Approximating Cumulative Pebbling Cost is Unique Games Hard

Jeremiah Blocki[1], Seunghoon Lee[1], Samson Zhou[2]

[1]Department of Computer Science, Purdue University
[2]School of Computer Science, Carnegie Mellon University

November 6, 2019

Contents

(Parallel) Graph Pebbling and Cumulative Pebbling Cost (cc(G))

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G).

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

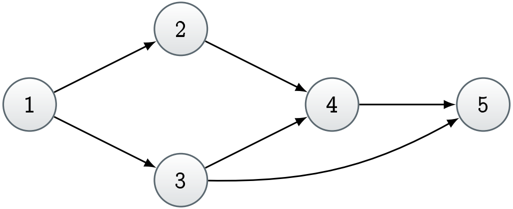
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



(Parallel) Graph Pebbling and Cumulative Pebbling Cost (cc(G))

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G).

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

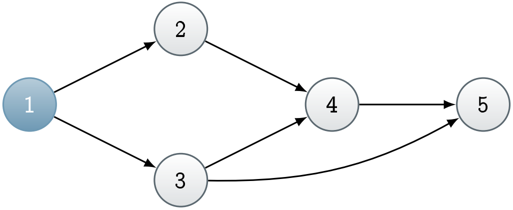
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



$$P_1 = \{1\}$$

(Parallel) Graph Pebbling and Cumulative Pebbling Cost (cc(G))

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G)**.**

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

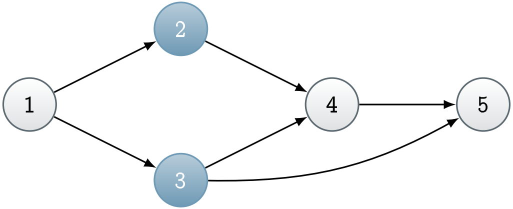
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



$$P_1 = \{1\}, P_2 = \{2, 3\}$$

(Parallel) Graph Pebbling and Cumulative Pebbling Cost (cc(G))

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G).

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

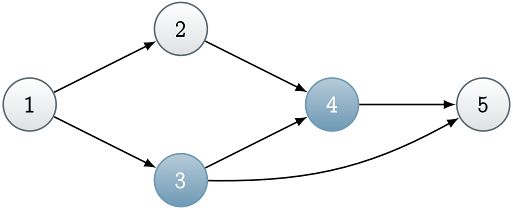
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



$$P_1 = \{1\}, P_2 = \{2,3\}, P_3 = \{3,4\}$$

(Parallel) Graph Pebbling and Cumulative Pebbling Cost (cc(G))

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G)**.**

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

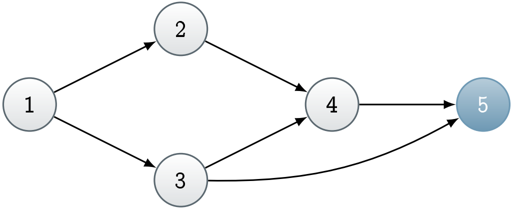
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



$$P_1 = \{1\}, P_2 = \{2,3\}, P_3 = \{3,4\}, P_4 = \{5\}$$

(Parallel) Graph Pebbling and Cumulative Pebbling Cost (cc(G))

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G).

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

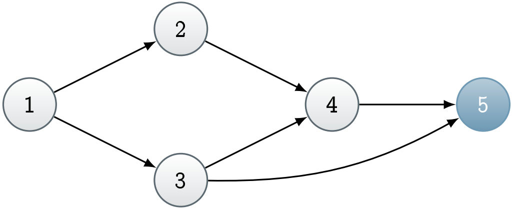
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



$P_1 = \{1\}, P_2 = \{2,3\}, P_3 = \{3,4\}, P_4 = \{5\}$

$$\therefore \underbrace{\mathrm{cc}(G)}_{\text{take minimum}} \leq \sum_{i=1}^{t} |P_i| = 1$$

(Parallel) Graph Pebbling and Cumulative Pebbling Cost ($\mathsf{cc}(G)$)

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of $\mathsf{cc}(G)$.

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate $\mathsf{cc}(G)$ for any constant factor

Technical Ingredients.

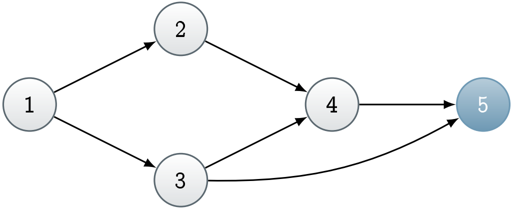
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



$$P_1 = \{1\}, P_2 = \{2, 3\}, P_3 = \{3, 4\}, P_4 = \{5\}$$

$$\therefore \underbrace{\mathsf{cc}(G)}_{\text{take minimum}} \leq \sum_{i=1}^{t} |P_i| = 1 + 2$$

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G)**.**

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

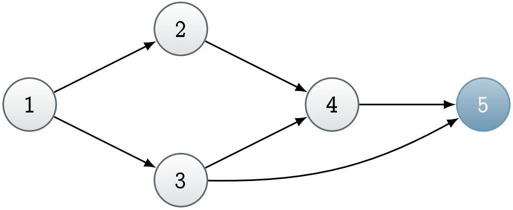
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



$$P_1 = \{1\}, P_2 = \{2,3\}, P_3 = \{3,4\}, P_4 = \{5\}$$

$$\therefore \underbrace{\mathrm{cc}(G)}_{\text{take minimum}} \le \sum_{i=1}^{t} |P_i| = 1 + 2 + 2$$

(Parallel) Graph Pebbling and Cumulative Pebbling Cost (cc(G))

Overview

We Are Here

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G).

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

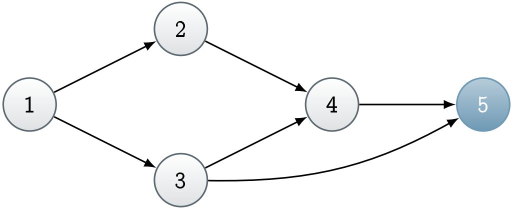
- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Goal. Place pebbles on all sink nodes.

Pebbling Rules. (informal)

- Initially, the graph is unpebbled and start with the root nodes.
- We can add a new pebble only if its parents were all pebbled.
- (Parallel) We can place multiple pebbles at the same time.
- We can discard pebbles at any time if not needed.

(Parallel) Pebbling Example.



$$P_1 = \{1\}, P_2 = \{2,3\}, P_3 = \{3,4\}, P_4 = \{5\}$$

$$\therefore \underbrace{\mathrm{cc}(G)}_{\text{take minimum}} \leq \sum_{i=1}^{t} |P_i| = 1 + 2 + 2 + 1 = 6.$$

Significance of cc(G) and a Challenging Problem

Overview

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

We Are Here

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G).

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Challenging Problem.

- Given a DAG G, find the (approximately) minimum cost pebbling

Why We Care About cc(G)**?**

- Analysis of data-independent Memory-Hard Functions (iMHFs)

Theorem [AS15] (informal)

For a secure memory hard function for password hashing, it suffices to find a DAG G with *constant indegree* and *maximum* cc(G).

- Amortization / Parallelism ($\mathrm{cc}(G^{\times n}) = n \times \mathrm{cc}(G)$)

Challenges.

- We don't know how to compute cc(G) exactly for any given G
- Large gaps between upper/lower bounds for known constructions

Example

$$\frac{10^{-6} \cdot N^2}{\log N} \le \mathrm{cc}(\mathrm{DRSample}) \le \frac{1 \cdot N^2}{\log N}.$$

Our Main Result: Hardness of Approximating cc(G)

Overview

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G).

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

We Are Here

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Our Result.

- [BZ18] proved that computing cc(G) is NP-Hard
- This did not rule out the existence of a constant-factor approximation algorithm for cc(G)
- Our result is the hardness of any constant factor approximation to the cost of graph pebbling even for DAGs with constant indegree.

Theorem

Given a DAG G with constant indegree, it is Unique Games hard to approximate cc(G) within any constant factor.

