Parametrized Complexity
Part 2

Aeren

October 18, 2022

Correction

Correction

There was a (possible) error in the definition of the parameterized reduction.

Correction

There was a (possible) error in the definition of the parameterized reduction.

DEFINITION

Let A and B be parameterized problems. A **parameterized reduction** of A onto B maps an instance (x, k) of A to an instance (x', k') of B such that

1. x' depends only on x.
2. k' depends only on k.
3. The function that maps x to x' can be computed in $f(k) \cdot poly(|x|)$ time.
4. k' is bounded by a computable function of k.
5. $(x, k) \in A$ if and only if $(x', k') \in B$.

Introduction

Introduction

- As with polynomial-time-reduction, parameterized reduction also yields its own hierarchical complexity class structure, which will be the main topic of today's lecture.

The Complexity Class W[1]

[PROBLEM] Nondeterministic-Turing-Machine-Acceptance(NTMA$_k$)

The Complexity Class W[1]

[PROBLEM] Nondeterministic-Turing-Machine-Acceptance(NTMA_k)

Instance: A nondeterministic turing machine with $O(n)$ states, alphabets, and choices at each step, and an integer $k \geq 0$.

The Complexity Class W[1]

[PROBLEM] Nondeterministic-Turing-Machine-Acceptance(NTMA_k)

Instance: A nondeterministic turing machine with $O(n)$ states, alphabets, and choices at each step, and an integer $k \geq 0$.

Question: Is there an accepting path of length k for 0^n?

The Complexity Class W[1]

[PROBLEM] Nondeterministic-Turing-Machine-Acceptance(NTMA_k)

Instance: A nondeterministic turing machine with $O(n)$ states, alphabets, and choices at each step, and an integer $k \geq 0$.
Question: Is there an accepting path of length k for 0^n?

- It is believed, though not as strong as P$\neq$NP, that $\mathrm{NTMA}_k \notin$FPT.

The Complexity Class W[1]

[PROBLEM] Nondeterministic-Turing-Machine-Acceptance(NTMA_k)

Instance: A nondeterministic turing machine with $O(n)$ states, alphabets, and choices at each step, and an integer $k \geq 0$.
Question: Is there an accepting path of length k for 0^n?

- It is believed, though not as strong as P$\neq$NP, that $\mathrm{NTMA}_k \notin$FPT.
- We can now define a new complexity class under the assumption that $\mathrm{NTMA}_k \notin$FPT.

The Complexity Class W[1]

Polynomial Time Reduction	Parameterized Reduction

The Complexity Class W[1]

Polynomial Time Reduction	Parameterized Reduction
Problem A is **NP** if A is reducible to SAT	Problem A is **W[1]** if A is reducible to NTMA_k

The Complexity Class W[1]

Polynomial Time Reduction	Parameterized Reduction
Problem A is **NP** if A is reducible to SAT	Problem A is **W[1]** if A is reducible to NTMA_k
Problem A is **NP-hard** if SAT is reducible to A	Problem A is **W[1]-hard** if NTMA_k is reducible to A

The Complexity Class W[1]

Polynomial Time Reduction	Parameterized Reduction
Problem A is **NP** if A is reducible to SAT	Problem A is **W[1]** if A is reducible to NTMA$_k$
Problem A is **NP-hard** if SAT is reducible to A	Problem A is **W[1]-hard** if NTMA$_k$ is reducible to A
Problem A is **NP-complete** if A is NP and NP-hard	Problem A is **W[1]-complete** if A is W[1] and W[1]-hard

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]

- Given (G, k), set up a nondeterministic turing machine which have G built into it and guesses k vertices while guaranteeing that the set of guessed vertices is independent at each guesses.

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k \in$ W[1]

- Given (G, k), set up a nondeterministic turing machine which have G built into it and guesses k vertices while guaranteeing that the set of guessed vertices is independent at each guesses.
- As any independent set of size k can be found, if any, within path length of $O(k^2)$, we conclude that Independent-Set$_k$ $\in$W[1].

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]-hard

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]-hard

- Let Σ be the set of alphabet and Q the set of state of the given nondeterministic turing machine.

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]-hard

- Let Σ be the set of alphabet and Q the set of state of the given nondeterministic turing machine.
- Let a configuration of the machine be an element of $\Sigma^*(\Sigma \times Q)\Sigma^*$ describing the content of the tape, position of the head, and the state of the machine.

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]-hard

- Let Σ be the set of alphabet and Q the set of state of the given nondeterministic turing machine.
- Let a configuration of the machine be an element of $\Sigma^*(\Sigma \times Q)\Sigma^*$ describing the content of the tape, position of the head, and the state of the machine.
- The vertices of the graph will be grouped into k^2 cells labelled $\{1, 2, \cdots, k\} \times \{1, 2, \cdots, k\}$, where vertices in each cell form a clique.

