

In this lecture we continue the discussion of treewidth and study approximation algorithms using LP relaxation and rounding.

## Treewidth

### 24.1 Review

The treewidth of a graph measures roughly “how close” a graph is to a tree. The definition of treewidth follows from the definition of a tree decomposition.

**Definition 1** A *tree decomposition* of a graph  $G(V, E)$  is a pair  $(T, \chi)$  where  $T = (I, F)$  is a tree and  $\chi = \{\chi_i | i \in I\}$  is a family of subsets of  $V$  such that (a)  $\bigcup_{i \in I} \chi_i = V$ , (b) for each edge  $e = \{u, v\} \in E$ , there exists  $i \in I$  such that both  $u$  and  $v$  belong to  $\chi_i$ , and (c) for all  $v \in V$ , there is a set of nodes  $\{i \in I | v \in \chi_i\}$  that forms a connected subtree of  $T$ . The *width* of a tree decomposition  $(T, \chi)$  is given by  $\max_{i \in I} (|\chi_i| - 1)$ .

The *treewidth* of a graph  $G$  equals the minimum width over all possible tree decompositions of  $G$ .

Equivalently, treewidth can be defined in terms of elimination orderings. An *elimination ordering* of a graph is simply an ordering of its vertices. Given an elimination ordering of a graph  $G$ , one can construct a new graph by iteratively removing the vertices in order, every time adding an edge between each pair of neighbors of the vertex just removed. The induced treewidth of this elimination ordering is the maximum neighborhood size in this elimination process. The treewidth of  $G$  is then the minimum induced treewidth over all possible elimination orderings.

For example, the treewidth of a tree is 1, and the treewidth of a cycle is 2 (each time you remove a vertex, you connect its two neighbors to form a smaller cycle). Another class of graphs with treewidth 2 are series parallel graphs.

**Definition 2** A series parallel graph with head  $h$  and tail  $t$  is one of the following:

- a single vertex
- two series parallel graphs with heads  $h_1, h_2$  and tails  $t_1, t_2$  such that  $h = h_1, t_1 = h_2, t_2 = t$  (series connection)
- two series parallel graphs with heads  $h_1, h_2$  and tails  $t_1, t_2$  such that  $h = h_1 = h_2, t = t_1 = t_2$  (parallel connection)

Note the treewidth is not bounded by maximum degree. For example, a grid of  $n$  nodes with maximum degree 4 has treewidth of roughly  $\sqrt{n}$ .

In general, the problem of computing the treewidth of a graph is NP-complete. However, the problem is fixed-parameter tractable with respect to treewidth itself.

## 24.2 Fixed-parameter tractability of SAT

Consider a Boolean formula in conjunctive normal form (CNF), namely a formula expressed as the AND of a set of clauses, each of which is the OR of a set of literals, each of which is either a variable or the negation of a variable.

The satisfiability problem (SAT) is to determine whether or not a variable assignment exists that makes the formula true, i.e. at least one literal in each clause is true. This is one of the canonical NP-complete problems.

In this section, we will show that SAT is fixed-parameter tractable with respect to treewidth. Clearly, we need to define the graph that contains our treewidth parameter.

**Definition 3** Given a CNF formula  $F$  (i.e. a conjunction of clauses), the *primal graph* of  $F$ ,  $G(F)$ , is the graph whose vertices are the variables of  $F$ , where two variables are joined by an edge if they appear together in some clause.

Suppose that the primal graph  $G(F)$  has treewidth  $k$ . The proof below provides an algorithm that, given an optimal elimination ordering  $x_1, x_2, \dots, x_n$  of the vertices of  $G(F)$ , solves SAT in time  $O(nk \cdot 2^k)$ . Though we have restricted our attention to SAT, a similar approach can be used for any optimization problem that has an associated graph with local dependency structure.

**Claim 1** *SAT is fixed-parameter tractable with respect to treewidth.*

**Proof:** Let  $F$  be a CNF formula and  $G$  its primal graph. As in the tree decomposition procedure, eliminate the variables  $x_1, x_2, \dots, x_n$  in that order to construct a new graph  $G^*$ . When we eliminate  $x_i$ , we remove all edges incident to  $x_i$  in  $G$ , and add any non-existing edges between a pair of neighbors of  $x_i$  in  $G^*$ . Note that we can form a new CNF formula in which we combine all clauses containing  $x_i$  into one by taking the disjunction of all the literals in those clauses, except those containing the variable  $x_i$ . It is clear that  $G^*$  is the primal graph for the new formula.

