Fields Summer School on Approximability of CSPs

Ryan O'Donnell

June 26, 2011

1 Introduction to approximability

These lectures will be about the *approximability* of constraint satisfaction problems (CSPs).

Definition 1.1. Let D be a *domain* of cardinality q. We usually write $D = \{0,1,2,\dots,q-1\}$ and often call the elements of D *labels*. Let Γ be a nonempty finite set of nontrivial *relations (predicates)* over D, with each $R \in \Gamma$ having arity $\mathrm{ar}(R) \le k$. We usually write an r-ary relation as $R : D^r \to \{0,1\}$. Together, D and Γ form an algorithmic problem called CSP(Γ).

Definition 1.2. An *instance* $\mathscr{P}$ of CSP(Γ) over the n *variables* V consists of a *multiset* of m *constraints*. Each constraint $C \in \mathscr{P}$ is a pair (R,S), where $R \in \Gamma$ and S (the *scope* of C) is list of $\mathrm{ar}(R)$ many *distinct*[1] variables from V. We assume that each variable is in the scope of at least one constraint, so $m \ge n/k$.

The algorithmic goal, given an instance $\mathscr{P}$, is to find an assignment for $\mathscr{P}$ which is as "good" as possible.

Definition 1.3. An *assignment* for an instance $\mathscr{P}$ of CSP(Γ) is any mapping $F : V \to D$. The assignment *satisfies* a constraint $C = (R,S)$ if $R(F(S)) = 1$, where the notation $F(S)$ means $(F(S_1),\dots,F(S_r))$. The *value* of the assignment, $\mathrm{Val}_{\mathscr{P}}(F) \in [0,1]$, is the *fraction* of constraints it satisfies; this can be written

$$\mathrm{Val}_{\mathscr{P}}(F) = \operatorname*{avg}_{C=(R,S)\in\mathscr{P}} [R(F(S))].$$

If there is an F such that $\mathrm{Val}_{\mathscr{P}}(F) = 1$ we call F a *satisfying* assignment and say that $\mathscr{P}$ is *satisfiable*. More generally, the *optimum value* of the instance is $\mathrm{Opt}(\mathscr{P}) = \max_F\{\mathrm{Val}_{\mathscr{P}}(F)\}$.

Examples:

- k-Sat: $D = \{0,1\}$, Γ is all disjunctions of up to k literals.
- Ek-Sat: Same, but each disjunction is on *exactly* k literals.
- Horn-k-Sat: Same as k-Sat, but each disjunction contains at most one positive literal.
- Cut: $D = \{0,1\}$, $\Gamma = \{\neq\}$. Usually thought of as partitioning a graph into two pieces, with the goal of having all edges "cut" (crossing the partition).
- q-Cut: $D = \{0,1,\dots,q-1\}$, Γ again just contains binary disequality. Also known as q-Colouring.
- Ek-Lin(q): $D = \mathbb{Z}_q$, Γ is all linear equations of the form $\pm v_1 \pm v_2 \pm \cdots \pm v_k = c$, $c \in \mathbb{Z}_q$.

[1] We will make this not-necessarily-standard assumption.

- 3-CSP: $D = \{0,1\}$, Γ is all relations on up to three variables.
- L-Retraction, where L is a lattice poset: $D = L$, Γ contains the binary relation $\leq$ and all the unary relations $=_\ell$ for $\ell \in L$.
- Bijection(q), AKA Unique-Games(q): $|D| = q$, Γ is all binary *bijections*.
- Projection(q), AKA Label-Cover(q): $|D| = q$, Γ is all binary *projections* R, meaning that there is a $\pi : D \to D$ such that $R = \{(b, \pi(b)) : b \in D\}$.

From the point of view of computational complexity, the first thing to ask about CSP(Γ) is whether there is an efficient (i.e., poly(m)-time) algorithm which, when given a satisfiable instance $\mathscr{P}$, is guaranteed to find a satisfying assignment. E.g., for the Cut problem ($q = 2$, $\Gamma = \{\neq\}$), there is such an algorithm; for the 3-Sat problem, it's NP-hard.

But that is not the end of the story. Say you are given an instance $\mathscr{P}$ of the Cut problem and $\mathrm{Opt}(\mathscr{P}) \neq 1$. You still might want to find an assignment with as large a value as possible. Ideally, you would want to find an assignment achieving optimal value. But, as is well-known, the "Max-Cut" problem is NP-hard.[2] More precisely, if you look at the classic reduction from 3-Sat to Max-Cut you see it proves something like:

Theorem 1.4. *Given an instance $\mathscr{P}$ of Max-Cut with* $\mathrm{Opt}(\mathscr{P}) \geq 3/4$*, it is* NP*-hard to find an assignment F with* $\mathrm{Val}_{\mathscr{P}}(F) \geq 3/4$.

This is still not the end of the story. For example, if there were an efficient algorithm which was guaranteed to find an assignment of value at least 2/3, or even .7499, such an *approximation algorithm* could still be quite worthwhile "in practice". Let's make some definitions to help investigate these issues.

Definition 1.5. For real numbers $0 \leq \alpha \leq \beta \leq 1$, we say an algorithm (α, β)-*approximates* CSP(Γ) (pronounced "α out of β approximates") if it outputs an assignment with value at least α on any input instance with value at least β. (Mnemonic: the αlgorithm gets $\geq \alpha$ when the βest is $\geq \beta$.) We also consider an easier task: we say an algorithm (α, β)-*distinguishes* CSP(Γ) if it outputs 'NO' on instances $\mathscr{P}$ with $\mathrm{Opt}(\mathscr{P}) < \alpha$ and 'YES' on instances $\mathscr{P}$ with $\mathrm{Opt}(\mathscr{P}) \geq \beta$.

Remark 1.6. We prefer to give algorithms for the approximation problem and NP-hardness results for the distinguishing problem. Regarding reductions between distinguishing and approximating, see the exercises. Also note that a single algorithm may (α, β)-approximate CSP(Γ) for many different values of (α, β).

Remark 1.7. We also allow randomized algorithms; in this case we study the *expected* value of the solutions they produce. As an aside, all randomized algorithms known for CSPs are also known to be derandomizable.

For a given CSP, the question of when (α, β)-approximating is in P and when it is NP-hard can be rather fascinating. Here is the state of affairs just for the Max-Cut problem with $\beta = 5/6$:

Theorem 1.8.

1. (3/4, 3/4)-distinguishing Max-Cut is NP-hard. (Karp, 1972.)

[2] We usually add the prefix "Max-" to a CSP when considering it as an optimization problem.

2. $(3/4-\epsilon_0, 3/4)$-distinguishing Max-Cut is NP-hard, where $\epsilon_0 > 0$ is a certain universal constant. (Follows from the "PCP Theorem" of Feige–Goldwasser–Lovasz–Safra–Szegedy 1991, Arora–Safra 1992, Arora–Lund–Motwani–Sudan–Szegedy 1999.)

