Résolution exacte de problèmes NP-difficiles
Lecture 1: Branching Algorithms

January 11, 2021 Lecturer: Eunjung Kim

1 Parameterized algorithm

Parameterized problems: Consider a parameterized problem P. We understand P as a collection of all yes-instances of the problem. For example, if P denotes the problem VERTEX COVER , then we shall also see VERTEX COVER as the set of instances

$$\{(G,k) : G \text{ is a graph having a vertex cover of size at most } k\}.$$

So, whenever an instance (x,k) of a parameterized problem P is a yes-instance, we equivalently write as $(x,k) \in P$.

Parameterized algorithm: A parameterized problem P is said to be *fixed-parameter tractable* if there is an algorithm which decides the membership of any input instance (x,k) to P in time $f(k) \cdot |x|^c$ for some constant c and a function f, Such an algorithm is called a *parameterized algorithm* or an *fpt-algorithm*.

2 $\mathcal{O}^*(2^k)$-time algorithm for VERTEX COVER

The procedure **VC** in Algorithm 1 takes as an input instance a graph G and a non-negative integer k, and outputs YES or NO correctly.

Algorithm 1 Algorithm for VERTEX COVER

```
procedure VC(G, k)
    if G has no edge then return YES.
    else if k = 0 then return NO.
    else                                        ▷ G has an edge and k > 0
        Pick an edge uv.
        return VC(G − u, k − 1) or VC(G − v, k − 1).
    end if
end procedure
```

Below, we prove the correctness and the running time of the algorithm **VC**[1].

[1] A detailed proof is given for this algorithm to illustrate what a correctness proof in full should be like. For other algorithms that we'll encounter later, not so much details might be delineated.

Lemma 1. *Given an input instance (G, k) of* VERTEX COVER *, the procedure* **VC** *correctly decides whether an input instance (G, k) is* YES *or* NO *in time $\mathcal{O}(2^k \cdot poly(n))$.*

Proof: First, let us show the correctness of **VC**; that is, **VC**(G,k) returns YES if and only if (G, k) is a YES-instance. We prove this by induction on k. Notice that when $k = 0$, the procedure **VC** always returns some output (and do not branch). Therefore, in any recursive call during the execution of the procedure, the parameter k remains non-negative. If $k = 0$ or $|E| = \emptyset$, then it is trivial to verify that Lines 5 and 3 respectively returns a correct output. Therefore, we consider the case when $|E| > 0$ and $k > 0$.

If (G, k) is a YES-instance, that is G has a vertex cover X of size at most k, then at least one of u and v in Line 5 is included in X. Observe that at least one of the instances $(G - u, k - 1)$ and $(G - v, k - 1)$ is a YES-instance because $X \setminus u$ (respectively $X \setminus v$) is a vertex cover of $G - u$ (respectively $G - v$). By induction hypothesis, **VC** returns a correct output upon each of $(G - u, k - 1)$ and $(G - v, k - 1)$. This means that, by the fact that one of them is a YES-instance, the returned output [2] at Line 6 will be YES and thus the output of **VC**(G, k) is correct.

Now suppose that (G, k) is a NO-instance. Then the instance $(G - u, k - 1)$ (likewise $(G - v, k - 1)$) is a NO-instance. Indeed, if $G - u$ has a vertex cover X' of size at most $k - 1$ then $X' \cup \{u\}$ is also a vertex cover of G and we have $|X' \cup \{u\}| \leq k$, a contradiction. Therefore, by induction hypothesis **VC** outputs NO to both instances called in Line 6. Therefore, Line 6 returns NO. This proves the correctness of the algorithm.

To see the running time, observe that the search tree depicting the relations of the recursive calls has depth at most k. This is because each instance created during the recursive calls have $k \geq 0$ and every branching decreases the parameter value by 1. Therefore, there are at most 2^k leaves and $\mathcal{O}(2^k)$ nodes in the search tree. As the algorithms spends $poly(n)$ steps at each node, the running time follows. $\square$

3 $\mathcal{O}^*((3k)^k)$-time algorithm for FEEDBACK VERTEX SET

Lemma 2. *For any feedback vertex set X of $G = (V, E)$, it holds that*

$$\sum_{v \in X} (d(v) - 1) \geq |E| - |V| + 1.$$

Proof: All the edges of G are either counted as part of the forest $G - X$, or incident with X. Therefore,

$$\sum_{v \in X} deg(v) + |V| - |X| - 1 \geq |E|,$$

which yields the claimed inequality. $\square$

Lemma 3. *Let $G = (V, E)$ be a graph of degree at least three. Let $\{v_1, v_2, \ldots, v_n\}$ be the vertex set of G such that*

$$deg(v_1) \geq deg(v_2) \geq \cdots \geq deg(v_n).$$

Then for any feedback vertex set X, we have $X \cap \{v_i : i \leq 3k\} \neq \emptyset$.