Our Main Result: Hardness of Approximating cc(G)

Overview

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G)**.**

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

We Are Here

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

Technical Ingredients.

- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

Our Result.

- [BZ18] proved that computing cc(G) is NP-Hard
- This did not rule out the existence of a constant-factor approximation algorithm for cc(G)
- Our result is the hardness of any constant factor approximation to the cost of graph pebbling even for DAGs with constant indegree.

Theorem

Given a DAG G with constant indegree, it is Unique Games hard to approximate cc(G) *within any constant factor.*

Implication.

- Cryptanalysis of iMHFs is Hard!

Technical Ingredients

Overview

(Parallel) Graph Pebbling.

- Pebbling example
- Cumulative Pebbling Cost of G

Problem Statement.

- Given a DAG G find the (approx.) minimum cost pebbling

Significance of cc(G).

- Analysis of data-independent memory-hard functions
- Amortization / Parallelism

Results.

- Unique Games Hard to approximate cc(G) for any constant factor

We Are Here

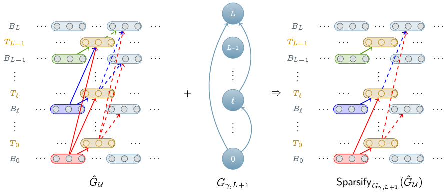
Technical Ingredients.

- Indegree reduction using γ-extreme depth robust graphs
- Superconcentrator overlay

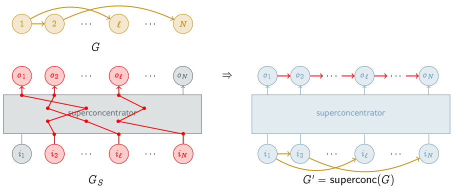
Svensson's Result [Sve12].

- cc(G) is related to the combinatorial property called Depth-Robustness
- Unique Games Hard to approximately test DAGs for Depth-Robustness
  - Challenge 1: Svensson's reduction dœsn't work for constant indegree graphs
  - Challenge 2: Connection between Depth-Robustness and cc(G) is not tight

Indegree Reduction Procedure using γ-Extreme DR Graph $G_{\gamma,L+1}$.



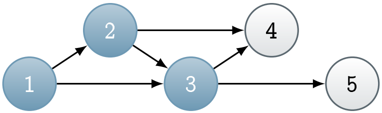
Superconcentrator Overlay.



We are now at...

Graph Pebbling (Sequential/Parallel)

Consider a directed acyclic graph (DAG) $G = (V, E)$.



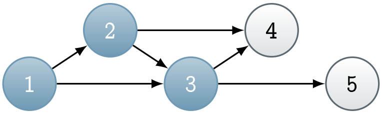
Goal: place pebbles on all sink nodes.

Pebbling Rules: $P = \{P_1, \cdots, P_t\} \subset V$ where $P_i \subseteq V$ denotes the set of pebbles in round i,

- $P_0 = \varnothing$, (initially, the graph is unpebbled)
- $\forall i \in [t], \quad v \in P_i \setminus P_{i-1} \implies \text{parents}(v) \subseteq P_{i-1}$, and
  (a new pebble can be added only if its parents were all pebbled in the previous round)
- $\forall i \in [t], \quad |P_i \setminus P_{i-1}| \leq 1$. (only in the sequential pebbling game)
- We will focus on the *parallel pebbling game* throughout this talk.

Graph Pebbling (Sequential/Parallel)

Consider a directed acyclic graph (DAG) $G = (V, E)$.

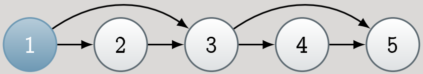


Goal: place pebbles on all sink nodes.

Pebbling Rules: $P = \{P_1, \cdots, P_t\} \subset V$ where $P_i \subseteq V$ denotes the set of pebbles in round i,

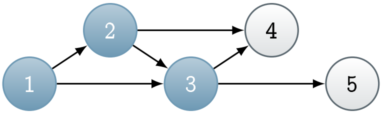
- $P_0 = \varnothing$, (initially, the graph is unpebbled)
- $\forall i \in [t], \quad v \in P_i \setminus P_{i-1} \Rightarrow$ parents$(v) \subseteq P_{i-1}$, and (a new pebble can be added only if its parents were all pebbled in the previous round)
- $\forall i \in [t], \quad |P_i \setminus P_{i-1}| \leq 1$. (only in the sequential pebbling game)
- We will focus on the *parallel pebbling game* throughout this talk.

Example



$P_1 = \{1\}$ (data value L_1 stored in memory)

Consider a directed acyclic graph (DAG) $G = (V, E)$.

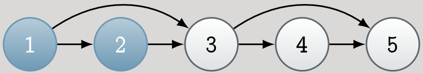


Goal: place pebbles on all sink nodes.

Pebbling Rules: $P = \{P_1, \cdots, P_t\} \subset V$ where $P_i \subseteq V$ denotes the set of pebbles in round i,

- $P_0 = \varnothing$, (initially, the graph is unpebbled)
- $\forall i \in [t], \quad v \in P_i \setminus P_{i-1} \Rightarrow$ parents$(v) \subseteq P_{i-1}$, and (a new pebble can be added only if its parents were all pebbled in the previous round)
- $\forall i \in [t], \quad |P_i \setminus P_{i-1}| \leq 1$. (only in the sequential pebbling game)
- We will focus on the *parallel pebbling game* throughout this talk.

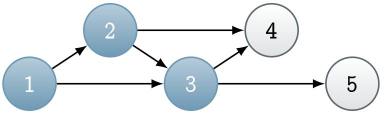
Example



$P_2 = \{1, 2\}$ (data values L_1 and L_2 stored in memory)

Graph Pebbling (Sequential/Parallel)

Consider a directed acyclic graph (DAG) $G = (V, E)$.

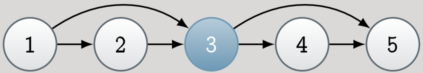


Goal: place pebbles on all sink nodes.

Pebbling Rules: $P = \{P_1, \cdots, P_t\} \subset V$ where $P_i \subseteq V$ denotes the set of pebbles in round i,

- $P_0 = \varnothing$, (initially, the graph is unpebbled)
- $\forall i \in [t], \quad v \in P_i \setminus P_{i-1} \ \Rightarrow$ parents$(v) \subseteq P_{i-1}$, and
  (a new pebble can be added only if its parents were all pebbled in the previous round)
- $\forall i \in [t], \quad |P_i \setminus P_{i-1}| \leq 1$. (only in the sequential pebbling game)
- We will focus on the *parallel pebbling game* throughout this talk.

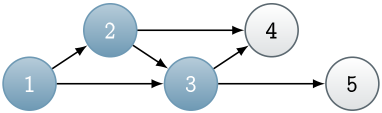
Example



$P_3 = \{3\}$ (data value L_3 stored in memory)

Graph Pebbling (Sequential/Parallel)

Consider a directed acyclic graph (DAG) $G = (V, E)$.

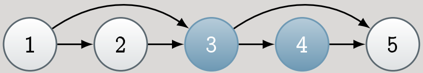


Goal: place pebbles on all sink nodes.

Pebbling Rules: $P = \{P_1, \cdots, P_t\} \subset V$ where $P_i \subseteq V$ denotes the set of pebbles in round i,

- $P_0 = \varnothing$, (initially, the graph is unpebbled)
- $\forall i \in [t], \quad v \in P_i \setminus P_{i-1} \Rightarrow$ parents$(v) \subseteq P_{i-1}$, and (a new pebble can be added only if its parents were all pebbled in the previous round)
- $\forall i \in [t], \quad |P_i \setminus P_{i-1}| \leq 1$. (only in the sequential pebbling game)
- We will focus on the *parallel pebbling game* throughout this talk.

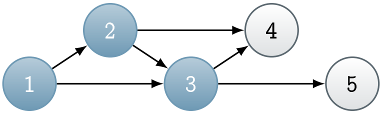
Example



$P_4 = \{3, 4\}$ (data values L_3 and L_4 stored in memory)

Graph Pebbling (Sequential/Parallel)

Consider a directed acyclic graph (DAG) $G = (V, E)$.