3. $(.878(3/4), 3/4)$-approximating for Max-Cut is in P. (Goemans–Williamson 1994.) Remark: $.878(3/4) \approx .659$, and here .878 is shorthand for $2/(\pi \sin\theta^*)$, where θ^* is the positive root of $\theta^* = \tan(\theta^*/2)$.

4. $(11/16+\epsilon, 3/4)$-distinguishing for Max-Cut is NP-hard for all $\epsilon > 0$. (Håstad 1997, plus Trevisan–Sorkin–Sudan–Williamson.) Remark: $11/16 = .6875$.

5. $(2/3, 3/4)$-distinguishing Max-Cut is not known to be in P and is not known to be NP-hard.

6. $(.878(3/4)+\epsilon, 3/4)$-distinguishing Max-Cut is in NP-hard for all $\epsilon > 0$, assuming the controversial "Unique-Games Conjecture" ("UGC"). (Khot–Kindler–Mossel–O'Donnell 2004 plus Mossel–O'Donnell–Oleszkiewiz 2005.)

Comparing results 3 and 6, we see an example of "matching" ("optimal") approximability and inapproximability results, assuming "UGC" (and of course $\mathsf{P} \neq \mathsf{NP}$).

Dream Goal: *For each* CSP(Γ) *and each* $\beta \in [0,1]$*, find a number* $\alpha_\Gamma(\beta)$ *and an efficient algorithm which* $(\alpha_\Gamma(\beta), \beta)$*-approximates* CSP($\Gamma$)*; and also, show that* $(\alpha_\Gamma(\beta)+\epsilon, \beta)$*-distinguishing* CSP($\Gamma$) *is* NP*-hard for all* $\epsilon > 0$.

The Dream Goal is completely accomplished for a few CSPs (e.g., k-Sat, k-Lin(q) for $k \geq 3$) and accomplished assuming the UGC for others (e.g., Max-Cut). Most remarkably, assuming the UGC it is "essentially" accomplished "in principle" for all CSPs (Raghavendra 2008). On the other hand, it not completely inconceivable that the Dream Goal is *invalid*; i.e., that there are Γ, α, β such that (α, β)-distinguishing CSP(Γ) is neither in P nor NP-hard.

This mini-course will show techniques for obtaining both approximating algorithms and hardness-of-approximation results for CSPs.

2 Slight approximation and inapproximability

Let us briefly get our bearings by discussing the most basic results.

2.1 Slight inapproximability

The famous (and difficult) "PCP Theorem" is equivalent to the statement that $(1-\epsilon_0, 1)$-distinguishing Max-3-Sat is NP-hard for some $\epsilon_0 > 0$. Given this result and any "gadget"-based reduction from 3-Sat to Max-CSP(Γ), one may deduce that Max-CSP(Γ) is "APX-hard":

Definition 2.1. We say Max-CSP(Γ) is APX-*hard* if there exists $\beta \in [0,1]$ and $\epsilon_0 > 0$ such that $(\beta - \epsilon_0, \beta)$-distinguishing CSP($\Gamma$) is NP-hard.

In light of this fact, for every CSP where the exact optimization problem is currently known to be NP-hard[3] we also have some slight inapproximability result. Still, the following problem is open:

[3] And this is almost all interesting CSPs.

Open problem. *Show the following dichotomy theorem: For every* CSP(Γ)*, either the optimization problem is in* P *or else* CSP(Γ) *is* APX*-hard.*

This dichotomy is known to hold in several cases, including $|D| = 2$ (Creignou 1995, see also Khanna–Sudan–Williamson 1997), $|D| = 3$ (Jonsson–Klasson–Krokhin 2006), whenever Γ includes all unary constraints (Deineko–Jonsson–Klasson–Krokhin 2008), and whenever Γ contains only one relation (Jonsson–Krokhin–Kuivinen 2008).

2.2 Slight approximability

Some CSP optimization problems seem very tough; e.g., Max-EkSat for $k \geq 3$. However there is always a "baseline" you can achieve using a ridiculous-seeming algorithm:

The Trivial Random algorithm: *Choose $F(v) \in D$ randomly and independently for each $v \in V$.*

Note that the Trivial Random algorithm doesn't even really look at the constraints, and its performance on instance $\mathscr{P}$ is completely independent of Opt($\mathscr{P}$)!

The (expected) value produced by the Trivial Random algorithm is always easily to analyze. E.g.:

Proposition 2.2. *The Trivial Random algorithm $(1-2^{-k},\beta)$-approximates Max-EkSat (for every β).*

Proof. Any constraint (clause) in an EkSat instance has 1 unsatisfying assignment and $2^k - 1$ satisfying ones. Thus each fixed constraint is satisfied with probability $1-2^{-k}$ by the Trivial Random algorithm. From Linearity of Expectation, it follows that the expected fraction of satisfied constraints is exactly $1-2^{-k}$. □

We have only a worse guarantee for the Trivial Random algorithm on Max-kSat: $(1/2,\beta)$-approximation. This is because all of the clauses may have width 1. (This is somewhat peculiar: the Trivial Random algorithm does worse the more the instance includes "seemingly easier" short clauses. We'll come back to this in the next section.)

Proposition 2.3. *The Trivial Random algorithm $(1/2,\beta)$-approximates Max-Cut, $(1/q,\beta)$-approximates Max-k-Lin(q), and $(1/q,\beta)$-approximates Unique-Games(q).*

Proof. Exercise. □

2.3 The Dream Goal attained

Remarkably, sometimes the Trivial Random algorithm fulfills the Dream Goal! By results of Håstad (1997), this is the case for both Max-E3Sat and Max-E3Lin(q). E.g., in later lectures we will sketch Håstad's proof of the following theorem:

Theorem 2.4. *$(1/2+\epsilon, 1-\epsilon)$-distinguishing E3-Lin(2) is* NP*-hard for all $\epsilon > 0$.*

This result fulfills the Dream Goal — it is optimal assuming P $\neq$ NP. This is because the Trivial Random algorithm efficiently $(1/2,\beta)$-approximates E3-Lin(2) (and also because $(1,1)$-approximating E3-Lin(2) is in P, see exercises).

3 Linear programming

Linear programming (LP) is one of the most powerful tools for approximation algorithms — not just for CSPs, but for many other kinds of optimization tasks. Herein we will describe what we call the *Basic LP relaxation* for CSPs.

Let $\mathscr{P}$ be an instance of some CSP(Γ) over domain D. To understand the Basic LP relaxation for $\mathscr{P}$, we first reformulate the algorithmic task associated to $\mathscr{P}$ as a "0-1 Integer Program (IP)".

