate is $<$.

2 Some Algorithms for CSPs

2.1 LP Rounding for SAT

Maximum Sat: We are given a collection of clauses C_i in the CNF, where each clause $C_i = (R_i, S_i)$. Our goal is to maximize the total weight of satisfied clauses.

LP Relaxation: We have a variable $\lambda_i[Q]$ for each clause $C_i = (R_i, S_i)$ where R_i is in the form $R_i = \{0,1\}^{|S_i|} \setminus B_i$; we should intepret B_i as the only assignment that does not satisfy clause C_i (since this is a CNF formula). For instance, for clause $C_i = x_a \cup \bar{x}_b$, we would have $R_i = \{0,1\}^2 \setminus \{(0,1)\}$, i.e. $B_i = (0,1)$. In other words, B_i represents the only false assignment for C_i. Also, we have variable $\mu_j[l]$ for each variable x_j and $l \in \{0,1\}$. The LP is summarized below:

(LP-SAT)

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} \sum_{Q \in R_i} \lambda_i[Q] \\ \text{s.t.} \quad & \mu_j[0] + \mu_j[1] = 1 \text{ for all } j \\ & \sum_{Q} \lambda_i[Q] = 1 \text{ for all } i \\ & \sum_{Q:Q[j]=l} \lambda_i[Q] = \mu_j[l] \text{ for all } i,\ j \in S_i,\ l \in D \end{aligned}$$

LP Rounding: We construct an assignment $f : \{x_1, \dots, x_n\} \to \{0,1\}$ as follows. For each variable x_j, we assign $f(x_j) = 1$ with probability $\mu_j[1]$; otherwise, $f(x_j) = 0$. In words, we interpret μ_j as a probability distribution on $D = \{0,1\}$ and use it to design a randomized algorithm.

Analysis: We first analyze the probability that a clause C_i is satisfied by the assignment f. Notice that such probability is $1 - Pr[(\forall j \in S_i) f(x_j) = B_i(x_j)]$. Since we perform the randomized rounding independently for each variable j, the second term is:

$$Pr[(\forall x_j \in S_i) f(x_j) = B_i(x_j)] = \prod_{x_j \in S_i} \mu_j[B_i(x_j)]$$

By using AM-GM, we get

$$1 - \prod_{j \in S_i} \mu_j[B_i(x_j)] \geq 1 - \left(1 - \frac{\sum_{x_j \in S_i} (1 - \mu_j[B_i(x_j)])}{|S_i|}\right)^{|S_i|}$$

We denote the term $\sum_{j \in S_i} (1 - \mu_j[B_i(x_j)])$ by p_i. So we have the probability of satisfying clause C_i at least $1 - (1 - p_i/|S_i|)^{|S_i|}$.

Lemma 2.1. $\sum_{Q \in R_i} \lambda_i[Q] \leq p_i$

Proof. Notice that $\sum_{Q \in R_i} \lambda_i[Q] = \sum_{Q:Q \neq B_i} \lambda_i[Q] \leq \sum_{j \in S_i} \sum_{Q:Q(j) \neq B_i(x_j)} \lambda_i[Q] = \sum_{j \in S_i} (1 - \mu_j[B_i(x_j)])$. □

Now we can apply the convexity of the function $h(x) = (1 - x/r)^r$, together with the Jensen's inequality, to finish the proof. Since h is convex, we have $h(x) \leq xh(1) + (1 - x)h(0)$ for all $x \in (0, 1)$.

2.2 Maximum Cut

Consider first the following quadratic program:

$$\max \sum_{ij \in E} \frac{(1 - y_i y_j)}{2}, (\forall i) y_i \in \{-1, 1\}$$

This quadratic (integer) program also captures the maximum cut problem. We now relax it such that each y_i is replaced by a vector $v_i \in \mathbb{R}^n$, so now we get the following convex program.

(SDP-MC)

$$\begin{aligned} \max \quad & \sum_{ij \in E} \frac{(1 - v_i \cdot v_j)}{2} \\ \text{s.t.} \quad & ||v_i||^2 = 1 \text{ for all } i \end{aligned}$$

Lemma 2.2. *This is a relaxation of MaxCut.*

Proof. Consider any cut (A, B). Define the solution $v_i = (-1, 0, \ldots, 0)^T$ for $i \in A$ and $v_i = (1, 0, \ldots, 0)^T$ for $i \in B$. Notice that $||v_i||^2 = 1$ for all i, and that $v_i \cdot v_j \in \{-1, 1\}$. Morevoer, any edge $ij \in E$ is cut by (A, B) iff $(1 - v_i \cdot v_j) = 2$. □

We can also check that this can be solved by Ellipsoid method. Define $a_{ij} = v_i \cdot v_j$, and matrix $A = (a_{ij})$. It is clear that there are vectors $v_1, \ldots, v_n$ satisfying the consraints if and only if A is a PSD matrix. One can in fact think of the above convex program as the LP with (infinitely many) constraints of the form $x^T Ax \geq 0$ for all vectors x.