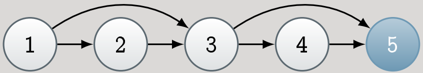


Goal: place pebbles on all sink nodes.

Pebbling Rules: $P = \{P_1, \cdots, P_t\} \subset V$ where $P_i \subseteq V$ denotes the set of pebbles in round i,

- $P_0 = \varnothing$, (initially, the graph is unpebbled)
- $\forall i \in [t], \quad v \in P_i \setminus P_{i-1} \Rightarrow$ parents$(v) \subseteq P_{i-1}$, and
  (a new pebble can be added only if its parents were all pebbled in the previous round)
- $\forall i \in [t], \quad |P_i \setminus P_{i-1}| \leq 1$. (only in the sequential pebbling game)
- We will focus on the *parallel pebbling game* throughout this talk.

Example



$P_5 = \{5\}$ (data value L_5 stored in memory)

Pebbling Complexity: The Cumulative Pebbling Cost $\mathsf{cc}(G)$

Let $\mathcal{P}_G^{\parallel}$ be the set of all valid *parallel* pebblings of G.

Definition

- The *cumulative cost of a pebbling* $P = (P_1, \cdots, P_t) \in \mathcal{P}_G^{\parallel}$ is

$$\mathsf{cc}(P) := |P_1| + \cdots + |P_t|.$$

- The *cumulative pebbling cost of a graph* G is defined by

$$\mathsf{cc}(G) = \min_{P \in \mathcal{P}_G^{\parallel}} \mathsf{cc}(P)$$

where the minimum is taken over all legal black pebblings of G.

Pebbling Complexity: The Cumulative Pebbling Cost $\mathsf{cc}(G)$

Let $\mathcal{P}_G^{\parallel}$ be the set of all valid *parallel* pebblings of G.

Definition

- The *cumulative cost of a pebbling* $P = (P_1, \cdots, P_t) \in \mathcal{P}_G^{\parallel}$ is

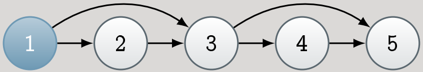
$$\mathsf{cc}(P) := |P_1| + \cdots + |P_t|.$$

- The *cumulative pebbling cost of a graph* G is defined by

$$\mathsf{cc}(G) = \min_{P \in \mathcal{P}_G^{\parallel}} \mathsf{cc}(P)$$

where the minimum is taken over all legal black pebblings of G.

Example



$$\mathsf{cc}(G) \le |P_1| + \cdots + |P_5| = 1 + 2 + 1 + 2 + 1 = 7.$$

Let $\mathcal{P}_G^{||}$ be the set of all valid *parallel* pebblings of G.

Definition

- The *cumulative cost of a pebbling* $P = (P_1, \cdots, P_t) \in \mathcal{P}_G^{||}$ is

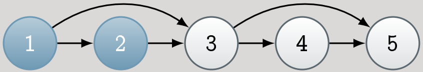
$$\mathrm{cc}(P) := |P_1| + \cdots + |P_t|.$$

- The *cumulative pebbling cost of a graph* G is defined by

$$\mathrm{cc}(G) = \min_{P \in \mathcal{P}_G^{||}} \mathrm{cc}(P)$$

where the minimum is taken over all legal black pebblings of G.

Example



$$\mathrm{cc}(G) \le |P_1| + \cdots + |P_5| = 1 + 2 + 1 + 2 + 1 = 7.$$

Pebbling Complexity: The Cumulative Pebbling Cost $\mathsf{cc}(G)$

Let $\mathcal{P}_G^{\parallel}$ be the set of all valid *parallel* pebblings of G.

Definition

- The *cumulative cost of a pebbling* $P = (P_1, \cdots, P_t) \in \mathcal{P}_G^{\parallel}$ is

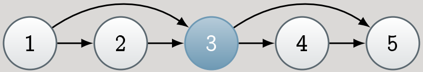
$$\mathsf{cc}(P) := |P_1| + \cdots + |P_t|.$$

- The *cumulative pebbling cost of a graph* G is defined by

$$\mathsf{cc}(G) = \min_{P \in \mathcal{P}_G^{\parallel}} \mathsf{cc}(P)$$

where the minimum is taken over all legal black pebblings of G.

Example



$$\mathsf{cc}(G) \le |P_1| + \cdots + |P_5| = 1 + 2 + 1 + 2 + 1 = 7.$$

Pebbling Complexity: The Cumulative Pebbling Cost $\mathsf{cc}(G)$

Let $\mathcal{P}_G^{||}$ be the set of all valid *parallel* pebblings of G.

Definition

- The *cumulative cost of a pebbling* $P = (P_1, \cdots, P_t) \in \mathcal{P}_G^{||}$ is

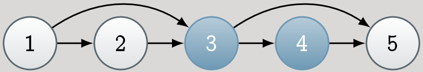
$$\mathsf{cc}(P) := |P_1| + \cdots + |P_t|.$$

- The *cumulative pebbling cost of a graph* G is defined by

$$\mathsf{cc}(G) = \min_{P \in \mathcal{P}_G^{||}} \mathsf{cc}(P)$$

where the minimum is taken over all legal black pebblings of G.

Example



$\mathsf{cc}(G) \le |P_1| + \cdots + |P_5| = 1 + 2 + 1 + 2 + 1 = 7.$

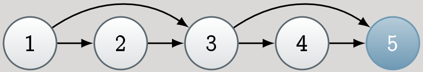
Pebbling Complexity: The Cumulative Pebbling Cost $\mathsf{cc}(G)$

Let $\mathcal{P}_G^{||}$ be the set of all valid *parallel* pebblings of G.

Definition

- The *cumulative cost of a pebbling* $P = (P_1, \cdots, P_t) \in \mathcal{P}_G^{||}$ is
$$\mathsf{cc}(P) := |P_1| + \cdots + |P_t|.$$
- The *cumulative pebbling cost of a graph* G is defined by
$$\mathsf{cc}(G) = \min_{P \in \mathcal{P}_G^{||}} \mathsf{cc}(P)$$
where the minimum is taken over all legal black pebblings of G.

Example



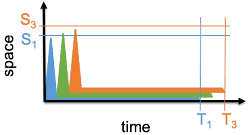
$$\mathsf{cc}(G) \le |P_1| + \cdots + |P_5| = 1 + 2 + 1 + 2 + 1 = 7.$$

Data-Independent Memory Hard Function (iMHF).

- **Intuition:** computation costs dominated by memory costs
- **Goal:** force attacker to lock up large amounts of memory for duration of computation

Amortization and Parallelism.

- Consider the Space×Time (ST)-Complexity $\mathsf{ST}(G) := \min_{P \in \mathcal{P}_G^{\parallel}} \left(t_P \times \max_{i \le t_P} |P_i|\right)$
- For parallel computation ST-complexity can scale badly in the number of evaluations of a function



$$ST_1 = S_1 \times T_1 \approx S_3 \times T_3 = ST_3$$

cost of computing f once

cost of computing f three times

- Cumulative pebbling cost scales well ($\mathsf{cc}(G^{\times n}) = n \times \mathsf{cc}(G)$)

Theorem [AS15] (informal)

For a secure memory hard function for password hashing, it suffices to find a DAG G with *constant indegree* and *maximum* $\mathsf{cc}(G)$.

We are now at...

The Main Result: Regarding the Hardness of Computing $\mathsf{cc}(G)$

- Blocki and Zhou [BZ18] recently showed that computing $\mathsf{cc}(G)$ is NP-Hard. However, this dœs not rule out the existence of a $(1+\varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$.
- Our main result is the hardness of any constant factor approximation to the cost of graph pebbling even for DAGs with constant indegree.