Basic 0-1 Integer Program (IP) for $\mathscr{P}$. The IP has a number of "variables"[4] which are supposed to take value 0 or 1. First we introduce *variable indicators* $\mu_v[\ell]$:

$$\forall v \in V, \quad \forall \ell \in D, \quad \mu_v[\ell] \in \{0,1\}. \tag{1}$$

The "interpretation" of $\mu_v[\ell] = 1$ is that variable v is assigned label ℓ. Accordingly, the IP includes the condition

$$\forall v \in V, \quad \sum_{\ell \in D} \mu_v[\ell] = 1. \tag{2}$$

Next we introduce the *constraint indicators* $\lambda_C[L]$:

$$\forall C = (R,S) \in \mathscr{P}, \quad \forall L : S \to D, \quad \lambda_C[L] \in \{0,1\}. \tag{3}$$

Again, the "interpretation" of $\lambda_C[L]$ is that L is "local" partial assignment to the variables in S. We also include the condition

$$\forall C = (R,S) \in \mathscr{P}, \quad \sum_{L : S \to D} \lambda_C[L] = 1. \tag{4}$$

Naturally, we want "consistency" between the variable indicators and the constraint indicators, so we also add the following *consistency conditions*:

$$\forall C = (R,S) \in \mathscr{P}, \quad \forall v \in S, \quad \forall \ell \in D, \quad \sum \Big\{ \lambda_C[L] : L(v) = \ell \Big\} = \mu_v[\ell]. \tag{5}$$

We remark (see exercises) that given (5), the conditions (1), (2) are superfluous. Finally, the objective is to maximize the fraction of satisfied constraints:

$$\text{maximize} \quad \operatorname*{avg}_{C=(R,S) \in \mathscr{P}} \sum \Big\{ \lambda_C[L] : L(S) \text{ satisfies } R \Big\}. \tag{6}$$

You should convince yourself that this IP — i.e., maximizing in (6) subject to (1–5) — is identical to the task of finding the optimal assignment for $\mathscr{P}$.

Of course, that means solving the Basic IP to optimality is NP-complete for any interesting CSP. However we can *relax* the IP to a *linear program* (LP) which can be solved optimally and efficiently (in theory and in practice).

[4] These are different from the variables V of $\mathscr{P}$; it's an unfortunate terminology clash.

Basic LP for $\mathscr{P}$. This is the same as the Basic IP except the variables are only constrained to be in $[0,1]$ rather than $\{0,1\}$. Then it's natural to think of groups as variables as forming *probability distributions*. More precisely, an *LP solution* $\mathscr{L}$ consists of *constraint distributions* $(\lambda_C)_{C\in\mathscr{P}}$ and *variable distributions* $(\mu_v)_{v\in V}$; the Basic LP relaxation is:

$$\text{maximize} \quad \text{LPVal}_{\mathscr{P}}(\mathscr{L}) := \operatorname*{avg}_{C=(R,S)\in\mathscr{P}} \left\{ \mathop{\mathbf{Pr}}_{L\sim\lambda_C}[L(S) \text{ satisfies } R] \right\},$$

$$\begin{aligned} \text{subject to} \quad & \lambda_C \text{ is a prob. dist. on local assignments } S \to D && \forall C=(R,S)\in\mathscr{P}, \\ & \mu_v \text{ is a probability distribution on } D && \forall v \in V, \\ & \mathop{\mathbf{Pr}}_{L\sim\lambda_C}[L(v)=\ell] = \mathop{\mathbf{Pr}}_{\mu_v}[\ell] && \forall C=(R,S)\in\mathscr{P}, \quad \forall v\in S, \quad \forall \ell \in D. \end{aligned}$$

We call the last constraint the *consistent marginals* condition.

Observe that this LP has at most $q^k m + qn$ real-valued variables, and they are subject to linear equalities and inequalities. Thus as mentioned, it can be solved optimally in polynomial time. We call the optimal value of the Basic LP $\text{LPOpt}(\mathscr{P}) \in [0,1]$. Since it is a relaxation of the Basic IP (which captured the CSP(Γ) problem exactly), we always have

$$\text{Opt}(\mathscr{P}) \le \text{LPOpt}(\mathscr{P}).$$

The hope, though, is that $\text{LPOpt}(\mathscr{P})$ is never *too* much bigger, and that an optimal solution $\mathscr{L}$ to the Basic LP can be used as a guide to find an actual assignment for $\mathscr{P}$ which is almost as good.

3.1 Example: Max-Sat

Let's spend some time on an illustrative example: Max-Sat.

Let $\mathscr{P}$ be an instance of Max-Sat and let $\mathscr{L} = ((\mu_v)_{v\in V}, (\lambda_C)_{C\in\mathscr{P}})$ be an optimal solution to the Basic LP relaxation. Each μ_v is a probability distribution on $D = \{0,1\}$. We are hoping to use these distributions to guide us to an actual assignment $F : V \to \{0,1\}$. The most obvious thing to try is

randomized rounding: choose $F(v)$ according to μ_v, independently for each $v \in V$.

We evaluate the performance of this algorithm by computing the expectation (over the choice of F) of the fraction of constraints satisfied. Consider some constraint (clause) C; say,

$$C = x_2 \vee x_5 \vee \overline{x}_9.$$

Then

$$\mathbf{Pr}[F \text{ satisfies } C] = 1 - \mu_{x_2}[0] \cdot \mu_{x_5}[0] \cdot \mu_{x_9}[1].$$

More generally, say that C has scope S and that $b_C : S \to \{0,1\}$ is the "bad" (forbidden) assignment for C. Then

$$\mathbf{Pr}[F \text{ satisfies } C] = 1 - \prod_{v\in S} \mu_v[b_C(v)]. \tag{7}$$

Our task is to relate this quantity to the contribution from C to $\text{LPVal}_{\mathscr{P}}(\mathscr{L})$.

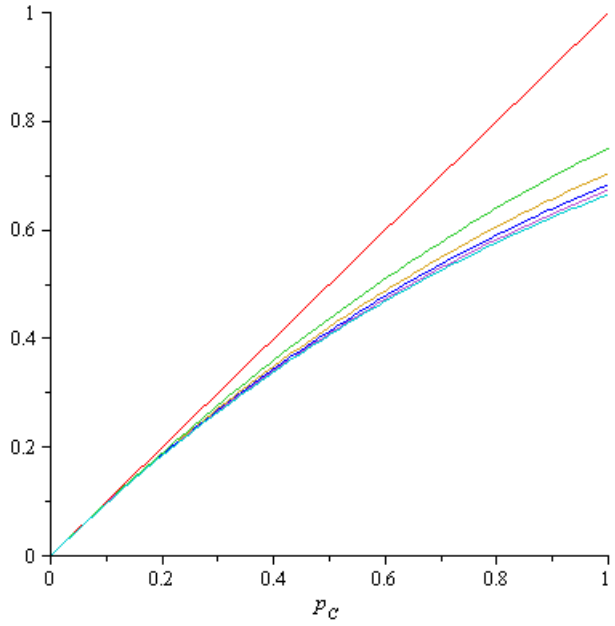
Given C with scope S, let's look at how the constraint distributions λ_C contribute to $\mathrm{LPVal}_{\mathscr{P}}(\mathscr{L})$:

$$\begin{aligned} p_C := \mathop{\mathbf{Pr}}_{L\sim\lambda_C}[L(S) \text{ satisfies } R] &\le \sum_{v\in S} \mathop{\mathbf{Pr}}_{L\sim\lambda_C}[L(v) \neq b_C(v)] && \text{(union bound)} \\ &= \sum_{v\in S}(1-\mu_v[b_C(v)]) && \text{(consistent marginals condition).} \end{aligned} \tag{8}$$