[2] Here, we construe 'or' as the boolean operation OR by equating YES to True and NO to False.

Proof: Suppose not. Observe

$$\sum_{i\leq 3k}(deg(v_i)-1) \geq 3\sum_{v\in X}(deg(v)-1) \geq 3\big(|E|-|V|+1\big),$$

because each vertex of X has degree at most $deg(v_{3k})$. Also, due to $X \subseteq \{v_{3k+1},\ldots,n\}$

$$\sum_{i>3k}(deg(v_i)-1) \geq \sum_{v\in X}(deg(v)-1) \geq |E|-|V|+1.$$

Summing the two equalities, we get

$$\sum_{i=1}^{n}(deg(v_i)-1) = 2|E|-|V| \geq 4\big(|E|-|V|+1\big),$$

and thus

$$3|V| > 2|E| = \sum_{v\in V} deg(v).$$

This contradicts the assumption that each v has degree at least three. $\square$

4 $\mathcal{O}^*(4^{k-\mathtt{lp}^*(G)})$-time algorithm for VERTEX COVER

Here, we consider the problem VERTEX COVER PARAMETERIZED ABOVE LP. An input instance and the question is the same as in VERTEX COVER , but the parameter under consideration is $k-\mathtt{lp}^*(G)$. Here k is the budget for vertex cover in the input, and $\mathtt{lp}^*(G)$ is the value of an optimal fractional solution to the following linear program LPVC for VERTEX COVER .

$$\begin{array}{lll} [\mathsf{LPVC}] & \min \sum_{u\in V(G)} x_u & \\ & x_u + x_v \geq 1 & \forall (u,v)\in E(G) \\ & x_u \geq 0 & \forall u \in V(G) \end{array}$$

Observe that if an optimal solution to the above LP is integral, then it corresponds to an optimal vertex cover. In general, an optimal solution x^* to LPVC is not necessarily integral.

The basis of this parameterization is the following fact:

the size of an optimal vertex cover of $G \geq \mathtt{lp}^*(G)$.

Therefore, the parameter $k-\mathtt{lp}^*(G)$ can be assumed to be non-negative; otherwise, it holds that $k < \mathtt{lp}^*(G)$, which means that any vertex cover of G has size strictly larger than k. Especially G does not have a vertex cover of size at most k. We can correctly declare (G,k) as a NO-instance in this case.

Lemma 4. *If $k < \mathtt{lp}^*(G)$, then the input instance (G,k) is a* NO*-instance.*

Half-integrality of LP for VERTEX COVER

Now we establish the half-integrality of LPVC, that is, the property that there is always an optimal fractional solution x^* to LPVC such that $x^*_v = \frac{1}{2}$ for every $v \in V$.

Let x^* be an optimal fractional solution fo LPVC, which is not necessarily half-integral. We partition[3] $V(G)$ according to their values in x^*.

- $H_0 := \{u \in V(G) : x^*_u > 0,5\}$
- $C_0 := \{u \in V(G) : x^*_u < 0,5\}$
- $R_0 := \{u \in V(G) : x^*_u = 0.5\}$

It is known that LP can be solved in polynomial time, hence the partition (R_0, H_0, C_0) can be found in polynomial time. We shall see that x^* can be converted into a half-integral solution, i.e. $x_u \in \{0, 0.5, 1\}$ while maintaining the optimality, and thus showing that there exists a *half-integral* optimal solution to LP formulation of VERTEX COVER .