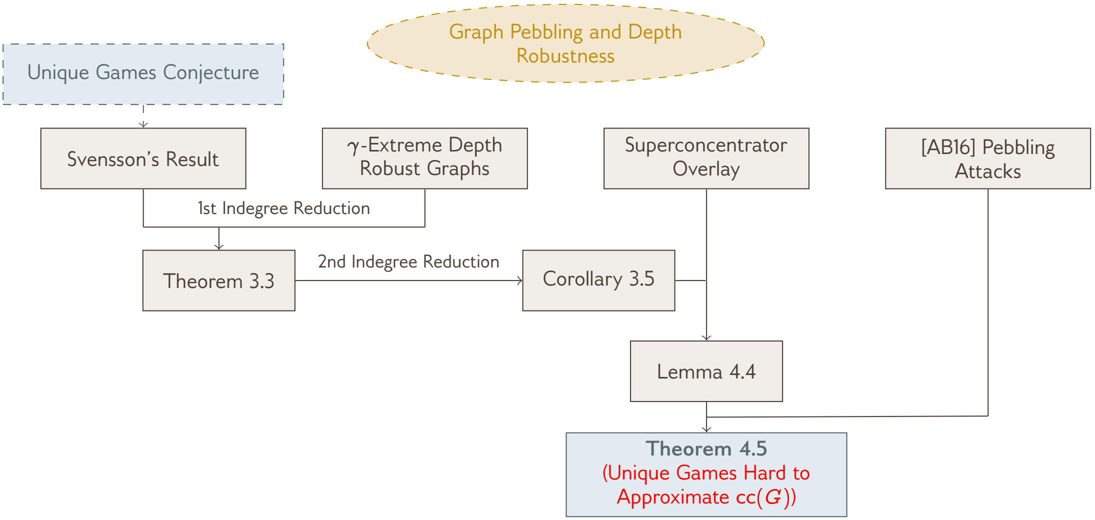
Theorem

Given a DAG G with constant indegree, it is Unique Games hard to approximate $\mathsf{cc}(G)$ within any constant factor.

Strategy?

- Svensson's result of Unique Games hardness to distinguish two cases for a DAG G
- Reduction to $\widetilde{G}$ with *gap* between the upper and lower bound of $\mathsf{cc}(\widetilde{G})$

Proof Overview



We are now at...

Definition (Unique Games)

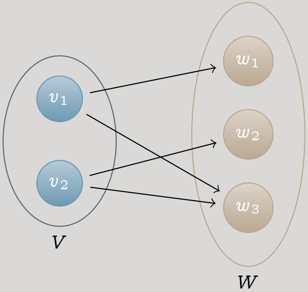
An instance of Unique Games $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ consists of a regular bipartite graph $G(V, W, E)$ and a set $[R]$ of labels. Each edge $(v, w) \in E$ has a constraint given by a permutation $\pi_{v,w} : [R] \to [R]$. The goal is to output a labeling $\rho : (V \cup W) \to [R]$ that maximizes the number of satisfied edges, where an edge is satisfied if $\rho(v) = \pi_{v,w}(\rho(w))$.

Unique Games Conjecture

Definition (Unique Games)

An instance of Unique Games $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ consists of a regular bipartite graph $G(V, W, E)$ and a set $[R]$ of labels. Each edge $(v, w) \in E$ has a constraint given by a permutation $\pi_{v,w} : [R] \to [R]$. The goal is to output a labeling $\rho : (V \cup W) \to [R]$ that maximizes the number of satisfied edges, where an edge is satisfied if $\rho(v) = \pi_{v,w}(\rho(w))$.

Example



Consider the following permutation assignment:

$$\pi_{v_1,w_1} : \{1,2,3,4,5\} \to \{2,5,1,3,4\}, \text{ (e.g. } \pi_{v_1,w_1}(1) = 2)$$
$$\pi_{v_1,w_3} : \{1,2,3,4,5\} \to \{3,2,5,4,1\},$$
$$\pi_{v_2,w_2} : \{1,2,3,4,5\} \to \{4,3,2,5,1\},$$
$$\pi_{v_2,w_3} : \{1,2,3,4,5\} \to \{3,1,4,5,2\}.$$

$\rho(v_1)$	$\rho(v_2)$	$\rho(w_1)$	$\rho(w_2)$	$\rho(w_3)$	(#satisfied edges)
1	2	3	4	5	3
2	3	5	1	4	0
3	4	2	5	1	1
⋮	⋮	⋮	⋮	⋮	⋮

Unique Games Conjecture

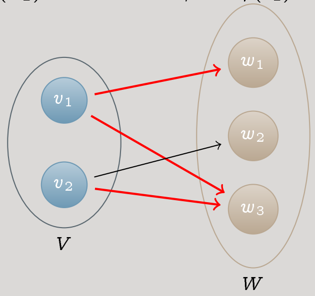
Definition (Unique Games)

An instance of Unique Games $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ consists of a regular bipartite graph $G(V, W, E)$ and a set $[R]$ of labels. Each edge $(v, w) \in E$ has a constraint given by a permutation $\pi_{v,w} : [R] \to [R]$. The goal is to output a labeling $\rho : (V \cup W) \to [R]$ that maximizes the number of satisfied edges, where an edge is satisfied if $\rho(v) = \pi_{v,w}(\rho(w))$.

Example

$\rho(w_1) = 3 \xrightarrow{\pi_{v_1,w_1}} 1 = \rho(v_1)$

$\rho(w_2) = 4 \xrightarrow{\pi_{v_2,w_2}} 5 \neq 2 = \rho(v_2)$



Consider the following permutation assignment:

$$\pi_{v_1,w_1} : \{1,2,3,4,5\} \to \{2,5,1,3,4\}, \text{ (e.g. } \pi_{v_1,w_1}(1) = 2)$$
$$\pi_{v_1,w_3} : \{1,2,3,4,5\} \to \{3,2,5,4,1\},$$
$$\pi_{v_2,w_2} : \{1,2,3,4,5\} \to \{4,3,2,5,1\},$$
$$\pi_{v_2,w_3} : \{1,2,3,4,5\} \to \{3,1,4,5,2\}.$$

$\rho(v_1)$	$\rho(v_2)$	$\rho(w_1)$	$\rho(w_2)$	$\rho(w_3)$	(#satisfied edges)
1	2	3	4	5	3
2	3	5	1	4	0
3	4	2	5	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

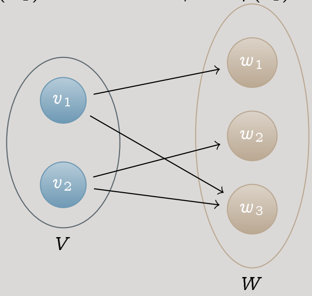
Unique Games Conjecture

Definition (Unique Games)

An instance of Unique Games $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ consists of a regular bipartite graph $G(V, W, E)$ and a set $[R]$ of labels. Each edge $(v, w) \in E$ has a constraint given by a permutation $\pi_{v,w} : [R] \to [R]$. The goal is to output a labeling $\rho : (V \cup W) \to [R]$ that maximizes the number of satisfied edges, where an edge is satisfied if $\rho(v) = \pi_{v,w}(\rho(w))$.

Example

$\rho(w_1) = 5 \xrightarrow{\pi_{v_1,w_1}} 4 \neq 2 = \rho(v_1)$



Consider the following permutation assignment:

$$\pi_{v_1,w_1} : \{1, 2, 3, 4, 5\} \to \{2, 5, 1, 3, 4\}, \text{ (e.g. } \pi_{v_1,w_1}(1) = 2)$$
$$\pi_{v_1,w_3} : \{1, 2, 3, 4, 5\} \to \{3, 2, 5, 4, 1\},$$
$$\pi_{v_2,w_2} : \{1, 2, 3, 4, 5\} \to \{4, 3, 2, 5, 1\},$$
$$\pi_{v_2,w_3} : \{1, 2, 3, 4, 5\} \to \{3, 1, 4, 5, 2\}.$$

$\rho(v_1)$	$\rho(v_2)$	$\rho(w_1)$	$\rho(w_2)$	$\rho(w_3)$	(#satisfied edges)
1	2	3	4	5	3
2	3	5	1	4	0
3	4	2	5	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Unique Games Conjecture

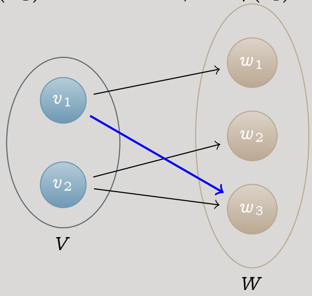
Definition (Unique Games)

An instance of Unique Games $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ consists of a regular bipartite graph $G(V, W, E)$ and a set $[R]$ of labels. Each edge $(v, w) \in E$ has a constraint given by a permutation $\pi_{v,w} : [R] \to [R]$. The goal is to output a labeling $\rho : (V \cup W) \to [R]$ that maximizes the number of satisfied edges, where an edge is satisfied if $\rho(v) = \pi_{v,w}(\rho(w))$.