To compare this with (7) we use the AM-GM bound:

$$\begin{aligned} \mathbf{Pr}[F \text{ satisfies } C] = 1 - \prod_{v\in S}\mu_v[b_C(v)] &= 1 - \mathop{\mathrm{GM}}_{v\in S}\{\mu_v[b_C(v)]\}^{|S|} \\ &\ge 1 - \mathop{\mathrm{AM}}_{v\in S}\{\mu_v[b_C(v)]\}^{|S|} \\ &= 1 - (1 - \mathop{\mathrm{AM}}_{v\in S}\{1-\mu_v[b_C(v)]\})^{|S|} \\ &\ge 1 - (1-p_C/|S|)^{|S|} && \text{(by (8))} \\ &\ge (1-(1-1/|S|)^{|S|})p_C, \end{aligned}$$

where the last step is an exercise, using concavity.



For every number $|S|$ it holds that $(1-1/|S|)^{|S|} \le 1/e$; hence

$$\mathbf{Pr}[F \text{ satisfies } C] \ge (1-1/e)p_C.$$

Thus we conclude that the expected value of the randomized rounding algorithm is

$$\mathbf{E}[\mathrm{Val}_{\mathscr{P}}(F)] = \mathop{\mathrm{avg}}_{C\in\mathscr{P}}\Big\{\mathbf{Pr}[F \text{ satisfies } C]\Big\} \ge (1-1/e)\mathop{\mathrm{avg}}_{C\in\mathscr{P}}\{p_C\} = (1-1/e)\mathrm{LPVal}_{\mathscr{P}}(\mathscr{L}) = (1-1/e)\mathrm{LPOpt}(\mathscr{P}).$$

Hence:

Theorem 3.1. *Randomized rounding gives an* $((1-1/e)\beta,\beta)$*-approximation for Max-Sat for all* β*.*

Since $1-(1-1/|S|)^{|S|}$ is a decreasing function of $|S|$, we can additionally conclude:

Theorem 3.2. *Randomized rounding gives an* $((1-(1-1/k)^k)\beta,\beta)$*-approximation for Max-*k*Sat for all* β*. E.g., it is a* $(\frac{3}{4}\beta,\beta)$*-approximation for Max-*2*Sat.*

In fact, you might notice something a bit strange: The LP-rounding algorithm for Max-Sat, which assigns 1 to each v with probability $\mu_v(1)$, obtains value at least $\frac{3}{4}$LPOpt as long as all clauses have width *at most* 2. On the other hand the trivial random algorithm, which assigns 1 to each v with probability 1/2, achieves value at least $\frac{3}{4}$ outright as long as all clauses have width *at least* 2. Can't we achieve the best of both worlds? The answer is yes:

Theorem 3.3. *Let* $\mathscr{P}$ *be a Max-Sat instance with optimal LP solution* $\mathscr{L} = ((\mu_v)_{v\in V},(\lambda_C)_{C\in\mathscr{P}})$*. Suppose we assign* $F(v) = 1$ *with probability* $\frac{1}{2}+\frac{1}{2}\mu_v(1)$*, independently for each* v*. Then* $\mathbf{E}[\mathrm{Val}_{\mathscr{P}}(F)] \geq \frac{3}{4}\mathrm{LPOpt}(\mathscr{P})$*, so this is a* $(\frac{3}{4}\beta,\beta)$*-approximation for Max-Sat.*

Proof. Exercise. □

Although these algorithms are quite good at approximating Max-Sat, they are not the best known. The best known results all use sophisticated *semidefinite programming (SDP)* techniques (a topic we will discuss later):

Theorem 3.4.

- *There is an efficient* $(\alpha_2(\beta),\beta)$*-approximation algorithm for Max-*2*-Sat satisfying* $\alpha_2(\beta) \geq .940\beta$ *(Lewin–Livnat–Zwick 2002) and* $\alpha_2(1-\epsilon) = 1-O(\sqrt{\epsilon})$ *(Charikar–Makarychev–Makarychev 2006).*
- *There is an efficient* $(\frac{7}{8}\beta,\beta)$*-approximation algorithm for Max-*3*-Sat (Karloff–Zwick 1996, Zwick 2002). (Recall that this is trivial for Max-E*3*-Sat! This result fulfills the Dream Goal for Max-*3*-Sat.)*
- *There is an efficient* $(\frac{7}{8},1)$*-approximation algorithm for Max-*4*-Sat (Halperin–Zwick 1999).*

It might be conjectured that there is an efficient $(\frac{7}{8}\beta,\beta)$-approximation algorithm for Max-Sat, but this is an open problem. The best result known has .833 in place of $\frac{7}{8}$ (Asano–Williamson 2000).

3.2 Connections to algebra

The step in LP-based approximation algorithms that requires creativity is deciding out how to use the Basic LP solution as a "guide" in coming up with an actual assignment of comparable value. This is often referred to as coming up with a "rounding function" to apply to the variable distributions μ_v.

Here is where there definitely fruitful connections with the algebraic theory of CSPs: in many cases, the *polymorphisms* for CSP(Γ) can be translated into good rounding functions. One example of this was demonstrated in recent (unpublished) work:

Theorem 3.5. *(Kun–O'Donnell–Zhou 2011, Tamaki–Yoshida 2011) Let* CSP(Γ) *have "width 1" in the algebraic sense; i.e., suppose it has a totally symmetric polymorphism. Then the Basic LP is a* robust satisfiability distinguisher *for* CSP(Γ)*; specifically, it is a* $(1-O(1/\log(1/\epsilon)),1-\epsilon)$*-distinguishing algorithm for all* $\epsilon > 0$*.*

One proof of this theorem involves translating the totally symmetric polymorphism into a "rounding function". Similarly:

Theorem 3.6. *(Kun–O'Donnell–Zhou 2011) Let* CSP(Γ) *be a "lattice CSP" (e.g., any lattice-retraction CSP). Then the Basic LP is an even better robust satisfiability distinguisher for* CSP(Γ)*: it is a* $(1-O(\epsilon),1-\epsilon)$*-distinguishing algorithm for all* $\epsilon > 0$*.*

Again, the proof involves translating a polymorphism into a rounding function.