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k \in$ W[1]-hard

- Let Σ be the set of alphabet and Q the set of state of the given nondeterministic turing machine.
- Let a configuration of the machine be an element of $\Sigma^*(\Sigma \times Q)\Sigma^*$ describing the content of the tape, position of the head, and the state of the machine.
- The vertices of the graph will be grouped into k^2 cells labelled $\{1, 2, \cdots, k\} \times \{1, 2, \cdots, k\}$, where vertices in each cell form a clique.
- By adding some more edges, we want this graph to have an independent set of size $k \times k$ if and only if the machine has an accepting path of length k.

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]-hard

- The cell (i, j) represent the state of the tape after i steps, and has a vertex for each element in $\Sigma \cup (\Sigma \times Q)$.

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]-hard

- The cell (i, j) represent the state of the tape after i steps, and has a vertex for each element in $\Sigma \cup (\Sigma \times Q)$.
- We add edges between every pair of vertices which is impossible to occur together. i.e.

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]-hard

- The cell (i, j) represent the state of the tape after i steps, and has a vertex for each element in $\Sigma \cup (\Sigma \times Q)$.
- We add edges between every pair of vertices which is impossible to occur together. i.e.
  1. For all $1 \leq i \leq k$ and $1 \leq j_1 < j_2 \leq k$, add edges between every vertices from $\Sigma \times Q$ in the cell (i, j_1) and in the cell (i, j_2).

The Complexity Class W[1]

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k \in$ W[1]-hard

- The cell (i, j) represent the state of the tape after i steps, and has a vertex for each element in $\Sigma \cup (\Sigma \times Q)$.
- We add edges between every pair of vertices which is impossible to occur together. i.e.
  1. For all $1 \le i \le k$ and $1 \le j_1 < j_2 \le k$, add edges between every vertices from $\Sigma \times Q$ in the cell (i, j_1) and in the cell (i, j_2).
  2. For all $1 \le i < k$ and j and $(p, a) \in \Sigma \times Q$, put edges between (p, a) at cell (i, j) and every vertices at the row $i + 1$ which is incompatible with (p, a) according to the transition function of the machine.

THEOREM

Independent-Set$_k$ is W[1]-complete.

PROOF) Independent-Set$_k$ $\in$ W[1]-hard

- The cell (i, j) represent the state of the tape after i steps, and has a vertex for each element in $\Sigma \cup (\Sigma \times Q)$.
- We add edges between every pair of vertices which is impossible to occur together. i.e.
  1. For all $1 \leq i \leq k$ and $1 \leq j_1 < j_2 \leq k$, add edges between every vertices from $\Sigma \times Q$ in the cell (i, j_1) and in the cell (i, j_2).
  2. For all $1 \leq i < k$ and j and $(p, a) \in \Sigma \times Q$, put edges between (p, a) at cell (i, j) and every vertices at the row $i + 1$ which is incompatible with (p, a) according to the transition function of the machine.
- Clearly, this graph has a clique of size k^2 if and only if the initial machine has an accepting path of length k. Therefore, Independent-Set$_k$ is W[1]-complete.

■

The Complexity Class W[1]

COROLLARY

Clique$_k$ is W[1]-complete.

The Complexity Class W[1]

THEOREM

Regular-Clique$_k$ is W[1]-complete.

The Complexity Class W[1]

THEOREM

Regular-Clique$_k$ is W[1]-complete.

PROOF)

- Clique$_k$ $\in$W[1] implies Regular-Clique$_k$ $\in$W[1], so it's suffice to show that Regular-Clique$_k$ $\in$W[1]-hard.

The Complexity Class W[1]

THEOREM

Regular-Clique$_k$ is W[1]-complete.

PROOF)

- Clique$_k$ $\in$W[1] implies Regular-Clique$_k$ $\in$W[1], so it's suffice to show that Regular-Clique$_k$ $\in$W[1]-hard.
- Given the input (G, k), let Δ be the maximum degree of a vertex of G.

The Complexity Class W[1]

THEOREM

Regular-Clique$_k$ is W[1]-complete.

PROOF)

- Clique$_k$ $\in$W[1] implies Regular-Clique$_k$ $\in$W[1], so it's suffice to show that Regular-Clique$_k$ $\in$W[1]-hard.
- Given the input (G, k), let Δ be the maximum degree of a vertex of G.
- Create Δ copies of G: $G_1, \cdots, G_\Delta$, and let v_i be the vertex corresponding to v in G_i.

The Complexity Class W[1]

THEOREM

Regular-Clique$_k$ is W[1]-complete.

PROOF)

- Clique$_k$ $\in$W[1] implies Regular-Clique$_k$ $\in$W[1], so it's suffice to show that Regular-Clique$_k$ $\in$W[1]-hard.
- Given the input (G, k), let Δ be the maximum degree of a vertex of G.
- Create Δ copies of G: $G_1, \cdots, G_\Delta$, and let v_i be the vertex corresponding to v in G_i.
- For each vertex v of G, create $\Delta - \deg(v)$ new vertices $v'_1, \cdots, v'_{\Delta-\deg(v)}$, and connect each v_i and v'_j.

