CS 5003: Parameterized Algorithms

Lectures 41-44

Krithika Ramaswamy

IIT Palakkad

Reference Books: Parameterized Algorithms by Cygan et al. and Kernelization by Fomin et al.

The Story So Far

- Parameterization by solution size
  - Vertex Cover
  - Feedback Vertex Set in undirected graphs and tournaments
  - Feedback Arc Set in tournaments
  - Odd Cycle Transversal
  - d-Hitting Set and d-Set Packing
  - Cluster Editing and d-Clustering
  - Long Path
- Parameterization by treewidth of input graph
  - Independent Set, Dominating Set, Clique
- Other parameterizations
  - Vertex Cover: $k - lp(G)$, $k - |M|$

What Next?

* FPT or Not?
  * Is Independent Set parameterized by solution size FPT?
    * Suppose not. Then, P ≠ NP
  * Is Clique parameterized by solution size FPT?
    * No unless P = NP?
  * Is Dominating Set parameterized by solution size FPT?
* FPT running time lower bounds
  * Can Vertex Cover be solved in $O^*(2^{o(k)})$ time?
* Ingredients to build a complexity theory for parameterized problems
  * An useful notion of reduction
  * A reasonable hypothesis

Parameterized Reductions

Let Q and Q′ be 2 parameterized problems. A parameterized reduction from Q to Q′ is an algorithm that given instance (x,k) of Q, outputs instance (y,r) of Q′ in $f(k)\,|x|^{O(1)}$ time for some computable function f s.t

* (x, k) is a yes-instance of Q if and only if (y, r) is a yes-instance of Q′
* $r \leq g(k)$ for some computable function g

|y| <= f(k)|x|^{O(1)} as we neet to write an instance. And h(r) <= h'(k) as r <= g(k).

Instance (x,k) of Q → $f(k)\,|x|^{O(1)}$ time → Instance (y,r) of Q′ $r \leq g(k)$

Facts

* If Q′ is FPT then Q is FPT
* If Q is not FPT then Q′ is not FPT

NP-hard reductions (polynomial-time reductions) are not necessarily helpful to make such conclusions

Parameterized Reductions

Parameterized reductions may not necessarily imply NP-hardness

Clique
Instance: An undirected graph G on n vertices and an integer k
Question: Does G have a clique of at least k vertices?
Parameter: k

Log-Clique
Instance: An undirected graph G on n vertices and an integer k
Question: Is it true that k<=log n and G has a clique of at least k vertices?
Parameter: k

log clique is NP-hard and thus it cannot have quasi polynomial time

Clique reduces to Log-Clique

* Add 2^k isolated vertices to the instance (G,k) of Clique to get the instance (H,k) of Log-Clique. Here, k <= log |V(H)|
* (G,k) is yes-instance iff (H,k) is yes-instance
* Clique is NP-hard but Log-Clique has a quasi-polynomial time ($|V(H)|^{O(\log |V(H)|)}$) algorithm

Parameterized reductions and polynomial-time reductions are incomparable

A Graph is regular if every vertex has same degree.

Parameterized Reductions

- Clique $\leq_{FPT}$ Independent Set
  - (G, k) is yes-instance of Clique iff (G^c, k) is yes-instance of Indep Set
- Independent Set $\leq_{FPT}$ Clique
  - (G, k) is yes-instance of Indep Set iff (G^c, k) is yes-instance of Clique
- Independent Set in regular graphs $\leq_{FPT}$ Clique in regular graphs
- Clique in regular graphs $\leq_{FPT}$ Independent Set in regular graphs

same reduc

- Independent Set in regular graphs $\leq_{FPT}$ Partial Vertex Cover

see image

  - (G, k) is yes-instance of Indep Set iff (G, k, dk) is yes-instance of PVC

    where G is a d-regular graph

Parameterized Reductions

see image for all these 4 reductions.