3.3 LP gaps

Theorem 3.3 shows that for Max-Sat, $\mathrm{LPOpt}(\mathscr{P})$ is never more than $\frac{4}{3}$ times $\mathrm{Opt}(\mathscr{P})$; i.e., the Basic LP relaxation is somewhat tight for Max-Sat. Unfortunately, this is as tight as it gets, in general. We can prove this by exhibiting an *LP gap instance*.

Definition 3.7. An (α,β)-*gap instance* for the Basic LP relaxation of Max-CSP(Γ) is an instance $\mathscr{P}$ with $\mathrm{Opt}(\mathscr{P}) \le \alpha$ and $\mathrm{LPOpt}(\mathscr{P}) \ge \beta$.

The existence of an (α,β)-gap instance is equivalent to the statement that the Basic LP algorithm is *not* and (α,β)-distinguishing algorithm. In particular it means that there is probably no (α,β)-approximation algorithm "based on" the Basic LP.

Here we give an example for Max-Sat — indeed, for Max-E2-Sat:

Proposition 3.8. *The following is a* $(\frac{3}{4},1)$*-gap instance for the Basic LP relaxation of Max-E2-Sat:* $\mathscr{P} = \{v \vee w, v \vee \overline{w}, \overline{v} \vee w, \overline{v} \vee \overline{w}\}$.

Definition 3.9. It's clear that no assignment can satisfy all four clauses; hence $\mathrm{Opt}(\mathscr{P}) \le \frac{3}{4}$. On the other hand, here is an LP solution with LP-value 1: Take both μ_v and μ_w to be the 1/2-1/2 distribution on $\{0,1\}$. It remains to show that for each of the four clauses C, we can choose λ_C supported on the satisfying assignments and with 1/2-1/2 marginals on both v and w. If, e.g., $C = v \vee w$, we can take λ_C to have weight 1/2 on the satisfying assignment '$v = 1, w = 0$' and weight 1/2 on the satisfying assignment '$v = 0, w = 1$'. We leave the remaining 3 cases as exercises.

This is an "optimal" gap instance for Max-E2-Sat, in the sense that the gap cannot be made wider by virtue of the algorithm described in Theorem 3.3. Thus approximation factor $\frac{3}{4}$ cannot really be beaten by an algorithm based on Basic LP. As mentioned, however, it *can* be beaten using a more sophisticated SDP algorithm.

The following example shows that the Basic LP is completely useless for the Max-Cut problem.

Proposition 3.10. *Let* $\mathscr{G}$ *be any graph. Then* $\mathrm{LPVal}(\mathscr{G}) = 1$.

Proof. Take the LP solution in which $\mu_v[0] = \mu_v[1] = 1/2$ for all $v \in V$ and $\lambda_{(u,v)}[0,1] = \lambda_{(u,v)}[1,0] = 1/2$ for all edges $(u,v) \in E$. □

In particular, any graph $\mathscr{G}$ with $\mathrm{Opt}(\mathscr{G}) = \alpha$ is an $(\alpha,1)$-gap instance. One can make α arbitrarily close to 1/2 (exercise).

4 Semidefinite programming

Semidefinite programming (SDP) is a generalization of linear programming. Later we will present the *Basic SDP* relaxation for CSPs, which in most cases can lead to improved approximation algorithms. As the ideas involved are more complicated, we begin with a very special case which already yields big improvements: the SDP relaxation for the Max-Cut problem. (Recall that the Basic LP relaxation for the Max-Cut problem is completely useless.)

4.1 The Max-Cut SDP

Let $\mathscr{G}$ be an instance of the Max-Cut CSP; we can think of $\mathscr{G}$ simply as an undirected graph $\mathscr{G} = (V, E)$. The following is the *Goemans–Williamson (GW)* SDP relaxation (actually proposed originally by Delorme and Poljak) for the Max-Cut problem $\mathscr{G}$:

$$\text{maximize} \quad \text{SDPVal}_{\mathscr{G}}(\mathscr{X}) := \operatorname*{avg}_{(v,w)\in E} \left\{\tfrac{1}{2} - \tfrac{1}{2}\langle X_v, X_w\rangle\right\} \tag{9}$$
$$\text{s.t.} \quad \mathscr{X} = (X_v)_{v\in V} \text{ is a collection of } n\text{-dimensional } \textit{vectors} \text{ with } \langle X_v, X_v\rangle = \|X_v\|_2^2 = 1 \ \forall v \in V.$$

Roughly speaking, the optimization problem is to embed the vertices of the graph onto the unit sphere (in high dimensions) so that edges of the graph are "spread out" as much as possible. (Here $\langle X_v, X_w\rangle$ is the inner product between the unit vectors X_v and X_w, which is in the range $[-1,1]$.)

What is rather astonishing is that although computing the optimum value $\text{SDPOpt}(\mathscr{G})$ of this optimization problem looks very complicated, there is actually an efficient algorithm which (essentially) does it. We will explain why shortly; for now, let's verify that the GW SDP is indeed a relaxation:

Proposition 4.1. $\text{Opt}(\mathscr{G}) \le \text{SDPOpt}(\mathscr{G})$.

Proof. Interpret the domain D of the Max-Cut problem as $\{-1,1\}$ rather than $\{0,1\}$ and let $F : V \to \{-1,1\}$ be an optimal cut in the graph $\mathscr{G}$. Now define the SDP solution $\mathscr{X} = (X_v)_{v\in V}$ simply by letting X_v be the vector $(F(v), 0, 0, \ldots, 0)$. This indeed has $\|X_v\|_2^2 = 1$, and it also has $\langle X_v, X_w\rangle = F(v)F(w)$. Hence

$$\text{SDPVal}_{\mathscr{G}}(\mathscr{X}) = \operatorname*{avg}_{(v,w)\in E}\left\{\tfrac{1}{2} - \tfrac{1}{2}F(v)F(w)\right\} = \operatorname*{avg}_{(v,w)\in E}\begin{Bmatrix} 1 & \text{if } F(v) \ne F(w), \\ 0 & \text{if } F(v) = F(w). \end{Bmatrix} = \text{Val}_{\mathscr{G}}(F) = \text{Opt}(\mathscr{G}). \qquad \square$$

Digression: solvability of the SDP. Let us now briefly justify why the GW SDP is actually polynomial-time solvable. The key is that it is actually a linear programming problem in disguise. First, observe that the objective function and the constraints of the SDP aren't really about the vectors per se; rather, they only involve the *inner products* $\rho_{vw} = \langle X_v, X_w\rangle$ between the vectors. We could write the SDP as:

$$\begin{aligned} \text{maximize} \quad & \operatorname*{avg}_{(v,w)\in E}\left\{\tfrac{1}{2} - \tfrac{1}{2}\rho_{vw}\right\} \\ \text{s.t.} \quad & \rho_{vv} = 1 \quad \forall v \in V, \\ & \text{the matrix } P := (\rho_{vw})_{v,w\in V} \text{ satisfies } P = XX^\top \text{ for some } n \times n \text{ matrix } X. \end{aligned} \tag{10}$$