For each new clause in this procedure, construct a truth table for that clause. The truth table is a list of all satisfying assignments of variables in the clause which is a partial satisfying assignment for all the original clauses. For example, if  $(x \vee y \vee z)$  and  $(\neg x \vee \neg y \vee z)$  are the only two clauses containing  $x$ , then on eliminating  $x$ , we form the new clause  $(y \vee z \vee \neg y)$ . The truth table for the new clause is derived from truth tables for the original clauses as follows:

$$\begin{array}{cccccc}
 \overline{( x \vee y \vee z )} & \overline{( \neg x \vee \neg y \vee z )} & & & \overline{( y \vee z \vee \neg y )} \\
 1 & 1 & 1 & 1 & 1 & 1 \\
 0 & 1 & 1 & 0 & 1 & 1 \\
 1 & 0 & 1 & 1 & 0 & 1 \\
 0 & 0 & 1 & 0 & 0 & 1 \\
 1 & 0 & 0 & 1 & 0 & 0 \\
 0 & 1 & 0 & 0 & 1 & 0 \\
 1 & 1 & 0 & 0 & 0 & 0
 \end{array}
 \longrightarrow
 \begin{array}{ccc}
 1 & 1 & - \\
 0 & 1 & - \\
 1 & 0 & - \\
 0 & 0 & -
 \end{array}$$

Repeating this procedure for each variable eliminated, we will eventually end up with a single truth table representing the entire formula. If the table has at least one entry, then there is an assignment which satisfies all the original clauses. In this case, we return *true*. Otherwise,  $F$  is not satisfiable, so we return *false*.

Since  $G$  has treewidth  $k$ , there are at most  $k$  neighbors of any vertex in an optimal elimination ordering, so we construct a truth table with at most  $k \cdot 2^k$  entries per vertex eliminated. Since there are  $n$  vertices, this yields a total running time of  $O(nk \cdot 2^k)$ . ■

## Randomized Approximation Algorithms

In this part we first introduce the method of LP relaxation and rounding for approximation to ILP. Then we consider two randomized algorithms for the MAX-SAT problem.

### 24.3 Vertex-Cover

Recall the *vertex cover* problem, in which we are given a graph and wish to pick a subset of vertices so that at least one endpoint of each edge is in the cover and the size of the cover is minimized. Previously, we showed that a greedy algorithm that iteratively picks both endpoints of any edge that is not yet covered is a 2-approximation. We show here an alternate construction that achieves the same bound.

Consider the following *integer* linear program (ILP). Let  $x_i = 1$  iff  $i$  is in cover.

$$\begin{array}{l}
 \min \sum x_i, \text{ such that} \\
 x_i + x_j \geq 1 \text{ for every edge } \{i, j\} \\
 0 \leq x_i \leq 1
 \end{array}$$

Note that this ILP captures the VC problem exactly, which implies that ILP is NP-hard.

Let us instead solve the LP relaxation obtained from this ILP by dropping the integrality constraint. The problem is that in general the solution found is not integral, so we need a way to *round* it to a feasible one. Take  $i$  into the cover, i.e. set  $\tilde{x}_i = 1$ , iff  $x_i \geq 1/2$ .

**Claim 2** *The rounded solution is a feasible vertex cover.*

**Proof:** For each edge  $\{i, j\}$  we have  $x_i + x_j \geq 1$ . Then we cannot have  $x_i, x_j < 1/2$ , so at least one of them will be included in the rounded solution. ■

**Claim 3** *The rounded solution is a 2-approximation to the fractional objective of the LP relaxation, and thus a 2-approximation to the ILP.*

**Proof:** The rounded  $\tilde{x}_i \leq 2x_i$  so  $\sum \tilde{x}_i \leq 2 \sum x_i = 2OPT$ . ■

It should be noted that, in practice, LP relaxations often perform much better than their bounds. This is true especially if we add more constraints (that are redundant in the ILP but actually linearly independent in the LP). One such method is branch & bound which prunes the search space when the upper bound on OPT (obtained via LP relaxation) is not promising.