Example

$\rho(w_3) = 1 \xrightarrow{\pi_{v_1,w_3}} 3 = \rho(v_1)$

$\rho(w_1) = 2 \xrightarrow{\pi_{v_1,w_1}} 5 \neq 3 = \rho(v_1)$



Consider the following permutation assignment:

$$\pi_{v_1,w_1} : \{1,2,3,4,5\} \to \{2,5,1,3,4\}, \text{ (e.g. } \pi_{v_1,w_1}(1) = 2)$$
$$\pi_{v_1,w_3} : \{1,2,3,4,5\} \to \{3,2,5,4,1\},$$
$$\pi_{v_2,w_2} : \{1,2,3,4,5\} \to \{4,3,2,5,1\},$$
$$\pi_{v_2,w_3} : \{1,2,3,4,5\} \to \{3,1,4,5,2\}.$$

$\rho(v_1)$	$\rho(v_2)$	$\rho(w_1)$	$\rho(w_2)$	$\rho(w_3)$	(#satisfied edges)
1	2	3	4	5	3
2	3	5	1	4	0
3	4	2	5	1	1
⋮	⋮	⋮	⋮	⋮	⋮

Unique Games Conjecture

Definition (Unique Games)

An instance of Unique Games $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ consists of a regular bipartite graph $G(V, W, E)$ and a set $[R]$ of labels. Each edge $(v, w) \in E$ has a constraint given by a permutation $\pi_{v,w} : [R] \to [R]$. The goal is to output a labeling $\rho : (V \cup W) \to [R]$ that maximizes the number of satisfied edges, where an edge is satisfied if $\rho(v) = \pi_{v,w}(\rho(w))$.

The following conjecture from [Kho02] has been extensively used to prove several strong hardness of approximation algorithm.

Conjecture (Unique Games Conjecture) [Kho02]

For any constants $\alpha, \beta > 0$, there exists a sufficiently large integer R (as a function of α, β) such that for Unique Games instance with label set $[R]$, no polynomial time algorithm can distinguish whether:

1. (completeness) the maximum fraction of satisfied edges of any labeling is at least $1 - \alpha$, or
2. (soundness) the maximum fraction of satisfied edges of any labeling is less than β.

- Approximation algorithm for $\mathrm{cc}(G)$?

We are now at...

Depth Robustness ($\leftrightarrow$ Depth Reducibility)

First, we define depth(G) to be the length of the longest directed path in a DAG G.

Definition

- A DAG $G = (V, E)$ is (e, d)-*depth robust* if

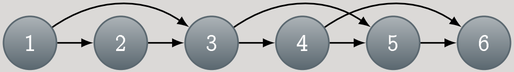
$$\forall S \subseteq V \text{ s.t. } |S| \le e \quad \Rightarrow \quad \text{depth}(G - S) \ge d.$$

- We say that G is (e, d)-*reducible* if G is not (e, d)-depth robust. That is,

$$\exists S \subseteq V \text{ s.t. } |S| \le e \text{ and depth}(G - S) < d.$$

Example

The following graph is $(e = 2, d = 2)$-reducible:



Depth Robustness ($\leftrightarrow$ Depth Reducibility)

First, we define depth(G) to be the length of the longest directed path in a DAG G.

Definition

- A DAG $G = (V, E)$ is (e, d)-*depth robust* if

$$\forall S \subseteq V \text{ s.t. } |S| \leq e \quad \Rightarrow \quad \text{depth}(G - S) \geq d.$$

- We say that G is (e, d)-*reducible* if G is not (e, d)-depth robust. That is,

$$\exists S \subseteq V \text{ s.t. } |S| \leq e \text{ and depth}(G - S) < d.$$

Example

The following graph is $(e = 2, d = 2)$-reducible:

Depth Robustness ($\leftrightarrow$ Depth Reducibility)

First, we define depth(G) to be the length of the longest directed path in a DAG G.

Definition

- A DAG $G = (V, E)$ is (e, d)-*depth robust* if

$$\forall S \subseteq V \text{ s.t. } |S| \leq e \quad \Rightarrow \quad \text{depth}(G - S) \geq d.$$

- We say that G is (e, d)-*reducible* if G is not (e, d)-depth robust. That is,

$$\exists S \subseteq V \text{ s.t. } |S| \leq e \text{ and } \text{depth}(G - S) < d.$$

A few facts about depth robustness:

- [AB16] For any (e, d)-reducible DAG G with N nodes,

$$\text{cc}(G) \leq \min_{g \geq d} \left(eN + gN \times \text{indeg}(G) + \frac{N^2 d}{g} \right).$$

- [ABP17] For any (e, d)-depth robust DAG G,

$$\text{cc}(G) \geq ed.$$

We are now at...

Technical Ingredients 1: Svensson's Result of Unique Games Hardness

Svensson [Sve12] proved the Unique Games hardness of a DAG G:

Theorem [Sve12]

For any constant $k, \varepsilon > 0$, it is Unique Games hard to distinguish between whether

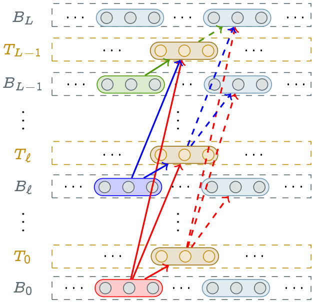
1. G is (e_1, d_1)-reducible with $e_1 = N/k$ and $d_1 = k$, and
2. G is (e_2, d_2)-depth robust with $e_2 = N(1 - 1/k)$ and $d_2 = \Omega(N^{1-\varepsilon})$.

- To prove this, reduction from an instance of Unique Games $\mathcal{U} = (G = (V, W, E), [R], \{\pi_{v,w}\}_{v,w})$ to a DAG $G_{\mathcal{U}}$ on N nodes.
  - G is (e_1, d_1)-reducible if $\mathcal{U}$ is satisfiable, and
  - G is (e_2, d_2)-depth robust if $\mathcal{U}$ is unsatisfiable.
- As mentioned before, we have nice upper and lower bounds for $\mathrm{cc}(G)$ from [ABP17] and [AB16]:

Theorem

- [ABP17] For any (e, d)-depth robust DAG G, we have $\mathrm{cc}(G) \geq ed$.
- [AB16] For any (e, d)-reducible DAG G with N nodes, we have $\mathrm{cc}(G) \leq \min_{g \geq d} \left(eN + gN \times \mathrm{indeg}(G) + \frac{N^2 d}{g} \right)$.

Svensson's Construction



1. The graph $\hat{G}_{\mathcal{U}}$ contains two types of vertices:
   - bit-vertices partitioned into bit-layers $B = B_0 \cup \cdots \cup B_L$,
   - test-vertices partitioned into test-layers $T = T_0 \cup \cdots \cup T_{L-1}$, and
   - all of the edges in the graph are between bit-vertices and test-vertices.
2. $\hat{G}_{\mathcal{U}}$ shows symmetry between the layers:
   - $B_\ell = \{b_1^\ell, \cdots, b_m^\ell\}$ and $T_\ell = \{t_1^\ell, \cdots, t_p^\ell\}$ (# of bit- and test-vertices in each layer is the same)
   - The edges between B_ℓ and T_ℓ (resp. T_ℓ and $B_{\ell+1}$) encode the edge constraints in the UG instance $\mathcal{U}$.
   - The directed edge (b_i^ℓ, t_j^ℓ) exists $\Leftrightarrow \forall \ell' \geq \ell$ the edge $(b_i^\ell, t_j^{\ell'})$ exists.
   - The directed edge $(t_j^\ell, b_i^{\ell+1})$ exists $\Leftrightarrow \forall \ell' > \ell$ the edge $(t_j^\ell, b_i^{\ell'})$ exists.
3. The number of layers $L = N^{1-\varepsilon}$.

