Depth-Robust Graphs and Their Cumulative Memory Complexity

Joël Alwen – IST Austria

Jeremiah Blocki – Purdue University

Krzysztof Pietrzak – IST Austria

Moderately Hard Function

Intuitive Properties:

1. Computable by honest party.
2. Brute-force evaluation is very expensive for adversary.

Moderately Hard Function

Intuitive Properties:

1. Computable by honest party.
2. Brute-force evaluation is very expensive for adversary.

Applications: Limit the rate of invocations of a critical function.

Moderately Hard Function

Intuitive Properties:

1. Computable by honest party.
2. Brute-force evaluation is very expensive for adversary.

Applications: Limit the rate of invocations of a critical function.

- Password Based Cryptography
  - Password Hashing (E.g. Login Server)
  - Key Derivation Functions

Moderately Hard Function

Intuitive Properties:

1. Computable by honest party.
2. Brute-force evaluation is very expensive for adversary.

Applications: Limit the rate of invocations of a critical function.

- Password Based Cryptography
  - Password Hashing (E.g. Login Server)
  - Key Derivation Functions
- Proofs-of-Effort
  - Distributed PoW for Consensus (E.g. Ethereum, Lightcoin, Dogecoin, etc.)
  - Against SPAM [ABMW05, DGN03, DNW05]
  - Against Sybil attacks.

Why “Memory” Hard?

In practice cost-effective brute-forcing often uses GPUs, FGPAs & ASICs.

- Bitcoin miners, DES Cracker [EFF98], Sagitta Password Cracker, etc.

- Why? ASICs provide a financial incentive.
  - Specifically: Computation is cheaper for custom hardware (e.g. ASICs) then general purpose CPUs.

- Want: More egalitarian notion of “hardness” than computation.

- $\mathbb{N}$
  1. Can be computed in sequential time n.
  2. Requires as much parallel space-time as possible for any function satisfying 1.

Why “Memory” Hard?

In practice cost-effective brute-forcing often uses GPUs, FGPAs & ASICs.

- Bitcoin miners, DES Cracker [EFF98], Sagitta Password Cracker, etc.

- Why? ASICs provide a financial incentive.
  - Specifically: Computation is cheaper for custom hardware (e.g. ASICs) then general purpose CPUs.

- Want: More egalitarian notion of “hardness” than computation.

- Goal: A notion of complexity that approximates the hardware cost of an ASIC performing the computation.

- $\mathbb{N}$
  1. Can be computed in sequential time n.
  2. Requires as much parallel space-time as possible for any function satisfying 1.

Why “Memory” Hard?

In practice cost-effective brute-forcing often uses GPUs, FGPAs & ASICs.

- Bitcoin miners, DES Cracker [EFF98], Sagitta Password Cracker, etc.

- Why? ASICs provide a financial incentive.
  - Specifically: Computation is cheaper for custom hardware (e.g. ASICs) then general purpose CPUs.

- Want: More egalitarian notion of “hardness” than computation.

- Goal: A notion of complexity that approximates the hardware cost of an ASIC performing the computation.

VLSI: “Area x Time” (AT) complexity used to measure efficiency of a circuit

- $\mathbb{N}$
  1. Can be computed in sequential time n.
  2. Requires as much <u>parallel</u> space-time as possible for any function satisfying 1.

Why “Memory” Hard?

In practice cost-effective brute-forcing often uses GPUs, FGPAs & ASICs.

- Bitcoin miners, DES Cracker [EFF98], Sagitta Password Cracker, etc.

- Why? ASICs provide a financial incentive.
  - Specifically: Computation is cheaper for custom hardware (e.g. ASICs) then general purpose CPUs.

- Want: More egalitarian notion of “hardness” than computation.

- Goal: A notion of complexity that approximates the hardware cost of an ASIC performing the computation.

$[\mathrm{Per09}]$: “expensive” $\approx$ large “space × parallel-time” (ST) complexity

VLSI: “Area x Time” (AT) complexity used to measure efficiency of a circuit

- $\mathbb{N}$
  1. Can be computed in sequential time n.
  2. Requires as much parallel space-time as possible for any function satisfying 1.

Why “Memory” Hard?

$\mathbb{N}$

In practice cost-effective brute-forcing often uses GPUs, FGPAs & ASICs.

- Bitcoin miners, DES Cracker [EFF98], Sagitta Password Cracker, etc.

- Why? ASICs provide a financial incentive.
  - Specifically: Computation is cheaper for custom hardware (e.g. ASICs) then general purpose CPUs.

- Want: More egalitarian notion of “hardness” than computation.

- Goal: A notion of complexity that approximates the hardware cost of an ASIC performing the computation.

$[\mathrm{Per09}]$: “expensive” ≈ large “space × parallel-time” (ST) complexity

1. Can be computed in sequential time n.
2. Requires as much parallel space-time as possible for any function satisfying 1. Requires as much parallel space-time as possible for any function satisfying 1.

Data-(in)dependence

- An MHF is a mode of operation usually over a round function.