- Clique $\leq_{FPT}$ Clique in regular graphs
- Clique in regular graphs $\leq_{FPT}$ Multicolored Clique in regular graphs
- Multicolored Clique $\leq_{FPT}$ Multicolored Independent Set
- Multicolored Independent Set $\leq_{FPT}$ Dominating Set

Multicolored Clique

Instance: A graph G with a partition of $V(G)$ into k parts $V_1, V_2, \ldots, V_k$

Question: Does G have a clique Q of size k s.t $|Q \cap V_i| = 1$ for each $i \in [k]$?

Parameter: k

Fixed-Parameter Intractability

* Independent Set, Independent Set in regular graphs
* Clique in regular graphs, Partial Vertex Cover
* Multicolored Clique in regular graphs
* Multicolored Clique in regular graphs
* Dominating Set

are all at least as hard as Clique. I.e., if any of these problems is FPT, then Clique is FPT

* Does Dominating Set $\leq_{FPT}$ Independent Set?
* Or is there a hierarchy among the set of fixed-parameter intractable problems?
  * Do Independent Set and Dominating Set occupy different levels of this hierarchy?
* W-hierarchy: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \ldots \subseteq$ XP $\cap$ paraNP
  * Defined based on boolean circuits
  * Independent Set is in W[1] and Dominating Set is in W[2]

The W-hierarchy

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq XP \cap paraNP$$

* XP = set of parameterized problems solvable in $n^{f(k)}$ time
* paraNP = set of parameterized problems whose unparameterized variant is in NP and is NP-complete for a finite values of the parameter
* FPT $\subset$ XP and it is believed that XP and paraNP are incomparable

Fixed-Parameter Intractability

Ingredients to build a complexity theory for parameterized problems

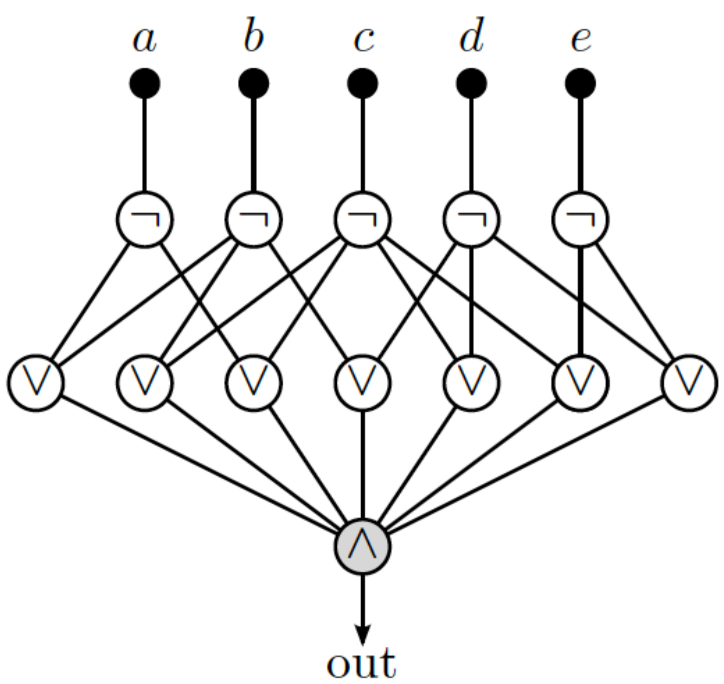
- Parameterized reductions
- Hypotheses: FPT $\subset$ W[1] $\subset$ W[2] $\subset \ldots \subset$ XP

* Partial Vertex Cover, Independent Set, Multicolored Clique, Multicolored Independent Set parameterized by solution size are not FPT unless FPT=W[1]
* Dominating Set, Set Cover, Hitting Set, Dominating Set on Tournaments, Connected Dominating Set parameterized by solution size are not FPT unless FPT=W[2]

Circuits

Definition: A Boolean circuit C is a DAG where the nodes have labels

* Every node with indegree 0 is an input node (input gate)
* Every node with indegree 1 is a negation node (negation gate)
* Every node with indegree >=2 is either an and-node or an or-node (AND gate or OR gate)
* Exactly one of the nodes with outdegree 0 is labeled as the output node (output gate)



edges are directed from top to bottom

* Depth(C) is the max no. of edges on a path from an input to the output
* Weft(C) is the max no. of gates with >2 inputs on a path from an input to the output

Circuit Satisfiability: Given a boolean circuit C, is there an assignment to the inputs of C that makes the output 1?