## 24.4 Random assignment to MAX-SAT

Recall the satisfiability problem (SAT) from the first part of the lecture was a decision problem. The *maximum satisfiability* problem (MAX-SAT) is to find a variable assignment that maximizes the number of true clauses.

One common variant of satisfiability is  $k$ SAT, in which each clause is restricted to be the OR of  $k$  literals, for fixed  $k$ . There is a straightforward polynomial time algorithm for 2SAT, but 3SAT is NP-complete.

We may similarly define MAX- $k$ SAT to be MAX-SAT restricted to those formulas containing only clauses of size  $k$ . It turns out, however, that even MAX-2SAT is NP complete.

Let us find the expected approximation ratio of the algorithm for MAX- $k$ SAT consisting of making each variable true or false at random.

We have

$$E[\# \text{ clauses satisfied}] = \sum_{\text{clause } c} \Pr(c \text{ is satisfied}) = (\# \text{ clauses}) \cdot (1 - (1/2)^k)$$

implying that we have a  $\frac{1}{1-2^{-k}}$ -approximation algorithm. For example, for  $k = 3$ , we have  $8/7$ , and in the worst case of  $k = 1$  we still expect to satisfy at least half of all clauses. Therefore we got a 2-approximation for general MAX-SAT.

We also note that it has been shown that the  $8/7$  approximation ration for MAX-3SAT is optimal unless  $P=NP$ , i.e. it is NP-hard to satisfy  $7/8 + \epsilon$  clauses for any  $\epsilon > 0$ , even in the expected sense.

## 24.5 LP relaxation of MAX-SAT

Consider the following integer linear program (ILP). Let binary variable  $z_j$  be 1 iff clause  $c_j$  is satisfied. Let binary variable  $y_i$  be 1 iff variable  $i$  is true. Let  $c_j^+$  denote the set of variables appearing non-negated in clause  $c_j$ , and let  $c_j^-$  denote those variables that are negated. The problem is as follows:

$$\begin{aligned} & \max \sum_j z_j \text{ such that} \\ & \forall j, \sum_{i \in c_j^+} y_i + \sum_{i \in c_j^-} (1 - y_i) \geq z_j \\ & 0 \leq z_j \leq 1, 0 \leq y_i \leq 1 \end{aligned}$$

It is fairly clear that this expresses the MAX-SAT problem if the variables are restricted to be integral. We consider the LP that results from relaxing this restriction.

We note that the solution we get in the relaxed problem may have fractional values for the variables  $y_i$ . Since these variables are meant to represent whether the variables  $x_i$  in the Boolean formula are true or not, we may have difficulty interpreting the LP solution as a variable assignment.

We introduce the technique of *randomized rounding*, a technique applicable to ILPs in general.

Given possibly fractional solutions  $y_i$ , randomized rounding sets  $x_i = 1$  with probability  $y_i$ , and  $x_i = 0$  with probability  $1 - y_i$ .

We note that a trivial consequence of this is that

$$E[x_i] = y_i.$$

We now work to estimate the probability that a given clause is satisfied. From the constraints on the LP, we know that the expected number of satisfied variables in a clause  $c_j$  is at least  $z_j$ . Define a sequence  $\beta_k$  by

$$\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k = \left(1, \frac{3}{4}, \sim .704, \dots \rightarrow 1 - \frac{1}{e} \approx .632\right).$$

**Claim 4** Given a clause  $c_j$  of  $k$  variables, the probability that it is satisfied is at least  $\beta_k z_j$ .

**Proof:** To simplify the argument, we assume that all literals are positive (non-negated). We note that the probability that  $c_j$  is satisfied is the complement of the probability that none of its variables is true, which is

$$\prod_{i \in c_j} (1 - y_i).$$

We note that the  $y_i$ s are constrained by

$$\sum_{i \in c_j} y_i \geq z_j.$$

Thus appealing to inequality tricks (Jensen's inequality and the fact that the logarithm function is convex) we conclude that this probability is minimized when

$$y_i = z_j/k.$$

Thus we can lower bound the probability of  $c_j$  being satisfied by

$$1 - \left(1 - \frac{z_j}{k}\right)^k \geq \beta_k z_j,$$

as desired. ■

We now observe that this claim implies that this algorithm is a  $\frac{1}{1-1/e} = \frac{e}{e-1}$ -approximation: since the sequence  $\beta_k$  is always at least  $1 - \frac{1}{e}$ , the expected number of satisfied clauses is at least  $(1 - \frac{1}{e}) \sum z_j$ , which is thus within  $(1 - \frac{1}{e})$  factor of optimal since the solution to the relaxed problem  $\sum z_j$  is at least as good as the solution to the ILP.