The rows of matrix X correspond to the vectors X_v. Notice that this is a linear program over the n^2 variables ρ_{vw}, except that there is the extra condition (10). If you know a little linear algebra then you know that (10) is equivalent to saying that the matrix P is *positive semidefinite* (which explains the name "semidefinite programming"). Note that if $P = XX^\top$ for some matrix X, then for all row vectors $y \in \mathbb{R}^V$,

$$\sum_{v,w} y_v y_w \rho_{vw} = yPy^\top = yXX^\top y^\top = (yX)(yX)^\top = \langle yX, yX\rangle \ge 0. \tag{11}$$

In fact, the converse is also true. The following is a basic theorem from linear algebra:

Theorem 4.2. *An $n \times n$ symmetric real matrix P is called* positive semidefinite *if any of the following equivalent conditions holds:*

- $yPy^\top \geq 0$ *for all* $y \in \mathbb{R}^n$.
- *There exists an* $n \times n$ *matrix* X *such that* $P = XX^\top$.
- *All of the eigenvalues of* P *are nonnegative.*

By virtue of this theorem, we can replace condition (10) in the SDP formulation with the condition that (11) holds for all $y \in \mathbb{R}^V$. Notice that this is simply a linear inequality on the variables ρ_{vw} for all fixed $y \in \mathbb{R}^V$. Hence the SDP is equivalent to a linear program... with infinitely many constraints. Nevertheless, the theory of linear programming says we can efficiently solve such an LP[5] so long as there is an efficient "separation oracle". This means an algorithm which, when given a candidate setting of the variables $P = (\rho_{vw})_{v,w}$, either outputs that it satisfies all linear inequalities or else finds an unsatisfied inequality. A separation oracle for the SDP condition can be implemented by (efficiently) computing the eigenvalues of P: if they are all nonnegative then P is positive semidefinite; otherwise, any eigenvector y corresponding to a negative eigenvalue yields an unsatisfied inequality in (11).

Using the SDP solution to produce a cut. As with linear programming, we would now like to use an optimal solution $\mathscr{X} = (X_v)_{v \in V}$ to the GW SDP relaxation for $\mathscr{G}$ to produce a good cut. Bearing in mind that the SDP tries to "spread apart" the pairs of vectors corresponding to edges $(v,w) \in \mathscr{G}$, Goemans and Williamson suggested the following "rounding algorithm":

GW Rounding: *Pick a random hyperplane through the origin in* $\mathbb{R}^n$*, partition the vertices* v *according to which side of the hyperplane* X_v *is on.*

Before analyzing this as an approximation algorithm, let's first make sense of what it means to pick a random hyperplane. By this we mean we want the normal vector Z for the hyperplane to have a rotationally symmetric distribution. In practice and in theory, the best way to do this is to make the components $Z_1, \dots, Z_n$ independent random *Gaussians*. (The pdf of a Gaussian is $(2\pi)^{-1/2}\exp(-z^2/2)$, so the pdf of the n-dimensional Gaussian Z is $(2\pi)^{-n/2}\exp(-(z_1^2+\cdots+z_n^2)/2)$; this is indeed rotationally symmetric since it only depends on $\|z\|_2^2 = z_1^2+\cdots+z_n^2$.) Thus we can state the GW rounding algorithm a bit more precisely:

1. Choose $Z \in \mathbb{R}^n$ to be a random n-dimensional Gaussian, a rotationally symmetric distribution.
2. Output the Max-Cut assignment defined by $F(v) = \mathrm{sgn}(\langle Z, X_v \rangle)$ for all $v \in V$.

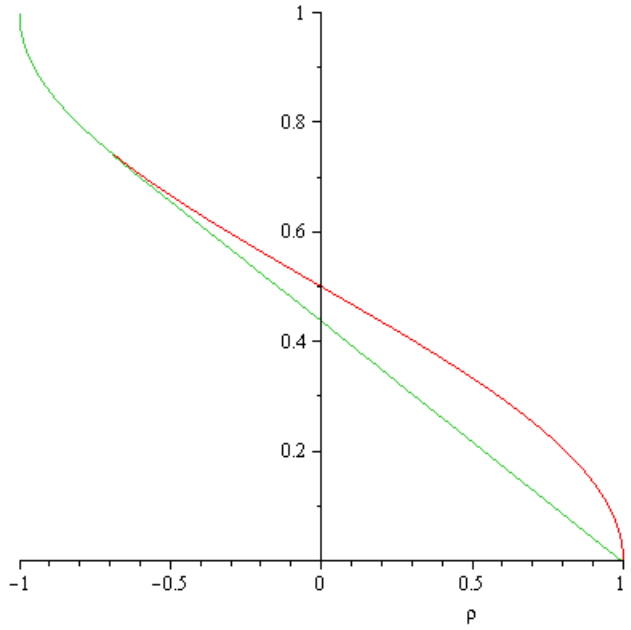
Analysis of the GW algorithm. As with our LP analyses we work on an edge-by-edge basis. Let $(v,w) \in \mathscr{G}$ and suppose that $\langle X_v, X_w \rangle = \rho_{vw}$. By the rotational symmetry of Z, the question of whether the hyperplane perpendicular to Z "splits" X_v and X_w reduces to a two-dimensional problem. [DRAW PICTURE] I.e., for the analysis we may assume X_v and X_w lie on the unit circle in $\mathbb{R}^2$, and the random hyperplane is just a uniformly random diameter of the circle. Now it's clear that

$$\mathbf{Pr}[\text{probability } X_v, X_w \text{ split}] = \frac{\text{angle}(X_v, X_w)}{\pi} = \arccos(\rho_{vw})/\pi;$$

hence

$$\mathbf{E}[\mathrm{Val}_{\mathscr{G}}(F)] = \operatorname*{avg}_{(v,w)\in E}\{\arccos(\rho_{vw})/\pi\}.$$

[5] Up to accuracy $\pm\epsilon$ in the objective and in satisfying the conditions, in time $\mathrm{poly}(n, \log(1/\epsilon))$. We will henceforth ignore the issue of imperfect accuracy, as it is not hard to show it has a negligible affect on all subsequent results.



