CS 252, Lecture 8: Constraint Satisfaction Problems

1 Introduction

A key endeavor in theoretical computer science is the following:

*When and why do polynomial time algorithms exist? Can we explain the **P** vs **NP** completeness divide?*

In general, characterizing problems for which there is a polynomial time algorithms turns out to be hard. And thus, we focus on structured class of problems to shed light on the above. To be specific, we are interested in the following question of dichotomy: is it true that every problem in our class is either in **P** or is **NP** hard?

Turns out that such a dichotomy does not exist in general. Ladner's theorem states that if $\mathbf{P} \neq \mathbf{NP}$, then there is a language in **NP** that is neither in **P** nor is **NP** complete. We need the assumption of $\mathbf{P} \neq \mathbf{NP}$ since if $\mathbf{P} = \mathbf{NP}$, then every problem in **P** is also **NP** complete as well. However, the proof of Ladner is non-constructive and we are interested if this phenomenon occurs in "natural" problems. One such candidate is the Constraint Satisfaction Problems (CSP).

2 CSP

Informally, in an instance of a Constraint Satisfaction Problem, there are a collection of variables and a set of constraints on some subset of variables. Our goal is to determine if there is a way to assign values to the variables to satisfy all the constraints. We start with some examples:

1. 3-SAT: In the 3-SAT problem, we are given a set of variables $x_1, x_2, \ldots, x_n$, and we have a set of constraints involving at most three variables. Each constraint is a disjunction of literals[1]. For example, the set of constraints $(x_1 \vee x_2 \vee \overline{x_4}), (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}), (x_2 \vee x_3)$ form an instance of 3-SAT. The objective is to figure out if there is a assignment from $\{0, 1\}$ to the variables $x_1, x_2, \ldots, x_n$ such that all the constraints can be satisfied. For example, in the above instance, assigning $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$ satisfies all the three constraints. (Here, we stick with the convention that 1 is True and 0 is False). 3-SAT is the canonical **NP** complete problem.

2. 1-in-3-SAT: Here, the constraints involve three literals and is satisfied if and only if exactly one of the three literals is true. For example, the problem instance could be $(\overline{x_1}, x_2, \overline{x_4}), (x_1, x_2, x_3), (x_2, x_3, x_4)$. There is no satisfying assignment to this instance. If $x_2 = 1$, the first constraint forces $x_1 = x_4 = 1$,

[1] A literal is either a variable x_i or its negation $\overline{x_i}$.

but then the second constraint has more than two true literals. If $x_2 = 0$, $x_4 = \overline{x_1}$, $x_3 = \overline{x_1}$ and $x_3 = \overline{x_4}$ all three of which can't hold simultaneously.

3. 2-SAT: It is similar to 3-SAT, except that here there are at most two literals in each constraint.
4. k-SAT: k-SAT is a generalization of 3-SAT in which there are at most k literals in each clause.
5. Horn-SAT: In the Horn-SAT problem, there is at most one positive literal (any any number of negated literals) in each clause.
6. Dual Horn-SAT: In the Dual Horn-SAT problem, there is at most negated literal (and any number of positive literals) in each clause.
7. NAE-3-SAT: Here, each constraint involves three literals, and is satisfied if and only if not all the literals have equal value i.e. there is at least one literal which is assigned true, and at least one literal which is assigned false. This problem makes sense even if we assume that no variable is negated in the constraints.
8. ODD-3-SAT: Here, each constraint involves three literals, and is satisfied if and only if there are odd number of literals that are assigned true.
9. So far we have considered only Boolean CSPs. We can also look at CSPs over non-Boolean domains. For example, if the domain is $\{0, 1, 2\}$ and the allowed constraints is (x, y) which is $x \neq y$, we get the classical graph three coloring problem.

Now, we formally define the Constraint Satisfaction Problems. For ease of presentation, we stick with the Boolean case.