Image Source: Parameterized Algorithms by Cygan et al., Springer, 2015

Circuits

Circuit Satisfiability: Given a boolean circuit C, is there an assignment to the inputs of C that makes the output 1?

Defn: The weight of an assignment is the number of inputs that are assigned 1

Weighted Circuit Satisfiability: Given a boolean circuit C and an integer k, is there an assignment to the inputs of C of weight k that makes the output 1?

The levels of the W-hierarchy are defined by restricting Weighted Circuit Satisfiability to various classes of circuits.

Circuits

for diagram b -> output is 1 if all the or gates evaluate to true which will happen if the input bits are zero

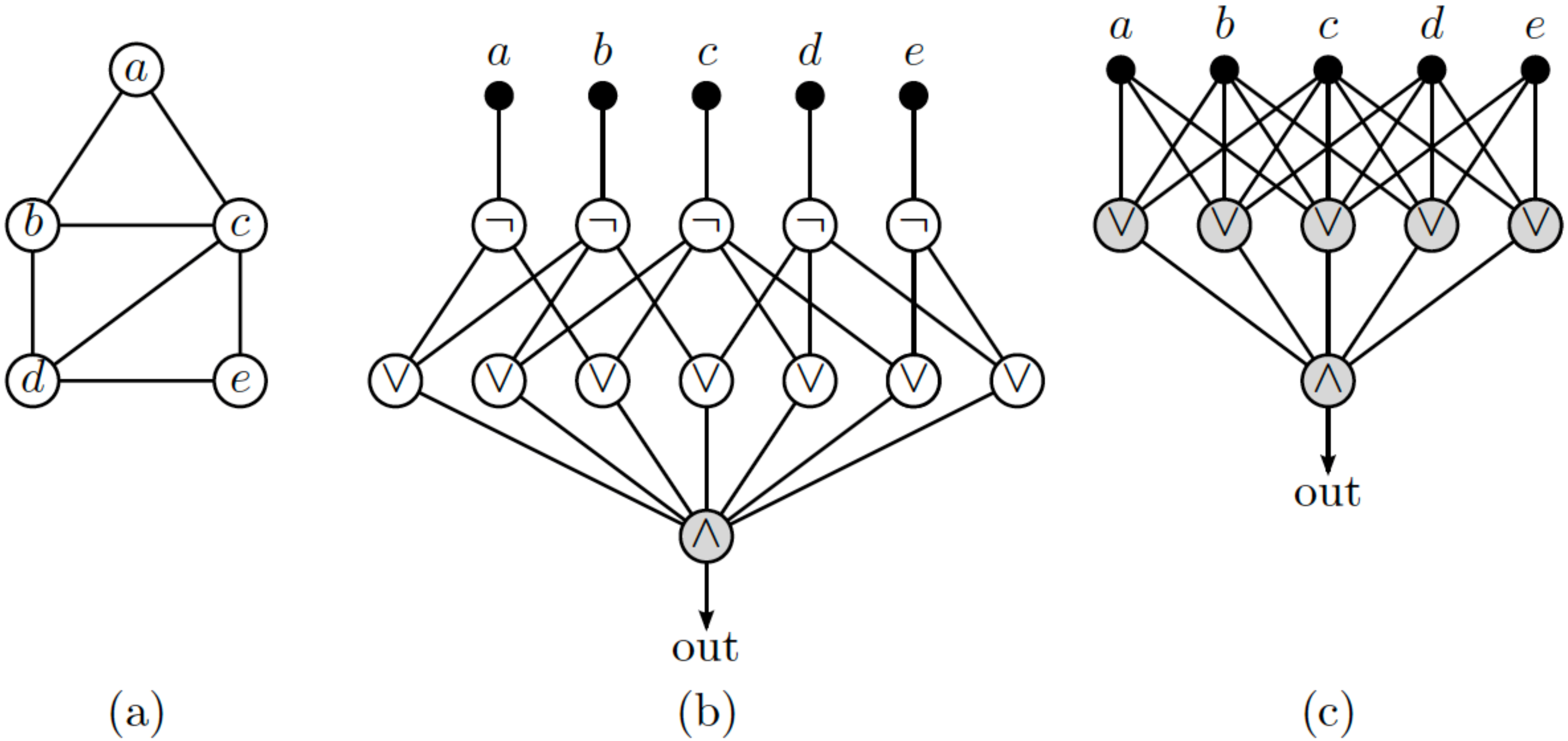


Fig. 13.3: (a) A graph G with five vertices and seven edges. (b) A depth-3, weft-1 circuit satisfied by the independent sets of G. Each or-node corresponds to an edge of G and has indegree 2. (c) A depth-2, weft-2 circuit satisfied by the dominating sets of G. Each or-node corresponds to a vertex of G and its inneighbors correspond to the closed neighborhood to the vertex

Image Source: Parameterized Algorithms by Cygan et al., Springer, 2015

The W-hierarchy

* Independent Set and Dominating Set reduce to Weighted Circuit Satisfiability
  * Circuit for Dom Set is more complicated than the one for Indep Set
* Let C(t,d) denote the class of circuits with weft <=t and depth <=d
* Defn: (W-hierarchy) For t>=1, a parameterized problem Q belongs to the class W[t] if there is a parameterized reduction from Q to Weighted Circuit Satisfiability on C(t,d) for some d>=1.
* Independent Set is in W[1] and Dominating Set is in W[2]
* To show that a problem is in W[t], reduce it to a problem in W[t] in $f(k)n^{O(1)}$ time