The most careful way to complete the analysis is the following. Define $A(\rho)$ to be the "lower convex envelope" of the function $\arccos(\rho)/\pi$: it is the green plot above. To three places of precision, it's

$$A(\rho) = \begin{cases} \arccos(\rho)/\pi & \text{if } \rho \le -.690, \\ -\frac{.878}{2}\rho + \frac{.878}{2} & \text{if } \rho \ge -.690. \end{cases}$$

Thus by convexity,

$$\mathbf{E}[\mathrm{Val}_{\mathscr{G}}(F)] = \operatorname*{avg}_{(v,w)\in E}\{\arccos(\rho_{vw})/\pi\} \ge \operatorname*{avg}_{(v,w)\in E}\{A(\rho_{vw})\} \ge A\left(\operatorname*{avg}_{(v,w)\in E}\rho_{vw}\right).$$

But recall that since we started with an optimal SDP solution,

$$\operatorname*{avg}_{(v,w)\in E}\left\{\tfrac{1}{2} - \tfrac{1}{2}\rho_{vw}\right\} = \mathrm{SDPOpt}(\mathscr{G});$$

thus we conclude:

Theorem 4.3. *(Goemans–Williamson.) The SDP rounding algorithm described achieves*

$$\mathbf{E}[\mathrm{Val}_{\mathscr{G}}(F)] \ge A\,(1 - 2\mathrm{SDPOpt}(\mathscr{G})).$$

In particular, it is a $(.878\beta, \beta)$*-approximation algorithm for Max-Cut, and also a* $(1-(2/\pi)\sqrt{\epsilon}, 1-\epsilon)$*-approximation for* $\epsilon \le .155$*.*

$$\mathbf{E}[\mathrm{Val}_{\mathscr{G}}(F)] \ge A\left(\tfrac{1}{2} - \tfrac{1}{2}\mathrm{SDPOpt}(\mathscr{G})\right).$$

4.2 SDP gaps

Just as with the Basic LP, we can ask about how tight the GW SDP relaxation is for the Max-Cut problem. Unfortunately, at least for $\beta \ge .845$, the Goemans–Williamson theorem is optimal:

Theorem 4.4. *(Feige–Schechtman 2000) Let $\beta \geq .845$. Then for any $\epsilon > 0$, there is a Max-Cut instance $\mathscr{G}$ with* $\mathrm{SDPOpt}(\mathscr{G}) \geq \beta - \epsilon$ *but* $\mathrm{Opt}(\mathscr{G}) \leq \arccos(1-2\beta)/\pi + \epsilon$.

Proof. (Sketch.) We consider a graph $\mathscr{G}_n$ whose vertices are the points of the unit sphere in $\mathbb{R}^n$, $\{v \in \mathbb{R}^n : \|v\|_2^2 = 1\}$. (This is actually an infinite graph, but what we describe can be "discretized".) We put an edge between v and w if and only if $\langle v, w \rangle \leq 1 - 2\beta$. (Again, this is actually a "measure" on edges.)

First we want to check that $\mathrm{SDPOpt}(\mathscr{G}) \geq \beta - \epsilon$. We consider the simplest possible SDP solution $\mathscr{S}$: the vector X_v for variable v is simply v itself. Then

$$\mathrm{SDPVal}_{\mathscr{G}}(\mathscr{S}) = \operatorname*{avg}_{(v,w)\in E} \left\{ \tfrac{1}{2} - \tfrac{1}{2}\langle X_v, X_w \rangle \right\}.$$

Certainly this is *at most* β, since $\langle X_v, X_w \rangle \geq 1 - 2\beta$ always. But actually, it is also *at least* $\beta - o_n(1)$, because in high dimensions, "almost all" the edge measure is "concentrated" around inner-product $1 - 2\beta$.

On the other hand, we want to check that $\mathrm{Opt}(\mathscr{G}) \leq \arccos(1-2\beta)/\pi + \epsilon$. The key claim is:

Claim 4.5. *The optimal cuts in $\mathscr{G}_n$ are precisely the ones given by hyperplanes through the origin.*

By rotational symmetry, all hyperplanes through the origin give the same cut value. Furthermore, this cut value is indeed $\arccos(1-2\beta)/\pi + o_n(1)$: that's by the analysis of the Goemans–Williamson rounding algorithm, since picking a (random) hyperplane cut is exactly what it does.

So it remains to justify Claim 4.5. This follows almost immediately from a "rearrangement inequality" due to Baernstein and Taylor (1976); the proof is a "symmetrization" argument. □

4.3 The Basic SDP relaxation for general CSPs

We close by describing the "Basic SDP" relaxation for general CSPs. It is a direct generalization of the Basic LP relaxation.

Basic SDP for $\mathscr{P}$. An *SDP solution* $\mathscr{S}$ consists of two parts. The first part is *constraint distributions* $(\lambda_C)_{C\in\mathscr{P}}$, exactly as in the Basic LP. The second part is a collection of *jointly distributed random variables* (over some probability space) which we call *pseudoindicator random variables* $(I_v[\ell])_{v\in V, \ell\in D}$. The condition linking these two parts is what we call the *consistent first and second moment conditions*. To be precise, the Basic SDP relaxation is:

$$\text{maximize} \quad \mathrm{SDPVal}_{\mathscr{P}}(\mathscr{S}) := \operatorname*{avg}_{C=(R,S)\in\mathscr{P}} \left\{ \mathop{\mathbf{Pr}}_{L\sim\lambda_C} [L(S) \text{ satisfies } R] \right\}, \tag{12}$$

$$\text{s.t.} \quad \lambda_C \text{ is a prob. dist. on local assignments } S \to D \qquad \forall C = (R,S) \in \mathscr{P}, \tag{13}$$

$$\mathop{\mathbf{Pr}}_{L\sim\lambda_C} [L(v) = \ell] = \mathbf{E}\Big[I_v[\ell] \Big] \qquad \forall C = (R,S) \in \mathscr{P}, \quad \forall v \in S, \quad \forall \ell \in D, \tag{14}$$

$$\mathop{\mathbf{Pr}}_{L\sim\lambda_C} [L(v) = \ell \text{ and } L(v') = \ell'] = \mathbf{E}\Big[I_v[\ell] \cdot I_{v'}[\ell'] \Big] \qquad \forall C, \quad \forall v, v' \in S \text{ not nec. distinct}, \quad \forall \ell, \ell' \in D \text{ not nec. distinct}. \tag{15}$$

As always we define $\mathrm{SDPOpt}(\mathscr{P})$ to be the maximum value of $\mathrm{SDPVal}_{\mathscr{P}}(\mathscr{S})$. It is quite easy to see that the Basic SDP is indeed a relaxation of the CSP(Γ) optimization problem and that it is at least as "tight" as the Basic LP: i.e.,

$$\mathrm{Opt}(\mathscr{P}) \le \mathrm{SDPOpt}(\mathscr{P}) \le \mathrm{LPOpt}(\mathscr{P}). \tag{16}$$

We leave this as an exercise. Let us make two more remarks, the justification of which we also leave to the exercises:

Remark 4.6. In any Basic SDP solution, we have that for all $v \in V$, the random variable $\sum_{\ell \in D} I_v[\ell]$ is $\mathbf{1}$, the *constantly* 1 random variable.

Remark 4.7. If we drop the "consistent first moment" condition (14) from the Basic SDP, we get an *equivalent* formulation. I.e., any solution $\mathscr{S}$ to the Basic SDP which doesn't satisfy (14) can be transformed into a solution $\mathscr{S}'$ which *does* satisfy (14) and has $\mathrm{SDPVal}_{\mathscr{P}}(\mathscr{S}') = \mathrm{SDPVal}_{\mathscr{P}}(\mathscr{S})$.