Definition 1. (Constraint Template, Constraints) A constraint template $\Gamma = \{P_1, P_2, \ldots, P_t\}$ is a set of predicates P_is where each predicate P_i is a Boolean function $P_i : \{0,1\}^{a_i} \to \{0,1\}$ with arity a_i. A constraint over a constraint template Γ is an expression of the form $P(v_1, v_2, \ldots, v_k)$ where $V = \{v_1, v_2, \ldots, v_k\}$ is the set of the variables, and $P \in \Gamma$ is a predicate of arity k. The constraint is said to be satisfied by an assignment $\sigma : V \to \{0,1\}$ if and only if $P(\sigma(v_1,), \sigma(v_2), \ldots, \sigma(v_k)) = 1$.

We assume that the set Γ and the arity of each predicate in Γ are finite.

Definition 2. (CSP(Γ)) An instance of CSP(Γ) is a set of variables V and a collection of constraints over Γ involving variables from V. Given such an instance, the objective is to determine if there is an assignment to V that satisfies all the constraints.

Note that the definition does not involve literals, and only variables. We can adjust the constraint template to account for this. For example, the 3-SAT is equal to CSP(Γ) with $\Gamma = \{P_0, P_1, P_2, P_3\}$ where $P_0(x, y, z) = x \vee y \vee z, P_1(x, y, z) = \overline{x} \vee y \vee z, P_2(x, y, z) = \overline{x} \vee \overline{y} \vee z, P_3(x, y, z) = \overline{x} \vee \overline{y} \vee \overline{z}$.

3 Dichotomy Theorem

If an instance of a CSP(Γ) is satisfiable, there is a short certificate to verify it: just the assignment of the variables that satisfies all the constraints. This proves that CSP(Γ) is in **NP** for every constraint template Γ. A natural question is the following: for which templates Γ, the problem CSP(Γ) can be solved in polynomial time? For which problem is it **NP** complete? Are there constraint templates for which it is neither?

This question is answered by the CSP Dichotomy Theorem.

Theorem 3. *(CSP Dichotomy Theorem) For every constraint template Γ over a finite domain, CSP(Γ) is either in **P** or is **NP** complete.*

The Boolean case of the above is proved by Schaefer in 1978, and the general case is proved independently by Bulatov and Zhuk in 2017.

We take a look back at the previous examples of CSPs:

1. 3-SAT: It is the canonical **NP** complete problem.
2. 1-in-3-SAT: It is **NP** complete as well.
3. 2-SAT can be solved in polynomial time by solving an auxiliary graph connectivity problem.
4. k-SAT: Since it generalized 3-SAT, it is **NP** complete too.
5. Horn-SAT and Dual Horn-SAT can be solved in polynomial time. It is a good exercise to come up with efficient algorithms to solve these problems.
6. NAE-3-SAT is **NP** complete. It is equivalent to asking whether a 3-uniform hypergraph[2] can be colored with 2 colors.
7. ODD-3-SAT can be solved in polynomial time using Gaussian elimination.
8. Graph 3-coloring is **NP** complete.

In light of the CSP dichotomy theorem, we are interesting in knowing which CSPs are solvable in polynomial time and which ones are **NP** complete. As we mentioned earlier, this question is answered by the Schaefer's theorem for the Boolean case.

Theorem 4. *(Schaefer's theorem) For all Boolean templates Γ, CSP(Γ) is either in **P** or is **NP** complete. Furthermore, CSP(Γ) is in **P** if and only if at least one of the following holds:*

1. *All the predicates in Γ that are not constantly false are satisfied by assigning 1 to each of the variables.*
2. *All the predicates in Γ that are not constantly true are satisfied by assigning 0 to each of the variables.*

[2]Hypergraphs are a generalized graphs, where each edge can connect more than two vertices.

3. *Each predicate in* Γ *is equivalent to a conjunction of* 2*-SAT clauses.*
4. *Each predicate in* Γ *is equivalent to a conjunction of Horn-SAT clauses.*
5. *Each predicate in* Γ *is equivalent to a conjunction of Dual Horn-SAT clauses.*
6. *Each predicate in* Γ *is equivalent to a conjunction of linear equations.*

4 Polymorphisms

The modern view of CSPs is taken through the lens of closure properties of the CSPs called as *polymorphisms*. Roughly speaking, polymorphisms are ways in we can combine satisfying assignments to a given CSP to obtain other satisfying assignments.