Lemma 5. *For every subset $H'_0 \subseteq H_0$, we have $|H'_0| \leq |N(H'_0) \cap C_0|$.*

Proof: Take $\epsilon = \min\{x^*_u - 0.5 : u \in H'_0\}$ and note that $\epsilon > 0$. Consider a solution x' defined as:

$$x'_u = \begin{cases} x^*_u - \epsilon & \text{if } u \in H'_0 \\ x^*_u + \epsilon & \text{if } u \in N(H'_0) \cap C_0 \\ x^*_u & \text{otherwise} \end{cases}$$

It is easy to verify that x' is a feasible LP solution. The objective value of x' equals

$$\sum_{u \in V(G)} x^*_u + \epsilon(-|H'_0| + |N(H'_0) \cap C_0)|).$$

From the optimality of x^*, our claim follows. □

Lemma 6. *The following solution $\tilde{x}$ is optimal to* LPVC*:*

$$\tilde{x}_u = \begin{cases} 0.5 & \textit{if } u \in R_0 \\ 1 & \textit{if } u \in H_0 \\ 0 & \textit{if } u \in C_0 \end{cases}$$

Proof: Let H'_0 be the set of vertices in H_0 with $x^*_u < 1$ and C'_0 be the set of vertices in C_0 with $x^*_u > 0$. Among all optimal solutions to LP yielding the partition (R_0, H_0, C_0), choose x^* so as to minimize the set $H'_0 \cup C'_0$. We shall show that $H'_0 \cup C'_0 = \emptyset$, which establishes $x^* = \tilde{x}$ and thus the statement.

[3] H, C and R represent 'Head', 'Crown' and 'Rest'.

For the sake of contradiction, $H_0' \cup C_0' \neq \emptyset$. Take $\delta = \min\{1 - x_u^* : u \in H_0'\} \cup \{x_u^* : u \in C_0'\}$ and note that $\delta > 0$. Consider a solution x' defined as:

$$x'_u = \begin{cases} x_u^* + \delta & \text{if } u \in H_0' \\ x_u^* - \delta & \text{if } u \in C_0' \\ x_u^* & \text{otherwise.} \end{cases}$$

As we chose minimum δ, it holds $x' \geq 0$. It is straightforward to verify that x' is feasible.

Now we argue that x' is an optimal LP solution. From Lemma 5, we know $|H_0'| \leq |N(H_0') \cap C_0|$. Also observe that $N(H_0') \cap C_0 \subseteq C_0'$ since otherwise, there would be an edge (u, v) with $u \in H_0'$ and $v \in C_0 \setminus C_0'$ such that $x_u^* + x_v^* < 1 + 0 = 1$, contradicting the feasibility of x^*. We derive $|H_0'| \leq |C_0'|$. Then, the objective value of x' equals

$$\sum_{u \in V(G)} x_u^* + \delta(|H_0'| - |C_0'|),$$

which does not exceed $\sum_{u \in V(G)} x_u^*$. Therefore, x' is an optimal solution such that $\{u \in H_0 : x'_u < 1\} \cup \{u \in C_0 : x'_u > 0\} \subsetneq H_0' \cap C_0'$, a contradiction. This complete the proof. □

We remark that the proof of Lemma 6 is constructive: one can use the procedure in the proof to transform an arbitrary optimal solution to a half-integral solution. Fom now on we assume that x^* is a half-integral solution as in Lemma 6.

Preprocessing: how to get all-$\frac{1}{2}$ optimal solution

From the previous discussion, we know that a half-integral optimal solution x^* for VERTEX COVER can be obtained in polynomial time. We define the set $V_a(x^*)$ as the set of all vertices v such that $x_v^* = a$ for $a \in \{0, .5, 1\}$. When x^* is obvious from the context, we drop it. From the half-integrality of x^*, it is obvious that $(V_0, V_1, V_{\frac{1}{2}})$ forms a partition of V.