The Complexity Class W[1]

THEOREM

Regular-Clique$_k$ is W[1]-complete.

PROOF)

- Clique$_k$ $\in$W[1] implies Regular-Clique$_k$ $\in$W[1], so it's suffice to show that Regular-Clique$_k$ $\in$W[1]-hard.
- Given the input (G, k), let Δ be the maximum degree of a vertex of G.
- Create Δ copies of G: $G_1, \cdots, G_\Delta$, and let v_i be the vertex corresponding to v in G_i.
- For each vertex v of G, create $\Delta - \deg(v)$ new vertices $v'_1, \cdots, v'_{\Delta-\deg(v)}$, and connect each v_i and v'_j.
- Since for each $l \geq 3$, all clique of size l of the resulting graph is contained in one of G_is, it has a clique of size k if and only if the original graph has a clique of size k.

■

COROLLARY

Regular-Independent-Set$_k$ is W[1]-complete.

The Complexity Class W[1]

[PROBLEM] Multicolored-Clique$_k$(resp. Multicolored-Independent-Set$_k$)

The Complexity Class W[1]

[PROBLEM] Multicolored-Clique$_k$(resp. Multicolored-Independent-Set$_k$)

Instance: A graph $G = (V, E)$ and a partition $V = V_1 \cup \cdots \cup V_k$.

The Complexity Class W[1]

[PROBLEM] Multicolored-Clique$_k$(resp. Multicolored-Independent-Set$_k$)

Instance: A graph $G = (V, E)$ and a partition $V = V_1 \cup \cdots \cup V_k$.
Question: Is there a clique(resp. independent set) with one vertex from each V_i?

The Complexity Class W[1]

[PROBLEM] Multicolored-Clique$_k$(resp. Multicolored-Independent-Set$_k$)

Instance: A graph $G = (V, E)$ and a partition $V = V_1 \cup \cdots \cup V_k$.
Question: Is there a clique(resp. independent set) with one vertex from each V_i?

THEOREM

Multicolored-Clique$_k$(resp. Multicolored-Independent-Set$_k$) is W[1]-complete.

The Complexity Class W[1]

[PROBLEM] Multicolored-Clique$_k$(resp. Multicolored-Independent-Set$_k$)

Instance: A graph $G = (V, E)$ and a partition $V = V_1 \cup \cdots \cup V_k$.
Question: Is there a clique(resp. independent set) with one vertex from each V_i?

THEOREM

Multicolored-Clique$_k$(resp. Multicolored-Independent-Set$_k$) is W[1]-complete.

PROOF) Trivial.
■

W-Hierarchy

We've looked at W[1]-complete problems so far. Now we'll look at some problems which is believed not to be W[1]-complete.

W-Hierarchy

We've looked at W[1]-complete problems so far. Now we'll look at some problems which is believed not to be W[1]-complete.

[PROBLEM] Dominating-Set$_k$

W-Hierarchy

We've looked at W[1]-complete problems so far. Now we'll look at some problems which is believed not to be W[1]-complete.

[PROBLEM] Dominating-Set$_k$

Instance: A graph $G = (V, E)$ and an integer $k \geq 0$.

W-Hierarchy

We've looked at W[1]-complete problems so far. Now we'll look at some problems which is believed not to be W[1]-complete.

[PROBLEM] Dominating-Set$_k$

Instance: A graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Is there a set of vertices D, called a **dominating set**, of size k where every vertex u of G is either in D or adjacent to a vertex in D?

W-Hierarchy

We've looked at W[1]-complete problems so far. Now we'll look at some problems which is believed not to be W[1]-complete.

[PROBLEM] Dominating-Set$_k$

Instance: A graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Is there a set of vertices D, called a **dominating set**, of size k where every vertex u of G is either in D or adjacent to a vertex in D?

- It is unlikely that Dominating-Set$_k$ is W[1]: after guessing k vertices for the dominating set D, we have to check whether every vertex is adjacent to D, which takes $O(k \cdot n)$ time.

W-Hierarchy

We've looked at W[1]-complete problems so far. Now we'll look at some problems which is believed not to be W[1]-complete.

[PROBLEM] Dominating-Set$_k$

Instance: A graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Is there a set of vertices D, called a **dominating set**, of size k where every vertex u of G is either in D or adjacent to a vertex in D?

- It is unlikely that Dominating-Set$_k$ is W[1]: after guessing k vertices for the dominating set D, we have to check whether every vertex is adjacent to D, which takes $O(k \cdot n)$ time.
- In case of Clique$_k$, it is W[1] because it is "local": after guessing k vertices, we don't have to look at every other vertices.

W-Hierarchy

We've looked at W[1]-complete problems so far. Now we'll look at some problems which is believed not to be W[1]-complete.

[PROBLEM] Dominating-Set$_k$

Instance: A graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Is there a set of vertices D, called a **dominating set**, of size k where every vertex u of G is either in D or adjacent to a vertex in D?