$\Rightarrow$ indeg$(\hat{G}_{\mathcal{U}}) \geq L$ (and can be as large as $\Omega(N)$ in general.)

Theorem [Sve12]

For any integer $k \geq 2$ and constant $\varepsilon > 0$, it is Unique Games hard to distinguish between whether

1. G is (e_1, d_1)-reducible with $e_1 = N/k$ and $d_1 = k$, and
2. G is (e_2, d_2)-depth robust with $e_2 = N(1 - 1/k)$ and $d_2 = \Omega(N^{1-\varepsilon})$.

Challenges of Applying Svensson's Construction

The result of Alwen et al. [ABP17] and [AB16] tells us that

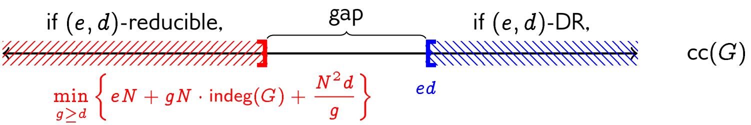
- $\mathsf{cc}(G_{\mathcal{U}}) \geq e_2 d_2$, and
- $\mathsf{cc}(G_{\mathcal{U}}) \leq \min_{g \geq d_1} \left(e_1 N + gN \times \mathsf{indeg}(G_{\mathcal{U}}) + \frac{N^2 d_1}{g} \right)$

$\Rightarrow$ no gap between the upper/lower bounds since $\mathsf{indeg}(G_{\mathcal{U}}) = \mathcal{O}(N)$ implies

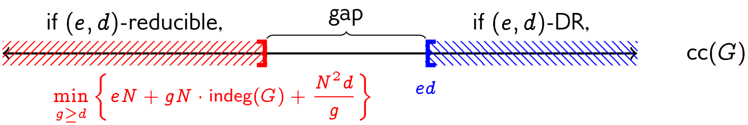
$$gN \times \mathsf{indeg}(G_{\mathcal{U}}) = \mathcal{O}(gN^2) \gg \Omega(N^{2-\varepsilon}) = e_2 d_2.$$

$\Rightarrow$ need to reduce the indegree (how? using γ-extreme depth-robust graphs.)

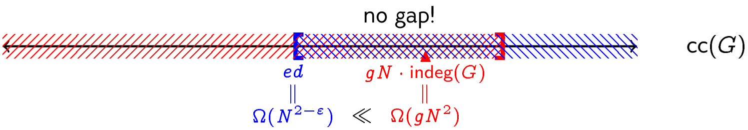
What we want:


if (e, d)-reducible,
gap
if (e, d)-DR,
$\mathsf{cc}(G)$
$\min_{g \geq d} \left\{ eN + gN \cdot \mathsf{indeg}(G) + \frac{N^2 d}{g} \right\}$
ed

What we want:



When applying Svensson's Theorem directly:



What do we do? Reduce $\mathsf{indeg}(G)$!

We are now at...

Technical Ingredients 2: γ-Extreme Depth Robust Graphs (Indegree Reduction)

- As discussed before, Svensson's construction has too large indegree ($\mathcal{O}(N)$) for the purposes of bounding $\mathrm{cc}(G)$. How to reduce indegree?

Definition

A DAG $G_{\gamma,N}$ on N nodes is said to be γ-*extreme depth-robust* if it is (e,d)-depth robust for any $e, d > 0$ such that $e + d \leq (1-\gamma)N$.

Svensson's Graph $\hat{G}_{\mathcal{U}}$

γ-Extreme DR Graph $G_{\gamma,L+1}$

transformation Sparsify

- keep the edge $(b^{\ell}, t^{\ell'}) \Leftrightarrow$ $\ell = \ell'$ or $(\ell, \ell') \in E(G_{\gamma,L+1})$
- keep the edge $(t^{\ell'}, b^{\ell}) \Leftrightarrow$ $(\ell', \ell) \in E(G_{\gamma,L+1})$

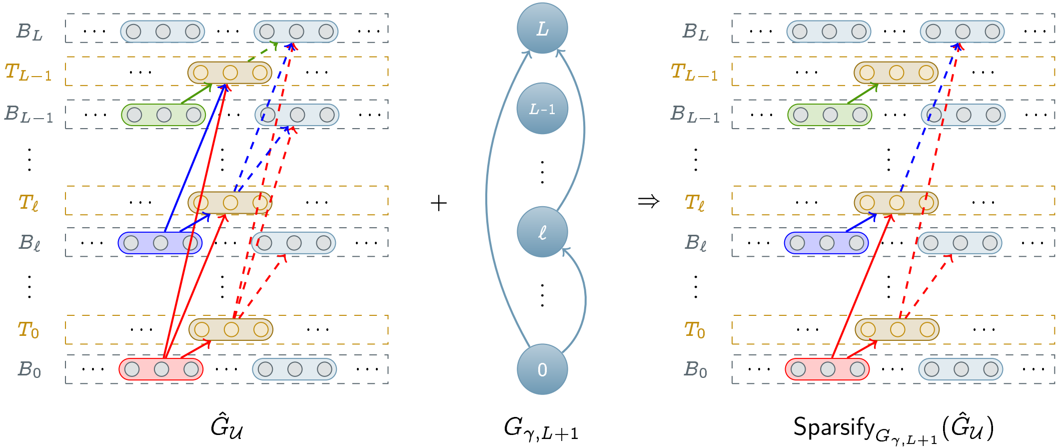
$\mathrm{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$

- Indegree and outdegree $\mathcal{O}(N^{\varepsilon} \log^2 N) \ll \mathcal{O}(N)$

- Alwen *et al.* [ABP18] showed that for any constant $\gamma > 0$, there exists a family $\{G_{\gamma,N}\}_{N=1}^{\infty}$ of γ-extreme depth-robust DAGs with maximum indegree and outdegree $\mathcal{O}(\log N)$.
- Then $\mathrm{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$ will have degree at most $\mathcal{O}(\mathrm{indeg}(G_{\gamma,L+1}) \times \mathrm{outdeg}(G_{\gamma,L+1}) \times N/(L+1)) = \mathcal{O}(N^{\varepsilon} \log^2 N)$.

Technical Ingredients 2: γ-Extreme Depth Robust Graphs (Indegree Reduction)

Example.



Technical Ingredients 2: γ-Extreme Depth Robust Graphs (Indegree Reduction)

Theorem [Sve12]

For any integer $k \geq 2$ and constant $\varepsilon > 0$, it is Unique Games hard to distinguish between whether

1. G is (e_1, d_1)-reducible with $e_1 = N/k$ and $d_1 = k$, and
2. G is (e_2, d_2)-depth robust with $e_2 = N(1 - 1/k)$ and $d_2 = \Omega(N^{1-\varepsilon})$.