Rounding: We can think of vector solutions v_i as lying on the unit sphere. We are looking for a cut that separate as many vectors as possible, so why don't we pick a random cut? More precisely, pick a random vector $r \in \mathbb{S}^{d-1}$ and then define $A = \{i : v_i \cdot r \geq 0\}$. The cut $(A, V \setminus A)$ will be our cut. What is the probability that each edge $ij \in E$ is separated? This is exactly the value:

$$\frac{\theta_{ij}}{\pi} = \frac{\cos^{-1}(v_i \cdot v_j)}{\pi}$$

where θ_{ij} denotes the angle between the two vectors v_i and v_j.

3 Generic LP/SDP Formulation

3.1 LP

Recall that we have three types of constraints in the LP relaxation. First, we introduce distribution $\mu_j[\ell]$ for each variable x_j and each label $\ell \in D$. Also, there are constriants that enforce the variables $\{\mu_j[\ell]\}_{\ell \in D}$ being a distribution:

$$(\forall j) \sum_{\ell \in D} \mu_j[\ell] = 1$$

We also have a "local" distributions for constraints. For each constraint (R_i, S_i), for each possible assignment $Q \in D^r$, we have LP-variable $\lambda_i[Q]$ that indicates how to assign the labels to variables in S_i.

$$(\forall i) \sum_{Q} \lambda_i[Q] = 1$$

We can think of each variable μ_j and λ_i as a probability distribution. At this point, there is nothing that ensures the "consistency" of these two types of distributions, so the LP is not making any sense yet. We would need constraints that guarantee somehow that the μ and the λ are talking about the same distribution of solutions.

$$(\forall i)(\forall j : x_j \in S_i)(\forall \ell \in D) \sum_{Q : Q(x_j) = \ell} \lambda_i[Q] = \mu_j[\ell]$$

Distribution Viewpoint: All these constraints can be written concisely as simply saying that we are looking for two collections of distributions $\{\lambda_i\}_{i=1}^m$ and $\{\mu_j\}_{j=1}^n$ such that their first moments agree:

$$(\forall i)(\forall j : x_j \in S_i)(\forall \ell \in D) \mathbf{Pr}_{Q \sim \lambda_i} [Q(x_j) = \ell] = \mathbf{Pr}_{\ell' \sim \mu_j} [\ell' = \ell]$$

This viewpoint will be useful when we want to strengthen the relaxation further, as we will see in the next section when we want to write an SDP relaxation for all CSPs.

3.2 SDP

We will first construct a collection of random variables satisfying certain constraints. Then we will argue that these random variables can be computed by SDP. We will have a random variable $Y_{j,\ell}$ for each CSP-variable x_j and label $\ell \in D$. This random variable denotes the event that $f(x_j) = \ell$, where f is the tentatively "best" assignment function.

Similar to the case of LP, we have random variable λ_i for each constraint (R_i, S_i). Let $\mathcal{D}$ be the unknown "optimal distribution" that we are looking for. In the integral world, this optimal distribution would assign $\mathbf{Pr}[Y_{j,\ell} = 1] = 1$ if $f(x_j) = \ell$; and $\mathbf{Pr}[Y_{j,\ell} = 1] = 0$ otherwise. We need to somehow "glue together" the random variables λ_i and the random variables $Y_{j,\ell}$. Consider two types of constraints:

- **First moment:** For each constraint i, for each $x_j \in S_i$, and for each $\ell \in D$, we have

$$\mathbf{Pr}_{Q \sim \lambda_i} [Q(x_j) = \ell] = \mathbf{E}_{\mathcal{D}} [Y_{j,\ell}]$$

- **Second moment:** For each constraint i, for each $x_j, x_{j'} \in S_i$, and for each $\ell, \ell' \in D$, we have

$$\mathbf{Pr}_{Q \sim \lambda_i} [Q(x_j) = \ell \wedge Q(x_{j'}) = \ell'] = \mathbf{E}_{\mathcal{D}} [Y_{j,\ell} Y_{j',\ell'}]$$

One should keep in mind that the random variables $Y_{j,\ell}$ are not necessarily independent, so the expected value of a product is not the product of two expected values. Let us denote the new relaxation by (SDP).