- It is unlikely that Dominating-Set$_k$ is W[1]: after guessing k vertices for the dominating set D, we have to check whether every vertex is adjacent to D, which takes $O(k \cdot n)$ time.
- In case of Clique$_k$, it is W[1] because it is "local": after guessing k vertices, we don't have to look at every other vertices.
- We'll later define the class W[2], where Dominating-Set$_k$ has been proven to be complete on.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

- Input: $G = (V, E)$ with the partition $V = V_1 \cup \cdots \cup V_k$.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

- Input: $G = (V, E)$ with the partition $V = V_1 \cup \cdots \cup V_k$.
- Create a graph G' by adding some vertices and edges as follows:
  1. Connect every pair of vertices in V_i for all $1 \leq i \leq k$.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

- Input: $G = (V, E)$ with the partition $V = V_1 \cup \cdots \cup V_k$.
- Create a graph G' by adding some vertices and edges as follows:
  1. Connect every pair of vertices in V_i for all $1 \leq i \leq k$.
  2. Create new vertices x_i and y_i for all $1 \leq i \leq k$.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

- Input: $G = (V, E)$ with the partition $V = V_1 \cup \cdots \cup V_k$.
- Create a graph G' by adding some vertices and edges as follows:
  1. Connect every pair of vertices in V_i for all $1 \leq i \leq k$.
  2. Create new vertices x_i and y_i for all $1 \leq i \leq k$.
  3. For all $v_i \in V_i$, connect v_i with both x_i and y_i.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

- Input: $G = (V, E)$ with the partition $V = V_1 \cup \cdots \cup V_k$.
- Create a graph G' by adding some vertices and edges as follows:
  1. Connect every pair of vertices in V_i for all $1 \leq i \leq k$.
  2. Create new vertices x_i and y_i for all $1 \leq i \leq k$.
  3. For all $v_i \in V_i$, connect v_i with both x_i and y_i.
  4. For every edge $e = (u, v)$ with $u \in V_i$, $v \in V_j$, and $i \neq j$, create a new vertex w_e.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

- Input: $G = (V, E)$ with the partition $V = V_1 \cup \cdots \cup V_k$.
- Create a graph G' by adding some vertices and edges as follows:
  1. Connect every pair of vertices in V_i for all $1 \leq i \leq k$.
  2. Create new vertices x_i and y_i for all $1 \leq i \leq k$.
  3. For all $v_i \in V_i$, connect v_i with both x_i and y_i.
  4. For every edge $e = (u, v)$ with $u \in V_i$, $v \in V_j$, and $i \neq j$, create a new vertex w_e.
  5. Connect w_e with every vertex of $V_i - \{u\}$ and $V_j - \{v\}$.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

- Input: $G = (V, E)$ with the partition $V = V_1 \cup \cdots \cup V_k$.
- Create a graph G' by adding some vertices and edges as follows:
  1. Connect every pair of vertices in V_i for all $1 \leq i \leq k$.
  2. Create new vertices x_i and y_i for all $1 \leq i \leq k$.
  3. For all $v_i \in V_i$, connect v_i with both x_i and y_i.
  4. For every edge $e = (u, v)$ with $u \in V_i$, $v \in V_j$, and $i \neq j$, create a new vertex w_e.
  5. Connect w_e with every vertex of $V_i - \{u\}$ and $V_j - \{v\}$.
- As there's no edge between x_i and y_i, every dominating set of size k in the resulting graph must contain exactly one vertex from each V_i.

W-Hierarchy

THEOREM

Dominating-Set$_k$ is W[1]-hard.

PROOF) We reduce Multicolored-Independent-Set$_k$ to Dominating-Set$_k$.

- Input: $G = (V, E)$ with the partition $V = V_1 \cup \cdots \cup V_k$.
- Create a graph G' by adding some vertices and edges as follows:
    1. Connect every pair of vertices in V_i for all $1 \leq i \leq k$.
    2. Create new vertices x_i and y_i for all $1 \leq i \leq k$.
    3. For all $v_i \in V_i$, connect v_i with both x_i and y_i.
    4. For every edge $e = (u, v)$ with $u \in V_i$, $v \in V_j$, and $i \neq j$, create a new vertex w_e.
    5. Connect w_e with every vertex of $V_i - \{u\}$ and $V_j - \{v\}$.
- As there's no edge between x_i and y_i, every dominating set of size k in the resulting graph must contain exactly one vertex from each V_i.
- Now it is clear from the construction that the original graph has an independent set of size k if and only if the resulting graph has a dominating set of size k.

■

W-Hierarchy

[PROBLEM] Circuit-SAT$_k$

W-Hierarchy

[PROBLEM] Circuit-SAT$_k$

Instance: A boolean circuit C and an integer $k \geq 0$.

W-Hierarchy

[PROBLEM] Circuit-SAT$_k$

Instance: A boolean circuit C and an integer $k \geq 0$.
Question: Is there a valid assignment of the input where exactly k variables are set to True?