We start with 2-SAT as an example. Suppose that we have a 2-SAT instance I on n variables $x_1, x_2, \ldots, x_n$. Let $u_1 \in \{0,1\}^n$ be an assignment to the variables that satisfies all the constraints. Similarly, let u_2, u_3 be (not necessarily different from u_1) satisfying assignments to the given instance. Now, we define $v \in \{0,1\}^n$ as follows: for every $i \in \{1, 2, \ldots, n\}$, $v_i = \mathsf{MAJ}_3((u_1)_i, (u_2)_i, (u_3)_i)$, where MAJ_3 is a 3-ary Boolean function that outputs 1 if and only if at least two of the inputs bits are equal to 1.

We claim that v is also a satisfying assignment to the 2-SAT instance I. In order to prove this, we need to prove that every clause in I is satisfied by the assignment v. Consider an arbitrary clause in I: Suppose it is $x_i \vee \overline{x_j}$. Note that (v_i, v_j) is actually equal to one of $((u_1)_i, (u_1)_j), ((u_2)_i, (u_2)_j), ((u_3)_i, (u_3)_j)$, as we are using the MAJ_3 function. Thus, v also satisfies the constraint $x_i \vee \overline{x_j}$ as each of u_1, u_2, u_3 satisfy it. As this argument holds for each clause in I, v is a satisfying assignment to the instance I. Thus, given three satisfying solutions to an instance of 2-SAT problem, we can take majority function on each bit to get a new satisfying solution to the same instance. This can be viewed as a discrete form of convexity[3]. There are efficient algorithms to optimize over convex sets in optimization, and similarly, in CSPs, if there are non-trivial polymorphisms, there are efficient algorithms to solve them.

We now formally define polymorphisms.

Definition 5. (Polymorphisms) Let Γ is a Boolean constraint template. A function $f : \{0,1\}^m \to \{0,1\}$ is said to be a polymorphism of Γ of arity m if the following holds: For every instance I of variables $V = \{v_1, v_2, \ldots, v_n\}$ over Γ, and for every set of m satisfying assignments to I, $u_1, u_2, \ldots, u_m \in \{0,1\}^n$, the assignment

$$(f((u_1)_1, (u_2)_1, \ldots, (u_m)_1), f((u_1)_2, (u_2)_2, \ldots, (u_m)_2), \ldots, f((u_1)_n, (u_2)_n, \ldots, (u_m)_n))$$

also satisfies all the constraints in I.

There are some trivial polymorphisms for every constraint template Γ : simply output one of the original satisfying assignments i.e. $f(x_1, x_2, \ldots, x_m) = x_i$ for some $i \in \{1, 2, \ldots, m\}$. Such functions are known as dictators. For the 3-SAT problem, these are the only polymorphisms. Recall that 3-SAT is **NP** complete while 2-SAT is in **P**. In fact, this is not an accident, and this has been formalized in the modern CSP dichotomy theorem.

[3]A set S is convex if for any two points $u, v \in S$, the line joining u and v is also in S.

Theorem 6. *(Modern CSP Dichotomy Theorem, Informal) For any constraint template Γ, CSP(Γ) is in **P** if it has "interesting" polymorphisms. Else, it is **NP** complete.*

We now take a look at the polymorphisms of CSPs in Schaefer's theorem.

1. For the first case, the function that always outputs 1 is a polymorphism.
2. For the second case, the function that always outputs 0 is a polymorphism.
3. As we discussed earlier, MAJ of odd arities is a polymorphism of 2-SAT.
4. The binary $\mathsf{AND}(\wedge)$ is a polymorphism of Horn-SAT.
5. The binary $\mathsf{OR}(\vee)$ is a polymorphism of Dual Horn-SAT.
6. Parity, the sum of elements modulo 2, of odd arity is a polymorphism of linear equations.