- Indegree Reduction with $\text{Sparsify}_{G_{\gamma,L+1}}(\hat{G}_{\mathcal{U}})$
- Analysis with Graph Coloring and Weighted Depth Robustness

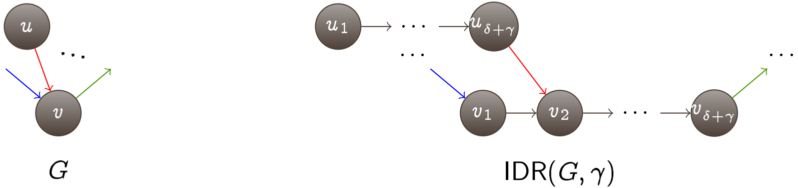
Theorem (3.3)

For any integer $k \geq 2$ and constant $\varepsilon > 0$, given a DAG G with N vertices and $\text{indeg}(G) = \mathcal{O}(N^{\varepsilon} \log^2 N)$, it is Unique Games hard to distinguish between the following cases:

- *(Completeness): G is $\left(\left(\frac{1-\varepsilon}{k}\right) N, k\right)$-reducible.*
- *(Soundness): G is $((1-\varepsilon)N, N^{1-\varepsilon})$-depth robust.*

Obtaining DAGs with Constant Indegree

- The second indegree reduction procedure $\mathsf{IDR}(G,\gamma)$ replaces each node $v \in V$ with a path $P_v = v_1, \cdots, v_{\delta+\gamma}$, where $\delta = \mathsf{indeg}(G)$.
- For each edge $(u,v) \in E$, we add the edge $(u_{\delta+\gamma}, v_j)$ whenever (u,v) is the j^{th} incoming edge of v.
- We observe that $\mathsf{indeg}(\mathsf{IDR}(G,\gamma)) = 2$.



Lemma ([ABP17])

- *If G is (e,d)-reducible, then $\mathsf{IDR}(G,\gamma)$ is $(e,(\delta+\gamma)d)$-reducible.*
- *If G is (e,d)-depth robust, then $\mathsf{IDR}(G,\gamma)$ is $(e,\gamma d)$-depth robust.*

Corollary (3.5)

For any integer $k \geq 2$ and constant $\varepsilon > 0$, given a DAG G with N vertices and $\text{indeg}(G) = 2$, it is Unique Games hard to decide whether G is (e_1, d_1)-reducible or (e_2, d_2)-depth robust for

- *(Completeness): $e_1 = \frac{1}{k} N^{\frac{1}{1+2\varepsilon}}$ and $d_1 = kN^{\frac{2\varepsilon}{1+2\varepsilon}}$.*
- *(Soundness): $e_2 = (1-\varepsilon) N^{\frac{1}{1+2\varepsilon}}$ and $d_2 = 0.9 N^{\frac{1+\varepsilon}{1+2\varepsilon}}$.*

Proof Sketch. Suppose G' is a graph with M vertices. With setting $\gamma = M^{2\varepsilon} - \delta$,

$$G' \text{ with } M \text{ vertices} \quad \longrightarrow \quad G = \text{IDR}(G', \gamma) \text{ with } (\delta + \gamma)M = M^{1+2\varepsilon} = N \text{ vertices}$$

or equivalently, $M = N^{\frac{1}{1+2\varepsilon}}$. By the previous Lemma,

$d_1 = (\delta + \gamma)k$

- $G = \text{IDR}(G', \gamma)$ is (e_1, d_1)-reducible for $e_1 = \frac{M}{k} = \frac{N^{1/(1+2\varepsilon)}}{k}$ and $d_1 = kM^{2\varepsilon} = kN^{\frac{2\varepsilon}{1+2\varepsilon}}$.
- $G = \text{IDR}(G', \gamma)$ is (e_2, d_2)-depth robust for $e_2 = (1-\varepsilon)M = (1-\varepsilon)N^{1/(1+2\varepsilon)}$, while $d_2 = \gamma M^{1-\varepsilon} = (M^{2\varepsilon} - \delta)M^{1-\varepsilon}$. Since $\delta = \mathcal{O}(M^{\varepsilon} \log^2 M)$, for sufficiently large M, $d_2 = 0.9 M^{1+\varepsilon} = 0.9 N^{\frac{1+\varepsilon}{1+2\varepsilon}}$.

□

We are now at...

Technical Ingredients 3: Superconcentrators

Recall that we have the following upper and lower bounds for $\mathsf{cc}(G_{\mathcal{U}})$:

$$\mathsf{cc}(G_{\mathcal{U}}) \geq e_2 d_2, \text{ and}$$
$$\mathsf{cc}(G_{\mathcal{U}}) \leq \min_{g \geq d_1} \left(e_1 N + gN \times \mathsf{indeg}(G_{\mathcal{U}}) + \frac{N^2 d_1}{g} \right).$$

- Even after indegree reduction, still no gap between the pebbling complexity of the two cases.

$$e_1 N = \frac{1}{k} N^{\frac{1}{1+2\varepsilon}} N = \frac{1}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} \gg (1-\varepsilon) N^{\frac{2+\varepsilon}{1+2\varepsilon}} = e_2 d_2.$$

Technical Ingredients 3: Superconcentrators

Recall that we have the following upper and lower bounds for $\mathrm{cc}(G_{\mathcal{U}})$:

$$\mathrm{cc}(G_{\mathcal{U}}) \geq e_2 d_2, \text{ and}$$

$$\mathrm{cc}(G_{\mathcal{U}}) \leq \min_{g \geq d_1} \left(e_1 N + gN \times \mathrm{indeg}(G_{\mathcal{U}}) + \frac{N^2 d_1}{g} \right).$$

- Even after indegree reduction, still no gap between the pebbling complexity of the two cases.

$$e_1 N = \frac{1}{k} N^{\frac{1}{1+2\varepsilon}} N = \frac{1}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} \gg (1-\varepsilon) N^{\frac{2+\varepsilon}{1+2\varepsilon}} = e_2 d_2.$$

Need to make it tighter!

Technical Ingredients 3: Superconcentrators

Recall that we have the following upper and lower bounds for $\mathsf{cc}(G_{\mathcal{U}})$:

$$\mathsf{cc}(G_{\mathcal{U}}) \geq e_2 d_2, \text{ and}$$

$$\mathsf{cc}(G_{\mathcal{U}}) \leq \min_{g \geq d_1} \left(e_1 N + gN \times \mathsf{indeg}(G_{\mathcal{U}}) + \frac{N^2 d_1}{g} \right).$$

- Even after indegree reduction, still no gap between the pebbling complexity of the two cases.

$$e_1 N = \frac{1}{k} N^{\frac{1}{1+2\varepsilon}} N = \frac{1}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} \gg (1-\varepsilon) N^{\frac{2+\varepsilon}{1+2\varepsilon}} = e_2 d_2.$$

→ Need to make it tighter!

Definition (Superconcentrator)

A *superconcentrator* is a graph that connects N input nodes to N output nodes so that any subset of k inputs and k outputs are connected by k vertex-disjoint paths for all $1 \leq k \leq N$. Moreover, the total number of edges in the graph should be $\mathcal{O}(N)$.

Lemma ([Pip77])

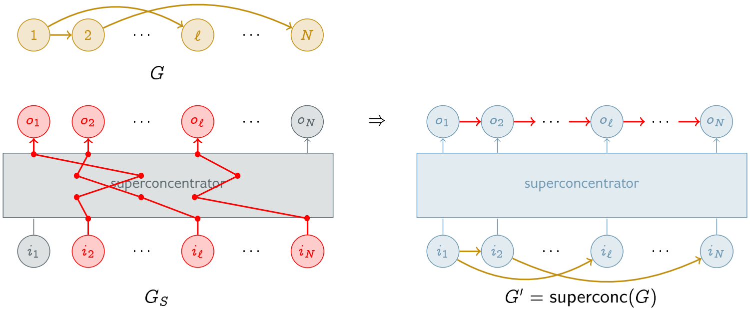
There exists a superconcentrator G with at most $42N$ vertices, containing N input vertices and N output vertices, such that $\mathsf{indeg}(G) \leq 16$ *and* $\mathsf{depth}(G) \leq \log(42N)$.

Technical Ingredients 3: Superconcentrator Overlay

Now we define the overlay of a superconcentrator on a graph G.