W-Hierarchy

THEOREM

1. Independent-Set$_k$ is reducible to Circuit-SAT$_k$, i.e. it is W[1]-hard.

W-Hierarchy

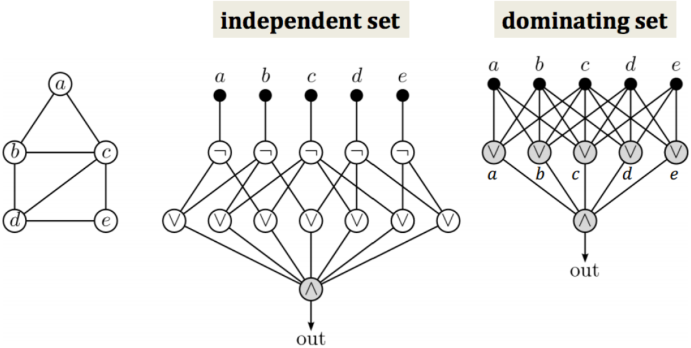
THEOREM

1. Independent-Set$_k$ is reducible to Circuit-SAT$_k$, i.e. it is W[1]-hard.
2. Dominating-Set$_k$ is reducible to Circuit-SAT$_k$.

W-Hierarchy

THEOREM

1. Independent-Set$_k$ is reducible to Circuit-SAT$_k$, i.e. it is W[1]-hard.
2. Dominating-Set$_k$ is reducible to Circuit-SAT$_k$.



W-Hierarchy

So far, we have:

W-Hierarchy

So far, we have:

- W[1]-complete
  - NTMA_k, $\{$Regular-, Multicolored-$\}\{\text{Clique}_k$, $\text{Independent-Set}_k\}$

W-Hierarchy

So far, we have:

- W[1]-complete
  - NTMA_k, $\{$Regular-, Multicolored-$\}\{\text{Clique}_k$, $\text{Independent-Set}_k\}$
- W[1]-hard but doesn't seem to be W[1]
  - Dominating-Set_k, Circuit-SAT_k

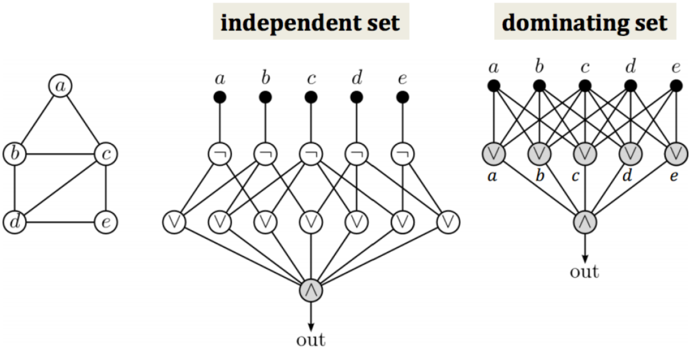
W-Hierarchy

So far, we have:

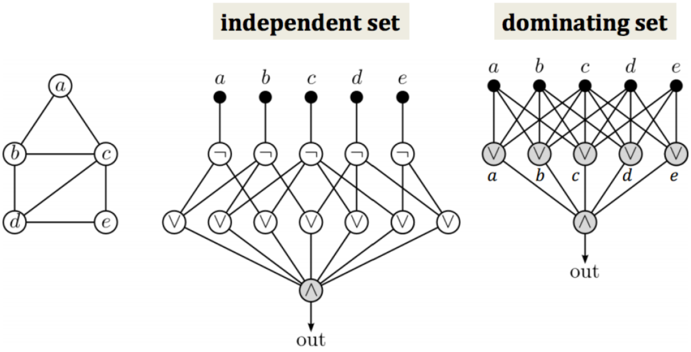
- W[1]-complete
  - NTMA_k, {Regular-, Multicolored-}{Clique_k, Independent-Set_k}
- W[1]-hard but doesn't seem to be W[1]
  - Dominating-Set_k, Circuit-SAT_k

As we've noted earlier, locality of Clique_k / Independent-Set_k plays a part in the distinguishment of these two groups.
We formalize this concept to define other classes in W-Hierarchy.

W-Hierarchy

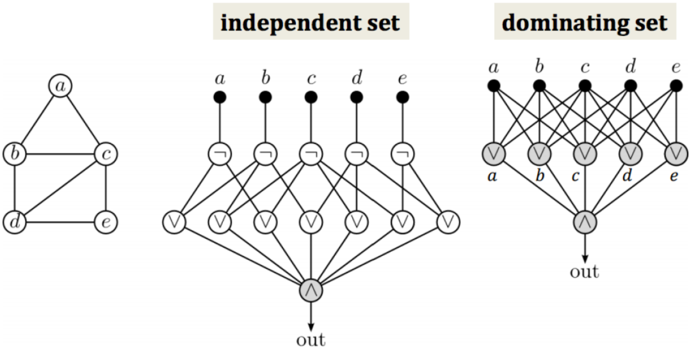


W-Hierarchy



1. In the Independent-Set$_k$ circuit, all the input-output path pass through only one gate that has large number of inputs.