Lemma 7. *For the graph $G[V_{\frac{1}{2}}]$, all-$\frac{1}{2}$ is an optimal fractional solution.*

Proof: If all-$\frac{1}{2}$ is not an optimal fractional solution to LPVC of $G[V_{\frac{1}{2}}]$, consider an optimal fractional solution z^* of $G[V_{\frac{1}{2}}]$ and observe

$$\sum_{v \in V_{\frac{1}{2}}} z_v^* < \sum_{v \in V_{\frac{1}{2}}} x_v^*.$$

We can obtain a new feasible solution to LPVC of G by replacing the value of x_v^* by z_v^* for all $v \in V_{\frac{1}{2}}$ (and keep other values unchanged). It is not difficult to verify that the new fractional solution is feasible and has strictly smaller objective value than x^*, contradicting the optimality of x^*. □

Lemma 8. *The instance $I' = (G', k')$ is equivalent to $I = (G, k)$, where $G' = G[V_{\frac{1}{2}}]$ and $k' = k - |V_1|$. Furthermore, it holds that $k' - \mathtt{lp}^*(G[V_{\frac{1}{2}}]) = k - \mathtt{lp}^*(G)$.*

Proof: The proof of the first statement is proved in the next lecture note. The second statement is derived from the following.

$$\begin{aligned}
\mathtt{lp}^*(G) &= \sum_{v\in V} x_v^* = \sum_{v\in V_0} x_v^* + \sum_{v\in V_1} x_v^* + \sum_{v\in V_{\frac{1}{2}}} x_v^* \\
&= \frac{1}{2}\cdot |V_{\frac{1}{2}}| + |V_1| \\
&= \mathtt{lp}^*(G[V_{\frac{1}{2}}]) + |V_1|,
\end{aligned}$$

where the last equality holds because of Lemma 7. □

Notice that Lemma 7 guarantees that all-$\frac{1}{2}$ is an optimal fractional solution to $G[V_{\frac{1}{2}}]$, but it does not guaranteed that it's the *unique* optimal solution. So how can we achieve the uniqueness? We do it *greedily*, namely, try each vertex v and fix its LP value to one and see if it leads to yet another optimal fractional solution. To be precise, here is the reduction rule. We write $|y^*|$ to denote $\sum_{v\in V} y_v^*$.

Reduction Rule 1. *If there is a vertex $v \in V$ and a half-integral solution y^* to* LPVC *such that $y_v^* = 1$ and $|y^*| = \mathtt{lp}^*(G)$, then define the new instance $I' = (G', k')$ with $G' = G[V_{\frac{1}{2}}(y^*)]$ and $k' = k - |V_1(y^*)|$.*

The soundness of Reduction Rule 1 is derived immediately from Lemma 8. It is clear that after applying Reduction Rule 1 exhaustively, all-$\frac{1}{2}$ is the unique optimal fractional solution to LPVC of the resulting graph. We just state this observation as the next statement.

Observation 1. *If (G, k) is irreducible with respect to Reduction Rule 1, then $x_v^* = \frac{1}{2}$ for all $v \in V$ is the unique fractional optimal solution to* LPVC.

Branching + Preprocessing with runtime analysis

Algorithm 2 Algorithm for VERTEX COVER ABOVE LP

procedure VC-above-LP(G, k)
  Apply Reduction Rule 1 until it can no longer be applied.
  ▷ Now, (G, k) has all-$\frac{1}{2}$ as the unique fractional optimal sol.
  if $k < \mathtt{lp}^*(G)$ **then return** NO. ▷ Lemma 4
  else if $k \geq 2\cdot\mathtt{lp}^*(G)$ **then return** YES. ▷ Lemma 9
  else ▷ G has an edge and $k - \mathtt{lp}^*(G) \geq 0$
    Pick an edge uv.
    return VC-above-LP$(G-u, k-1)$ or **VC-above-LP**$(G-v, k-1)$.
  end if
end procedure

The algorithm for the problem VERTEX COVER ABOVE LP is presented as **VC-above-LP**. The correctness of Line 4 is due to Lemma 4. To see the correctness of Line 5, we need the next lemma [4].

Lemma 9. *For any graph G, one can find a vertex cover Z of size at most $2 \cdot \texttt{lp}^*(G)$ in polynomial time. In particular, an instance (G, k) to* VERTEX COVER *is a* YES*-instance if $k \geq 2 \cdot \texttt{lp}^*(G)$.*

Proof: First compute an optimal half-integral solution x^* to LPVC of G and consider the usual partition $(V_0, V_1, V_{\frac{1}{2}})$. We take $Z = V_1 \cup V_{\frac{1}{2}}$. Since V_0 is an independent set, Z is a vertex cover. Now

$$|Z| = |V_1| + |V_{\frac{1}{2}}| \leq 2(|V_1| + 0.5 \cdot |V_{\frac{1}{2}}|) = 2 \cdot \texttt{lp}^*(G).$$