  * FPT ⊆ W[1] as any problem in FPT reduces to CLIQUE in $f(k)n^{O(1)}$ time

or more preferably run FPT algo (a) of Q to get yes, no answer and give trivial circuits for each of these.

The W-hierarchy

Defn: (W[t]-hardness) A parameterized problem Q is W[t]-hard if for every problem Q′ in W[t], there is a parameterized reduction from Q′ to Q

W[t]-completeness

i.e. the problem is in W[t] and is W[t] hard.

* Independent Set, Clique and Partial Vertex Cover are W[1]-complete
* Dominating Set, Hitting Set and Set Cover are W[2]-complete
* To show that Q is W[t]-hard, reduce a problem already known to be W[t]-hard to Q in $f(k)n^{O(1)}$ time
* If any W[1]-complete problem is in FPT then FPT=W[1]
* If any W[2]-complete problem is in W[1] then W[1]=W[2]
    * A parameterized reduction from Dominating Set to Independent Set is unlikely

In classical complexity theory, NP-complete problems are equivalent, but in parameterized complexity theory, there is a hierarchy of hard problems

Lower Bounds

cnf = and of clauses, where clauses = or of literals.

Can we show that Vertex Cover cannot be solved in $O^*(2^{o(k)})$ time?

Yes, under an assumption stronger than FPT ≠ W[1]

3-CNF-SAT: Given a boolean formula in conjunctive normal form with <= 3 literals in each clause, is there an assignment to the variables such that the formula evaluates to 1?

* (consequence of) Exponential Time Hypothesis (ETH): 3-CNF-SAT cannot be solved in time subexponential in the number of variables

  i.e. 2^{small oh(n)} not possible.

* (consequence of) Strong Exponential Time Hypothesis (SETH): CNF-SAT cannot be solved in $o(2^n)$ time where n is the number of variables

  i.e. (2 - eps)^{n} not possible.