W-Hierarchy



1. In the Independent-Set$_k$ circuit, all the input-output path pass through only one gate that has large number of inputs.
2. In the Dominating-Set$_k$ circuit, all the input-output path pass through two gates that has large number of inputs.

W-Hierarchy

DEFINITION

1. A **large gate** is a gate that has more than some fixed constant number. We let the constant be 2 for now.

W-Hierarchy

DEFINITION

1. A **large gate** is a gate that has more than some fixed constant number. We let the constant be 2 for now.
2. The **depth** of a circuit is the maximum length of an input-output path.

W-Hierarchy

DEFINITION

1. A **large gate** is a gate that has more than some fixed constant number. We let the constant be 2 for now.
2. The **depth** of a circuit is the maximum length of an input-output path.
3. The **weft** of a circuit is the maximum number of large gates on a an input-output path.

W-Hierarchy

DEFINITION

1. A **large gate** is a gate that has more than some fixed constant number. We let the constant be 2 for now.
2. The **depth** of a circuit is the maximum length of an input-output path.
3. The **weft** of a circuit is the maximum number of large gates on a an input-output path.
4. Let C[t,d] be the set of all circuits having weft at most t and depth at most d.

W-Hierarchy

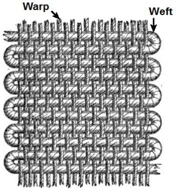
DEFINITION

1. A **large gate** is a gate that has more than some fixed constant number. We let the constant be 2 for now.
2. The **depth** of a circuit is the maximum length of an input-output path.
3. The **weft** of a circuit is the maximum number of large gates on a an input-output path.
4. Let C[t,d] be the set of all circuits having weft at most t and depth at most d.
5. Let $C[t,d]_k$ the set of problems deciding whether there's an assignment for the given circuit in C[t,d] with exactly k variables assigned to True.

W-Hierarchy

DEFINITION

1. A **large gate** is a gate that has more than some fixed constant number. We let the constant be 2 for now.
2. The **depth** of a circuit is the maximum length of an input-output path.
3. The **weft** of a circuit is the maximum number of large gates on a an input-output path.
4. Let C[t,d] be the set of all circuits having weft at most t and depth at most d.
5. Let $C[t,d]_k$ the set of problems deciding whether there's an assignment for the given circuit in C[t,d] with exactly k variables assigned to True.



W-Hierarchy

DEFINITION

Let A_k be a parameterized problem and $t \geq 0$ an integer.

W-Hierarchy

DEFINITION

Let A_k be a parameterized problem and $t \geq 0$ an integer.

1. $A_k \in$ W[t] if there's a computable function f such that there's a reduction from A_k to a problem in C[t,d]$_k$.

W-Hierarchy

DEFINITION

Let A_k be a parameterized problem and $t \geq 0$ an integer.

1. $A_k \in$ W[t] if there's a computable function f such that there's a reduction from A_k to a problem in C[t,d]$_k$.
2. A_k is W[t]-hard if, for all d, every problem in C[t,d]$_k$ is reducible to A_k.

W-Hierarchy

DEFINITION

Let A_k be a parameterized problem and $t \geq 0$ an integer.

1. $A_k \in$ W[t] if there's a computable function f such that there's a reduction from A_k to a problem in C[t,d]$_k$.
2. A_k is W[t]-hard if, for all d, every problem in C[t,d]$_k$ is reducible to A_k.
3. A_k is W[t]-complete, if it is W[t] and W[t]-hard.

W-Hierarchy

DEFINITION

Let A_k be a parameterized problem and $t \geq 0$ an integer.

1. $A_k \in$ W[t] if there's a computable function f such that there's a reduction from A_k to a problem in C[t,d]$_k$.
2. A_k is W[t]-hard if, for all d, every problem in C[t,d]$_k$ is reducible to A_k.
3. A_k is W[t]-complete, if it is W[t] and W[t]-hard.
4. $A_k \in$ W[SAT] if A_k is reducible to SAT$_k$.

W-Hierarchy

DEFINITION

Let A_k be a parameterized problem and $t \geq 0$ an integer.

1. $A_k \in$ W[t] if there's a computable function f such that there's a reduction from A_k to a problem in C[t,d]$_k$.
2. A_k is W[t]-hard if, for all d, every problem in C[t,d]$_k$ is reducible to A_k.
3. A_k is W[t]-complete, if it is W[t] and W[t]-hard.
4. $A_k \in$ W[SAT] if A_k is reducible to SAT$_k$.
5. $A_k \in$ W[P] if A_k is reducible to Circuit-SAT$_k$.

W-Hierarchy

DEFINITION

Let A_k be a parameterized problem and $t \geq 0$ an integer.

1. $A_k \in$ W[t] if there's a computable function f such that there's a reduction from A_k to a problem in C[t,d]$_k$.
2. A_k is W[t]-hard if, for all d, every problem in C[t,d]$_k$ is reducible to A_k.
3. A_k is W[t]-complete, if it is W[t] and W[t]-hard.
4. $A_k \in$ W[SAT] if A_k is reducible to SAT$_k$.
5. $A_k \in$ W[P] if A_k is reducible to Circuit-SAT$_k$.
6. $A_k \in$ XP if there exists computable functions f and g such that A_k can be solved in time $f(k) \cdot n^{g(k)}$.