Lemma 3.1. *The new relaxation is at least as strong as the generic LP relaxation.*

Proof. Let $\mathcal{D}$ be a joint distribution that satisfies all the (SDP) constraints. We have to define the right choices of $\mu_j[\ell]$. This can be done by simply $\mu_j[\ell] = \mathbf{E}_{\mathcal{D}}[Y_{j,\ell}]$. This immediately satisfies the consistency constraints. Notice that $\sum_\ell \mu_j[\ell] = 1$ because of the first moment constraint $\mathbf{Pr}_{Q \sim \lambda_i}[Q(x_j) = \ell] = \mu_j[\ell]$, and summing over all ℓ is imply summing over all possible ℓ in the distribution λ_i. $\square$

Let us investigate into these random variables a bit more. Now we know that the expectation of $\sum_{\ell \in D} Y_{j,\ell}$ is equal to 1 (from the proof of the lemma). But if these random variables make sense at all in this context, we should in fact have that $\sum_{\ell \in D} Y_{j,\ell} = 1$ with probability one! This is simply because these variables represent how the labels are assigned. Indeed, this is true because the variance of $\sum_{\ell \in D} Y_{j,\ell}$ is zero.

Lemma 3.2. *For any j,* $\mathsf{Var}(\sum_{\ell \in D} Y_{j,\ell}) = 0$

Proof. Recall the variance $\mathsf{Var}(\sum_{\ell \in D} Y_{j,\ell}) = \mathbf{E}\left[(\sum_{\ell \in D} Y_{j,\ell})^2\right] - \left(\mathbf{E}\left[\sum_{\ell \in D} Y_{j,\ell}\right]\right)^2$. The second term is one. We can expand the first term as $\mathbf{E}\left[\sum_{\ell \in D} Y_{j,\ell}^2\right] + 2\mathbf{E}\left[\sum_{\ell \neq \ell'} Y_{j,\ell} Y_{j,\ell'}\right]$, where each of the second term is equal to $\mathbf{Pr}_{Q \sim \lambda_i}\left[Q(x_j) = \ell \wedge Q(x_j) = \ell'\right] = 0$; the first term is equal to $\sum_{\ell \in D} \mathbf{Pr}_{Q \sim \lambda_i}\left[Q(x_j) = \ell\right] = 1$. $\square$

Solving the relaxation by SDP: We argue that the existence of such random variables is equivalent to asserting that a certain matrix is PSD. For each j, j', ℓ, ℓ', define $a_{(j,\ell),(j',\ell)}$ as the probability

$$\mathbf{Pr}_{Q \sim \lambda_i}\left[Q(x_j) = \ell \wedge Q(x_{j'}) = \ell'\right] = \sum_{Q : Q(x_j) = \ell, Q(x_{j'}) = \ell'} \lambda_i[Q]$$

We can explicitly write such linear constraints in our convex relaxation. This already represents the second moment constraints, and the only thing we need is to assert that there are random variables whose second moments are exactly $a_{(j,\ell)(j',\ell')}$. This is equivalent to saying that matrix $A = (a_{(j,\ell)(j',\ell')})$ is PSD.

More precisely, the SDP is obtained by simply replacing the term $\mathbf{Pr}_{Q \sim \lambda_i}\left[Q(x_j) = \ell \wedge Q(x_{j'}) = \ell'\right]$ by $a_{(j,\ell)(j',\ell')}$. Other probability terms can be derived easily, e.g. $\mathbf{Pr}\left[Q(x_j) = \ell\right]$ is just $a_{(j,\ell)(j,\ell)}$. If we want to view it as vector programs, we can replace $a_{(j,\ell)(j',\ell')}$ by the inner product $v_{(j,\ell)} \cdot v_{(j',\ell')}$ (introducing vectors $v_{(j,\ell)}$ for all j, ℓ).

3.3 Discussion

This SDP was proposed by Prasad Raghavendra in 2008. It is known to achieve the "best" approximation ratio among any polynomial-time algorithm, assuming UGC. Raghavendra showed that the SDP integrality gap curves give a limit to polynomial time algorithms, while presenting an algorithm that achieves the integrality gap.