Definition (Superconcentrator Overlay)

Let $G = (V(G), E(G))$ be a fixed DAG with N vertices and $G_S = (V(G_S), E(G_S))$ be a (priori fixed) superconcentrator with N input vertices input$(G_S) = \{i_1, \cdots, i_N\} \subseteq V(G_S)$ and N output vertices output$(G_S) = \{o_1, \cdots, o_N\} \subseteq V(G_S)$. We call a graph $G' = (V(G_S), E(G_S) \cup E_I \cup E_O)$ a *superconcentrator overlay* where $E_I = \{(i_u, i_v) : (u, v) \in E(G)\}$ and $E_O = \{(o_i, o_{i+1}) : 1 \leq i < N\}$ and denote as $G' = \text{superconc}(G)$.



Technical Ingredients 3: Superconcentrator Overlay

If G is (e, d)-depth robust, We have the following lower bound on the pebbling complexity from [BHK+19]:

$$\mathsf{cc}(\mathsf{superconc}(G)) \geq \min\left\{\frac{eN}{8}, \frac{dN}{8}\right\}.$$

The following lemma provides a *significantly tighter* upper bound on $\mathsf{cc}(\mathsf{superconc}(G))$ with an improved pebbling strategy.

Lemma (4.4)

Let G be an (e, d)-reducible graph with N vertices with $\mathsf{indeg}(G) = 2$. *Then*

$$\mathsf{cc}(\mathsf{superconc}(G)) \leq \min_{g \geq d}\left\{2eN + 4gN + \frac{43dN^2}{g} + \frac{24N^2\log(42N)}{g} + 42N\log(42N) + N\right\}.$$

- Full description for the improved pebbling strategy: see the full paper! (Link)
- Now we can tune parameters appropriately to obtain our main result.

We are now at...

Main Theorem: Unique Games Hardness of $\mathrm{cc}(G)$

Theorem

Given a DAG G, it is Unique Games hard to approximate $\mathrm{cc}(G)$ within any constant factor.

Proof Sketch. Let $k \geq 2$ be an integer that we shall later fix. Similarly, $\varepsilon > 0$ be a constant that we shall later fix. Given a DAG G with N vertices, it is Unique Games hard to decide whether

- G is (e_1, d_1)-reducible for $e_1 = \frac{1}{k} N^{\frac{1}{1+2\varepsilon}}$, $d_1 = kN^{\frac{2\varepsilon}{1+2\varepsilon}}$, and
- G is (e_2, d_2)-depth robust for $e_2 = (1-\varepsilon) N^{\frac{1}{1+2\varepsilon}}$, $d_2 = 0.9 N^{\frac{1+\varepsilon}{1+2\varepsilon}}$.
- If G is (e_1, d_1)-reducible, then

$$\begin{aligned}\mathrm{cc}(\mathrm{superconc}(G)) &\leq \min_{g \geq d_1} \left\{ 2e_1 N + 4gN + \frac{43 d_1 N^2}{g} + \frac{24 N^2 \log(42N)}{g} + 42N \log(42N) + N \right\} \\ &\leq \frac{7}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} \qquad \text{(for } g = e_1 \text{ and sufficiently large } N\text{.)}\end{aligned}$$

- If G is (e_2, d_2)-depth robust, then $\mathrm{cc}(\mathrm{superconc}(G)) \geq \min \left\{ \frac{e_2 N}{8}, \frac{d_2 N}{8} \right\} \geq \frac{1-\varepsilon}{8} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}$.

Let $c \geq 1$ be any constant. Setting $\varepsilon = \frac{1}{2}$ and $k = 102c^2$, we have

$$\frac{7}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} = \frac{1}{16c^2} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} \ll \frac{1}{16} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} = \frac{1-\varepsilon}{8} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}. \qquad \square$$

Main Theorem: Unique Games Hardness of $\mathrm{cc}(G)$

Theorem

Given a DAG G, it is Unique Games hard to approximate $\mathrm{cc}(G)$ within any constant factor.

Proof Sketch. Let $k \geq 2$ be an integer that we shall later fix. Similarly, $\varepsilon > 0$ be a constant that we shall later fix. Given a DAG G with N vertices, it is Unique Games hard to decide whether

- G is (e_1, d_1)-reducible for $e_1 = \frac{1}{k} N^{\frac{1}{1+2\varepsilon}}$, $d_1 = kN^{\frac{2\varepsilon}{1+2\varepsilon}}$, and
- G is (e_2, d_2)-depth robust for $e_2 = (1-\varepsilon) N^{\frac{1}{1+2\varepsilon}}$, $d_2 = 0.9 N^{\frac{1+\varepsilon}{1+2\varepsilon}}$.

(Corollary 3.5)

- If G is (e_1, d_1)-reducible, then

$$\mathrm{cc}(\mathrm{superconc}(G)) \leq \min_{g \geq d_1} \left\{ 2e_1 N + 4gN + \frac{43 d_1 N^2}{g} + \frac{24 N^2 \log(42N)}{g} + 42N \log(42N) + N \right\}$$

(Lemma 4.4)

$$\leq \frac{7}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} \quad \text{(for } g = e_1 \text{ and sufficiently large } N.\text{)}$$

- If G is (e_2, d_2)-depth robust, then $\mathrm{cc}(\mathrm{superconc}(G)) \geq \min\left\{ \frac{e_2 N}{8}, \frac{d_2 N}{8} \right\} \geq \frac{1-\varepsilon}{8} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}$.

Let $c \geq 1$ be any constant. Setting $\varepsilon = \frac{1}{2}$ and $k = 102c^2$, we have

$$\frac{7}{k} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} = \frac{1}{16c^2} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} \ll \frac{1}{16} N^{\frac{2+2\varepsilon}{1+2\varepsilon}} = \frac{1-\varepsilon}{8} N^{\frac{2+2\varepsilon}{1+2\varepsilon}}. \qquad \square$$

We are now at...

Open Questions

- What we have showed: UG-Hard to c-approx for any $c > 0$.
  - Worst case analysis
  - Can we do better for the natural families of graphs?
- Possibility of bigger gap hardness of approximation (e.g. $\mathcal{O}(\text{polylog}(n))$-approx?)
- Approximation hardness from P $\neq$ NP?
- Is there any efficient c-approximation algorithm for Red-Blue pebbling where $c = o(c_b/c_r)$?

References I

Joël Alwen and Jeremiah Blocki, *Efficiently computing data-independent memory-hard functions*, CRYPTO 2016, Part II (Matthew Robshaw and Jonathan Katz, eds.), LNCS, vol. 9815, Springer, Heidelberg, August 2016, pp. 241–271.

Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak, *Depth-robust graphs and their cumulative memory complexity*, EUROCRYPT 2017, Part III (Jean-Sébastien Coron and Jesper Buus Nielsen, eds.), LNCS, vol. 10212, Springer, Heidelberg, April / May 2017, pp. 3–32.

______, *Sustained space complexity*, EUROCRYPT 2018, Part II (Jesper Buus Nielsen and Vincent Rijmen, eds.), LNCS, vol. 10821, Springer, Heidelberg, April / May 2018, pp. 99–130.

Joël Alwen and Vladimir Serbinenko, *High parallel complexity graphs and memory-hard functions*, 47th ACM STOC (Rocco A. Servedio and Ronitt Rubinfeld, eds.), ACM Press, June 2015, pp. 595–603.

Jeremiah Blocki, Benjamin Harsha, Siteng Kang, Seunghoon Lee, Lu Xing, and Samson Zhou, *Data-independent memory hard functions: New attacks and stronger constructions*, CRYPTO 2019, Part II (Alexandra Boldyreva and Daniele Micciancio, eds.), LNCS, vol. 11693, Springer, Heidelberg, August 2019, pp. 573–607.

Jeremiah Blocki and Samson Zhou, *On the computational complexity of minimal cumulative cost graph pebbling*, Financial Cryptography and Data Security (FC 2018) (2018).

Subhash Khot, *On the power of unique 2-prover 1-round games*, Proceedings on 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 767–775.

Nicholas Pippenger, *Superconcentrators*, SIAM J. Comput. **6** (1977), no. 2, 298–304.

Ola Svensson, *Hardness of vertex deletion and project scheduling*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX, and 16th International Workshop, RANDOM. Proceedings, 2012, pp. 301–312.

Questions?