W-Hierarchy

The followings are known:

W-Hierarchy

The followings are known:

- $\mathrm{FPT} = \mathrm{W}[0] \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[\mathrm{SAT}] \subseteq \mathrm{W}[\mathrm{p}] \subseteq \mathrm{XP}$

W-Hierarchy

The followings are known:

- FPT $=$ W[0] $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[SAT] $\subseteq$ W[p] $\subseteq$ XP
- For integer $t \geq 0$, we can equivalently define W[t] the same way as the NTMA$_k$ definition of W[1], except that the machine can have t tapes.

W-Hierarchy

The followings are known:

- FPT $=$ W[0] $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[SAT] $\subseteq$ W[p] $\subseteq$ XP
- For integer $t \geq 0$, we can equivalently define W[t] the same way as the NTMA$_k$ definition of W[1], except that the machine can have t tapes.
- Dominating-Set$_k$ is W[2]-complete. Hence, it's unlikely that it is reducible to Independent-Set$_k$.

W-Hierarchy

The followings are known:

- FPT $=$ W[0] $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[SAT] $\subseteq$ W[p] $\subseteq$ XP
- For integer $t \geq 0$, we can equivalently define W[t] the same way as the NTMA$_k$ definition of W[1], except that the machine can have t tapes.
- Dominating-Set$_k$ is W[2]-complete. Hence, it's unlikely that it is reducible to Independent-Set$_k$.
- FPT $\neq$ XP

Lower Bounds Under ETH

Lower Bounds Under ETH

THEOREM

Assuming ETH, Vertex-Cover$_k$ requires $2^{\Omega}(k)n^c$ time to solve for all integer $c > 0$.

Lower Bounds Under ETH

THEOREM

Assuming ETH, Vertex-Cover$_k$ requires $2^{\Omega}(k)n^c$ time to solve for all integer $c > 0$.

PROOF)

- Recall that assuming ETH, Vertex-Cover requires $2^{\Omega(n)}$ time to solve.

Lower Bounds Under ETH

THEOREM

Assuming ETH, Vertex-Cover$_k$ requires $2^{\Omega}(k)n^c$ time to solve for all integer $c > 0$.

PROOF)

- Recall that assuming ETH, Vertex-Cover requires $2^{\Omega(n)}$ time to solve.
- If Vertex-Cover$_k$ could be solved in $2^{o(k)} \cdot n^c$ time, since $k \leq n$, this would yield a $2^{o(n)} \cdot n^c$ algorithm for Vertex-Cover, violating the ETH.

■

As Vertex-Cover has been shown to be solvable in time $O(k \cdot n + 1.28^k)$ by Chen et al., this bound is tight.

Lower Bounds Under ETH

Assuming ETH,

Lower Bounds Under ETH

Assuming ETH,

- Similarly to the previous theorem, Dominating-Set$_k$, Clique$_k$, and Hamiltonian-Cycle$_k$ can all be shown to require $2^{\Omega(k)} \cdot n^c$ time to solve for all integer $c > 0$.

Lower Bounds Under ETH

Assuming ETH,

- Similarly to the previous theorem, Dominating-Set$_k$, Clique$_k$, and Hamiltonian-Cycle$_k$ can all be shown to require $2^{\Omega(k)} \cdot n^c$ time to solve for all integer $c > 0$.
- Furthurmore, Planar-Vertex-Cover$_k$, Planar-Dominating-Set$_k$, and Planar-Directed-Hamiltonian-Cycle$_k$ can all be shown to require $2^{\Omega(\sqrt{k})} \cdot n^c$ time to solve for all integer $c > 0$.

Lower Bounds Under ETH

Assuming ETH,

- Similarly to the previous theorem, Dominating-Set$_k$, Clique$_k$, and Hamiltonian-Cycle$_k$ can all be shown to require $2^{\Omega(k)} \cdot n^c$ time to solve for all integer $c > 0$.
- Furthurmore, Planar-Vertex-Cover$_k$, Planar-Dominating-Set$_k$, and Planar-Directed-Hamiltonian-Cycle$_k$ can all be shown to require $2^{\Omega(\sqrt{k})} \cdot n^c$ time to solve for all integer $c > 0$.
- Albert et al. showed that Planar-Dominating-Set$_k$ can be done in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time. Thus, the bound above is tight.