## 24.6 Combined algorithm

We introduce a third algorithm to approximate MAX-SAT that consists of running the previous two algorithms and picking the solution that maximizes the number of satisfied clauses. While randomized rounding provides a solution at least half as good as OPT, and LP relaxation provides a solution within  $1 - \frac{1}{e}$  factor, the combined algorithm will provide a solution within 3/4 of optimal.

We note that the above algorithm performs at least as well as the algorithm where we pick randomly between the two choices suggested by randomized rounding and LP relaxation. Letting  $k_j$  denote the number of variables in clause  $j$ , we have that the expected number of variables satisfied in the random solution is at least

$$\frac{1}{2} \left[ \sum_j (1 - 2^{-k_j}) z_j + \sum_j (\beta_{k_j}) z_j \right] = \sum_j \frac{1}{2} (1 + \beta_{k_j} - 2^{-k_j}) \geq \frac{3}{4} \sum_j z_j.$$

Thus this is a 4/3-approximation, as desired.

## 24.7 Set Cover

As before, in the *set cover* problem we are given sets  $S_i$  containing elements in  $\{1 \dots n\}$  and asked to find the minimum number of sets such that their union "covers" all the elements. We have already seen a greedy algorithm that produces an  $O(\log n)$ -approximation. We present an LP relaxation algorithm that also provides an  $O(\log n)$ -approximation.

Consider the following ILP. Let  $y_i = 1$  iff  $S_i$  is in the cover. Find

$$\begin{aligned} \min \sum y_i, \text{ such that} \\ \sum_{S_i \ni j} y_i \geq 1 \\ 0 \leq y_i \leq 1. \end{aligned}$$

Observe that when  $y_i$  is restricted to be integral this captures the set cover problem. Now consider the result of optimizing the LP relaxation. If we try the above randomized rounding technique of putting set  $S_i$  in our cover with probability  $y_i$ , then we end up with the expected number of sets covering any point  $j$  to be 1. However, the set cover problem requires that *all* sets be covered, which is a considerably stronger statement. Thus regular randomized rounding might fail to give a solution to set cover.

We remedy this problem by modifying the randomized rounding scheme as follows: for some fixed  $\alpha \geq 1$ , choose set  $S_i$  with probability  $\alpha y_i$ . Note that the expected number of sets chosen is now  $\alpha \sum y_i \leq \alpha |\text{OPT}|$ , so that we will have an  $\alpha$ -approximation if the sets actually do cover  $1 \dots n$ .

In order to show that the chosen sets do provide a cover with high probability, we introduce the *Chernoff bound*, another fundamental technique in the probabilistic method.

**Theorem 1 (Chernoff Bound)** *Let  $X = X_1 + \dots + X_n$  be a sum of indicator (0-1) variables that are independent. Denote  $E[X]$  by  $\mu$ . Then*

$$\Pr(X \leq (1 - \epsilon)\mu) \leq e^{-\epsilon^2 \mu / 2}.$$

And if  $\epsilon \leq 2e - 1$ , then

$$\Pr(X \geq (1 + \epsilon)\mu) \leq e^{-\epsilon^2 \mu / 4},$$

while if  $\epsilon > 2e - 1$ , then

$$\Pr(X \geq (1 + \epsilon)\mu) \leq 2^{-(1+\epsilon)\mu}.$$

To apply this theorem, we note that the number of sets covering an element  $j$  is the sum of independently random indicator variables, with mean  $\alpha$ . We need to bound the probability that this random variable is 0. We apply the Chernoff bound with  $\epsilon = 1$ , which states that the probability that no set covers  $j$  is at most

$$e^{-\epsilon^2 \mu / 2} = e^{-\alpha / 2}.$$

Applying the union bound, we have that the probability that *any* element is uncovered is at most

$$ne^{-\alpha / 2}.$$

Setting  $\alpha = 3 \log_2 n$ , this probability becomes at most  $1/\sqrt{n} \ll 1$ . Thus we have a Monte Carlo  $O(\log n)$  approximation, as desired.