The second statement follows immediately. □

The rest of the correctness proof is rather tedious. To analyze the running time of **VC-above-LP**, consider the measure

$$\mu(G, k) = k - \texttt{lp}^*(G).$$

We want to argue that the measure decreases by at least $\frac{1}{2}$ in each branching of Line 8. The key observation is the following inequality (in fact, equality holds)

$$\texttt{lp}^*(G - v) \geq \texttt{lp}^*(G) - \frac{1}{2}.$$

Suppose this is not the case, that is, $\texttt{lp}^*(G - v) \leq \texttt{lp}^*(G) - 1$ and let y^* be an optimal fractional solution of $G - v$, i.e. $|y^*| = \texttt{lp}^*(G - v)$. We can extend y^* to a fractional solution z^* of G by setting the value at v equal to 1. Then

$$|z^*| = |y^*| + 1 = \texttt{lp}^*(G - v) + 1 \leq \texttt{lp}^*(G),$$

which means that z^* is an optimal fractional solution of G which is not all-$\frac{1}{2}$. This contradicts that Reduction Rule 1 has been applied exhaustively in Line 2. Now

$$\mu(G - v, k - 1) = (k - 1) - \texttt{lp}^*(G - v) \leq k - 1 - (\texttt{lp}^*(G) - \frac{1}{2}) = \mu(G, k) - \frac{1}{2}.$$

By symmetry, the branching at the endpoint u also decreases the measure by $\frac{1}{2}$.

During the branching algorithm, the measure μ can drop down to -0.5 at most (in which case, we declare as NO-instance). Therefore, the length of a longest root-to-leaf path in the search tree is $2\mu(G, k) = 2(k - \texttt{lp}^*(G))$. It follows that the number of leaves in the search tree is at most $4^{k - \texttt{lp}^*(G)}$, and the running time is $\mathcal{O}^*(4^{k - \texttt{lp}^*(G)})$.

Because of Line 5 (see Lemma 9 as well), we may assume that $k < 2 \cdot \texttt{lp}^*(G)$ since otherwise, we can immediately declare the given instance as YES, in constant time. Therefore, $k - \texttt{lp}^*(G) \leq \frac{k}{2}$. We summarize the result in the next statement.

[4] From $\texttt{lp}^*(G) \leq$ the size of an optimal vertex cover, we deduce that the size of Z is at most twice the size of an optimal vertex cover. That is, Lemma 9 presents a 2-approximation algorithm for VERTEX COVER

Lemma 10. *The algorithm* **VC-above-LP** *solves, given an instance* (G, k)*, the problem* VERTEX COVER ABOVE LP *in time* $\mathcal{O}^*(4^{k-\mathtt{lp}^*(G)})$ *as well as* $\mathcal{O}^*(2^k)$*-time* [5].

Problems List

VERTEX COVER

Instance: a graph $G = (V, E)$, a positive integer k.

Question: Does G have a vertex cover of size at most k, i.e. a set X of vertices such that $G - X$ is an independent set?

ODD CYCLE TRANSVERSAL

Instance: a graph $G = (V, E)$, a positive integer k.

Question: Does G have a vertex cover of size at most k, i.e. a set X of vertices such that $G - X$ is bipartite?

FEEDBACK VERTEX SET

Instance: a graph $G = (V, E)$, a positive integer k.

Question: Does G have a feedback vertex set (fvs) of size at most k, i.e. a set X of vertices such that $G - X$ is acyclic?

d-HITTING SET

Instance: a universe V, a family $\mathcal{E}$ of subsets of V each of which having size at most d, a positive integer k.

Question: Is there a hitting set of $\mathcal{E}$, i.e. a subset X of V such that $X \cap e \neq \emptyset$ for every $e \in \mathcal{E}$?

[5] Recall that we already have a simple branching algorithm for VERTEX COVER running in time $\mathcal{O}^*(2^k)$. Therefore, the second running time might seem unnecessary. However, almost identical algorithm work for problems such as Multiway Cut and others, which are generalizations of VERTEX COVER . For those problems, this LP-based branching approach is the only known way to achieve $\mathcal{O}^*(2^k)$-time algorithm.

k-PATH

Instance: a graph $G = (V, E)$, a positive integer k.

Question: Does G have a path on k vertices?