In some ways this optimization problem looks even *more* complicated than the Goemans–Williamson SDP relaxation for Max-Cut: here we are optimizing over collections of joint random variables. A first question is how an algorithm would even *represent* such a collection. One natural answer is: *by vectors*.

Definition 4.8. A *vector solution* to the Basic SDP is one in which all $I_v[\ell]$ are actually vectors in $\mathbb{R}^N$. They are thought of as joint random variables as follows: to make a draw from the collection $I_v[\ell]$, choose $i \in [N]$ uniformly at random and then output the ith coordinate of each vector $I_v[\ell]$.

Another nice aspect of vector solutions is that although they are special cases, we can restrict attention to them if we want:

Proposition 4.9. *Every SDP relaxation always has an optimal vector solution with $N = qn$, and this can be found efficiently.*

Proof. (Sketch.) By Remark 4.6 we may drop the first moment condition (14) in the Basic SDP formulation. Similar to the GW SDP, we now have a linear program over variables

$$\rho_{(v,\ell);(v',\ell')} = \mathbf{E}\Big[I_v[\ell] \cdot I_{v'}[\ell']\Big]$$

with the extra constraint

$$P = (\rho_{(v,\ell);(v',\ell')}) \text{ is the second-moment matrix of a collection of } N \text{ random variables.} \tag{17}$$

But it is straightforward to check (exercise) that this condition is also equivalent to the condition that P is positive semidefinite. Hence as with the GW SDP we can solve the Basic SDP formulation efficiently, and we can write the optimal solution P as $XX^\top$ for some matrix X, the rows of which give an optimal vector solution. □

Working with the Basic SDP relaxation can be somewhat complicated, but in some cases one can simplify it. This occurs especially for CSPs where the domain D is $\{0,1\}$ and/or the maximum constraint arity is 2. In particular, it is not too hard to show (exercise) that for the Max-Cut CSP, the Basic SDP relaxation is equivalent to the much simpler Goemans–Williamson SDP relaxation.

4.4 Rounding SDP solutions and connections to algebra

As with LP-based algorithms, the creativity involved in designing an SDP-based approximation algorithm comes in "rounding" the pseudoindicator random variables to actual assignments with comparable value. One step that seems to always be helpful is to get Gaussian random variables involved.

Definition 4.10. An SDP solution $\mathscr{S}$ is said to be *Gaussian* if the pseudoindicator random variables are jointly Gaussian.

An optimal Gaussian SDP solution always exists and can be obtained (i.e., sampled from) efficiently (exercise).

In fact, the work of Raghavendra (2008) roughly speaking shows that the optimal way to round pseudoindicators to actual assignments always follows this recipe:

1. Replace the pseudoindicator random variables with Gaussian ones, $G_v[\ell]$.
2. Make a large number Q of joint draws from this distribution.
3. Assign variable $v \in V$ a label by applying some "rounding function" f to the Q outcomes of the random variables $(G_v[\ell])_{\ell \in D}$.

Although it is not currently well-understood, it seems that there are connections between good rounding functions f and polymorphisms for the CSP. For example, the Goemans–Williamson algorithm is equivalent to using the above recipe with the "Majority" polymorphism for Max-Cut.

5 Exercises

1. Show that if there is an efficient $(1,1)$-distinguishing algorithm for CSP(Γ) then there is an efficient $(1,1)$-approximation algorithm.
2. Show that if there is an efficient (α,β)-approximation algorithm for CSP(Γ) then there is an efficient (α,β)-distinguishing algorithm. What about the converse?
3. Show that the following problems (i.e., $(1,1)$-distinguishing problems) are in P: Unique-Games, Horn-kSat, k-Lin(q) when q is prime, 2-Sat, any lattice-retraction problem.
4. Give an $O(1)$-time algorithm which $(7/8,\beta)$-distinguishes E3-Sat for any β.
5. Verify Proposition 2.3.
6. Suppose $(1-a,1-b)$-distinguishing CSP(Γ) is NP-hard. Show that $(1-\delta a,1-\delta b)$-distinguishing is also NP-hard for all $0 < \delta < 1$. (Perhaps you'll need to assume a, b, and δ are rational numbers.) Can you prove a similar result that involves making a and b larger?
7. Show that in the Basic IP formulation, conditions (1) and (2) are implied by condition (5). Show the analogous claim for the Basic LP formulation.
8. Show that in the Basic SDP relaxation, the maximization is over a nonempty set of possibilities, and that the maximum is achieved (i.e., we need not write sup).
9. Justify Remark 4.7.
10. Justify Remark 4.6.

11. Justify the inequality used in our analysis of the LP algorithm for Max-Sat: $1-(1-p_C/|S|)^{|S|} \geq (1-(1-1/|S|)^{|S|})p_C$ for all $0 \leq p_C \leq 1$, $|S| \in \mathbb{N}$.

12. Prove Theorem 3.3. (Hint: you will need to show $1-(\frac{3}{4}-\frac{1}{2}p/r)^r \geq \frac{3}{4}p$ for all $p \in [0,1]$, $r \in \mathbb{N}$.) Can you think of any similar algorithms that give the same approximation guarantee?

13. Complete the proof of Proposition 3.8.

14. For the Max-Cut problem on graph $\mathcal{G}$, show that $\mathrm{Opt}(\mathcal{G}) \geq 1/2$. Further, show this is tight in the sense that for all $\epsilon > 0$ there is a graph $\mathcal{G}$ with $\mathrm{Opt}(\mathcal{G}) \leq 1/2+\epsilon$.

15. Justify carefully our reduction to a two-dimensional problem in the analysis of the Goemans–Williamson algorithm.

16. Verify (16), i.e., that the Basic SDP relaxation is indeed a relaxation, and that it is at least as tight as the Basic LP relaxation. (Hints: For the first inequality, consider pseudoindicator random variables that are *actual* indicator random variables. For the second inequality, show that Basic SDP is equivalent to Basic LP if condition (15) is dropped.)

17. Justify Remark 4.6. (Hint: compute the mean and variance of the random variable $\sum_{\ell \in D} I_v[\ell]$.)

18. Justify Remark 4.7. (Hint: first, use the fact that it suffices to consider vector solutions to the SDP. Next, show that using rotations of space, we can make $\sum_{\ell \in D} I_v[\ell]$ equal to the all-1's vector. Finally, deduce the first moment condition from the second moment condition.)

19. Show that the condition 17 is equivalent to the condition that P is a positive semidefinite matrix.

20. Show that the Basic SDP relaxation for Max-Cut is equivalent to the Goemans–Williamson SDP relaxation.

21. Show that the Basic SDP relaxation always has an optimal Gaussian solution.