* (consequence of Sparcification Lemma [Impagliazzo, Paturi, Zane 01]): 3-CNF-SAT cannot be solved in $2^{o(m+n)}$-time where n is the no. of vars and m is no. of clauses
* SETH $\Rightarrow$ ETH $\Rightarrow$ FPT $\neq$ W[1] $\Rightarrow$ P $\neq$ NP
  * FPT $\neq$ W[1] is used to show that no $O^*(f(k))$ algorithm exists
  * ETH is used to show that no $O^*(2^{o(f(k))})$ algorithm exists
  * SETH is used to show that no $O^*(2^{(1-c)f(k)})$ algorithm exists

Kernelization Lower Bounds

- A parameterized problem L is a subset of $\Sigma^* \times \mathbb{N}$ where Σ is a finite alphabet
- A kernelization algorithm for L is a polynomial time algorithm that given instance (x, k) of L outputs instance (x', k') of L s.t
  - $(x, k) \in L$ iff $(x', k') \in L$
  - $|x'| + k' \leq g(k)$ for some computable function g
- L is FPT iff L has a kernelization algorithm

Question 1: Does L admit a polynomial kernel? i.e. g(k) is a polynomial function of k.

Question 2: What is the smallest kernel does L admit?

Kernelization Lower Bounds

Long Path: given a graph G and integer k, does G have a k-path? i.e. k simple path.

* Suppose Long Path has a kernel with no. of vertices <= k^5
* Take $t = k^{11}$ instances $(G_1, k), (G_2, k), \ldots, (G_t, k)$ of Long Path
* Let H be the disjoint union of $G_1, G_2, \ldots, G_t$
* (H, k) is yes-instance iff there exists i s.t (G_i, k) is yes-instance
* Let (H′, k′) be the kernel of (H, k)
* H′ has <= k^5 vertices encodable in k^{10} (< t) bits
* Most of the input instances have been discarded!
  * Unlikely as Longest Path is NP-hard

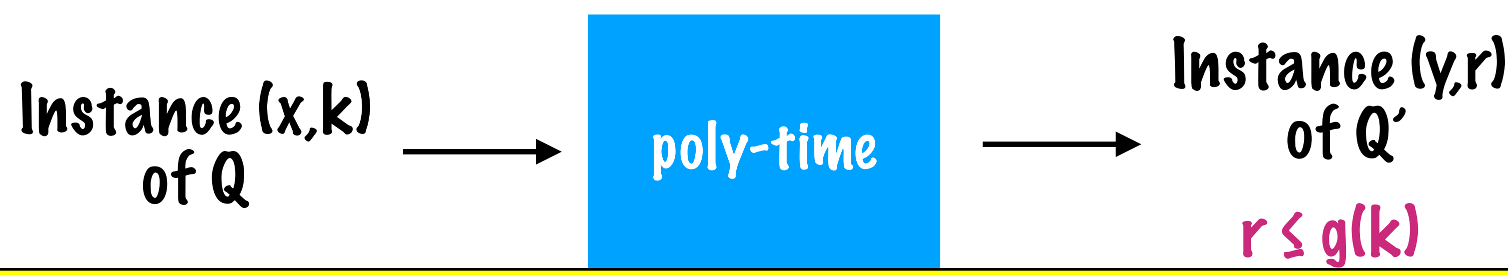
ignore the below theorem

Theorem: Long Path parameterized by k does not admit a polynomial kernel unless coNP⊆NP/poly.

Polynomial Parameter Transformations

Let Q and Q′ be 2 parameterized problems. A polynomial parameter transformation from Q to Q′ is a polynomial-time algorithm that given instance (x,k) of Q, outputs instance (y,r) of Q′ s.t

* (x, k) is a yes-instance of Q if and only if (y, r) is a yes-instance of Q′
* $r \leq g(k)$ for some polynomial function g



if Q is in NP hard and Q' is in NP then the below claim is true.

Facts

* If Q′ has poly kernel then Q has poly kernel - why?
* If Q does not have poly kernel then Q′ does not have poly kernel

PPT: An Example

- Long Path $\leq_{PPT}$ Path Packing
  - Given instance (G,k) of Long Path
    - Add k-1 vertex disjoint paths of length k to G to get G′
    - (G,k) is yes-instance of Long Path iff (G′,k) is yes-instance of Path Packing

long path is NP hard and path packing is in NP.

Path Packing

Instance: A graph G and an integer k

Question: Does G have a set of k vertex disjoint k-paths?

Parameter: k

again these k paths are simple.