Lower Bounds Under ETH

Assuming ETH,

- Similarly to the previous theorem, Dominating-Set$_k$, Clique$_k$, and Hamiltonian-Cycle$_k$ can all be shown to require $2^{\Omega(k)} \cdot n^c$ time to solve for all integer $c > 0$.
- Furthurmore, Planar-Vertex-Cover$_k$, Planar-Dominating-Set$_k$, and Planar-Directed-Hamiltonian-Cycle$_k$ can all be shown to require $2^{\Omega(\sqrt{k})} \cdot n^c$ time to solve for all integer $c > 0$.
- Albert et al. showed that Planar-Dominating-Set$_k$ can be done in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time. Thus, the bound above is tight.
- Demaine et al. showed that Planar-Vertex-Cover$_k$ can be done in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ time. Thus, the bound above is tight.

Lower Bounds Under ETH

Recall that assuming FPT $\neq$ W[1], Clique$_k$ and Independent-Set$_k$ cannot be solved in $f(k) \cdot n^{O(1)}$ time.

Lower Bounds Under ETH

Recall that assuming FPT $\neq$ W[1], Clique$_k$ and Independent-Set$_k$ cannot be solved in $f(k) \cdot n^{O(1)}$ time.

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

Lower Bounds Under ETH

Recall that assuming FPT $\neq$ W[1], Clique$_k$ and Independent-Set$_k$ cannot be solved in $f(k) \cdot n^{O(1)}$ time.

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

- Recall that 3COL requires $2^{\Omega(n)}$ time assuming ETH.

Lower Bounds Under ETH

Recall that assuming FPT $\neq$ W[1], Clique$_k$ and Independent-Set$_k$ cannot be solved in $f(k) \cdot n^{O(1)}$ time.

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

- Recall that 3COL requires $2^{\Omega(n)}$ time assuming ETH.
- We give a reduction from 3COL to Clique$_k$.

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

1. Input: $G = (V, E)$

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

1. Input: $G = (V, E)$
2. Split V into k groups $V_1, \cdots, V_k$, roughly n/k vertices each.

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

1. Input: $G = (V, E)$
2. Split V into k groups $V_1, \cdots, V_k$, roughly n/k vertices each.
3. For each $1 \leq i \leq k$ and each valid 3-colorings of V_i, add a new vertex to a vertex set V'. This step takes $O(k \cdot 3^{n/k})$ time.

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

1. Input: $G = (V, E)$
2. Split V into k groups $V_1, \cdots, V_k$, roughly n/k vertices each.
3. For each $1 \leq i \leq k$ and each valid 3-colorings of V_i, add a new vertex to a vertex set V'. This step takes $O(k \cdot 3^{n/k})$ time.
4. For each vertex $x \in V_i$ and $y \in V_j$ with $i < j$, if the coloring $x \cup y$ is valid, add the edge (x, y) to an edge set E'.

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

1. Input: $G = (V, E)$
2. Split V into k groups $V_1, \cdots, V_k$, roughly n/k vertices each.
3. For each $1 \leq i \leq k$ and each valid 3-colorings of V_i, add a new vertex to a vertex set V'. This step takes $O(k \cdot 3^{n/k})$ time.
4. For each vertex $x \in V_i$ and $y \in V_j$ with $i < j$, if the coloring $x \cup y$ is valid, add the edge (x, y) to an edge set E'.
5. $G' = (V', E')$ is the final graph.

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique_k and Independent-Set_k require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

- Clearly, G has a 3-coloring if and only if G' has a k-clique.

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

- Clearly, G has a 3-coloring if and only if G' has a k-clique.
- Now assume Clique$_k$ is solvable in $f(k) \cdot n^{o(k)}$ time. Then there exists a monotonously increasing unbounded function s such that Clique$_k$ is solvable in $f(k) \cdot n^{k/s(k)}$ time.

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique_k and Independent-Set_k require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

- Clearly, G has a 3-coloring if and only if G' has a k-clique.
- Now assume Clique_k is solvable in $f(k) \cdot n^{o(k)}$ time. Then there exists a monotonously increasing unbounded function s such that Clique_k is solvable in $f(k) \cdot n^{k/s(k)}$ time.
- Set k to be the maximum value such that $f(k) \leq n$ and $k^{k/s(k)} \leq n$, and let g be the minimum value between the inverse of the two functions above.

Lower Bounds Under ETH

THEOREM

Assuming ETH, for any computable function f, Clique$_k$ and Independent-Set$_k$ require $f(k) \cdot n^{\Omega(k)}$ time.

PROOF)

- Clearly, G has a 3-coloring if and only if G' has a k-clique.
- Now assume Clique$_k$ is solvable in $f(k) \cdot n^{o(k)}$ time. Then there exists a monotonously increasing unbounded function s such that Clique$_k$ is solvable in $f(k) \cdot n^{k/s(k)}$ time.
- Set k to be the maximum value such that $f(k) \leq n$ and $k^{k/s(k)} \leq n$, and let g be the minimum value between the inverse of the two functions above.
- Now the runtime of our 3-coloring algorithm is
  $f(k) \cdot ((k \cdot 3^{n/k})^{k/s(k)}) \leq n \cdot k^{k/s(k)} \cdot 3^{n/s(k)} \leq n^2 \cdot 3^{n/s(g(n))} \leq 2^{o(n)}$
  which violates the ETH.

∎

The End