Data-(in)dependence

- An MHF is a mode of operation usually over a round function.

- Is the memory access pattern of the honest (sequential) evaluation algorithms input-dependent or not?
  - No: data-independent MHF (iMHF). Example: Argon2i, Balloon Hashing.
  - Yes: data-dependent MHF (dMHF). Example: scrypt, Argon2d.

Data-(in)dependence

- An MHF is a mode of operation usually over a round function.

- Is the memory access pattern of the honest (sequential) evaluation algorithms input-dependent or not?
  - No: data-independent MHF (iMHF). Example: Argon2i, Balloon Hashing.
  - Yes: data-dependent MHF (dMHF). Example: scrypt, Argon2d.

iMHF advantage: Implementations easier to secure against certain cache-timing attacks.

- Important for some password based crypto applications.

iMHFs

- Password Hashing Competition
  - Winner: Argon2i [BDK15]
  - Finalists: Catena[FLW15], Lyr2 [SAASB15], Pomelo [W15],…
  - Other contestants: Rig-v2 [CJMS14], Gambit [P14], TwoCats [C14],…
- Since PHC: Balloon Hashing [BCGS16], Alwen-Serbinenko[AS15]

iMHFs

- Password Hashing Competition
  - Winner: Argon2i [BDK15]
  - Finalists: Catena[FLW15], Lyr2 [SAASB15], Pomelo [W15],…
  - Other contestants: Rig-v2 [CJMS14], Gambit [P14], TwoCats [C14],…
- Since PHC: Balloon Hashing [BCGS16], Alwen-Serbinenko[AS15]

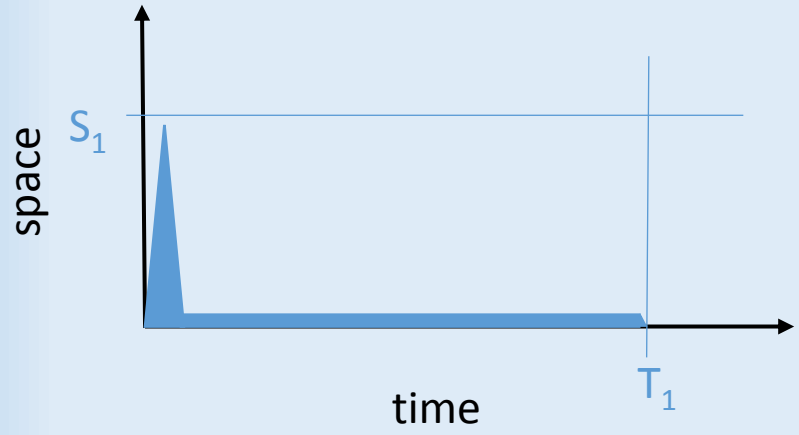
- Usually designed based on intuition and verified via cryptanalysis.
- Exceptions: Catena, Balloon Hashing, AS15
  - Balloon Hashing has security proof for sequential adversaries in ROM.
  - AS15 has proof for parallel adversaries in ROM.

Amortization and Parallelism

Problem:

Amortization and Parallelism

Problem:

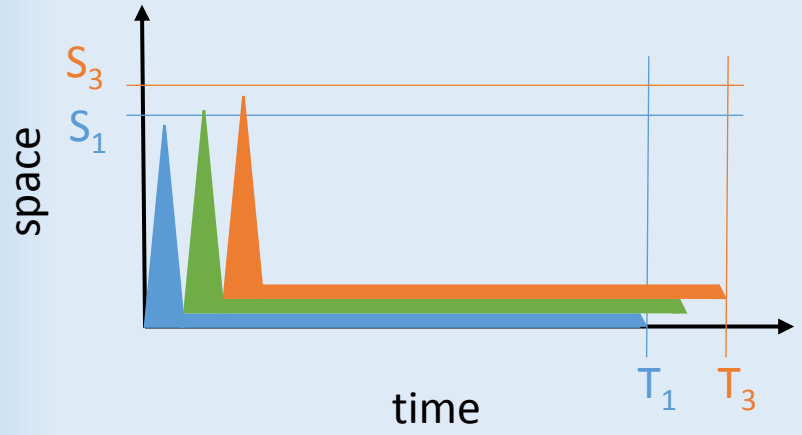


$ST_1 = S_1 \times T_1$

cost of computing
f once

Amortization and Parallelism

Problem:

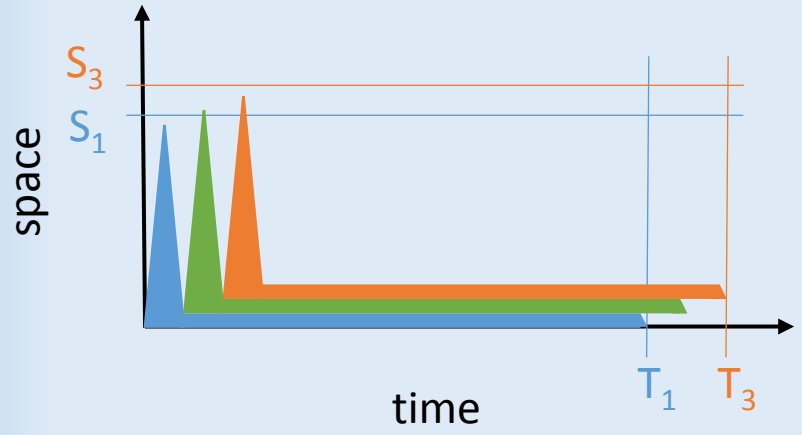


$ST_1 = S_1 \times T_1$

cost of computing
f once

Amortization and Parallelism

Problem:



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

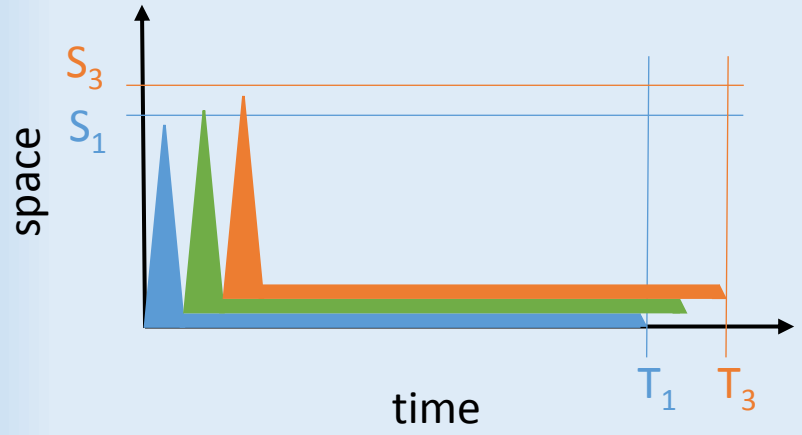
cost of computing *f* once

cost of computing *f* three times

Amortization and Parallelism

Problem: function f_n (consisting of n RO calls) such that: $ST\ f \times n\ = O(ST\ f)$

$n\,n\ \times n\ n\ n \times n \times n\ f \times n$



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

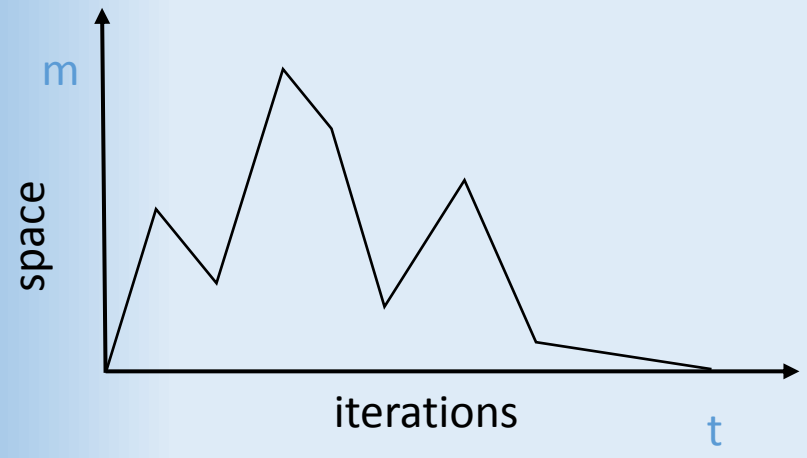
cost of computing *f* once

cost of computing *f* three times

[AS15] ∃ function f_n (consisting of n RO calls) such that: $ST\left(f^{\times\sqrt{n}}\right) = O(ST(f))$

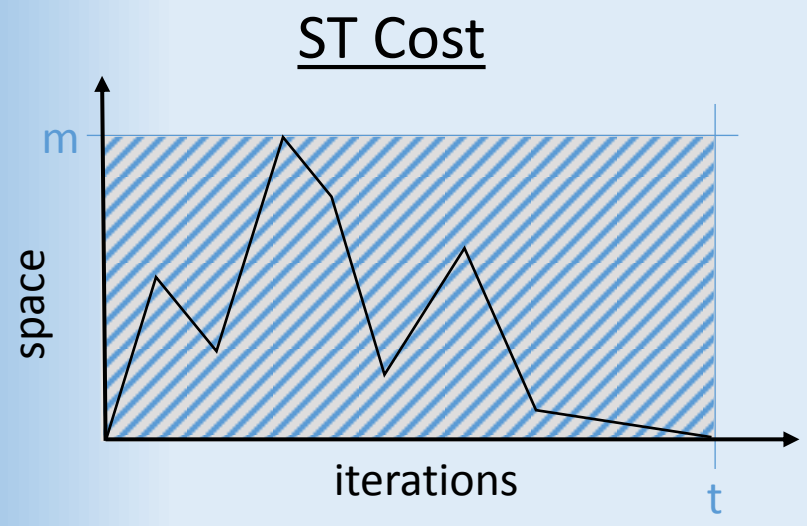
Cumulative Memory Complexity

- Fix an execution...



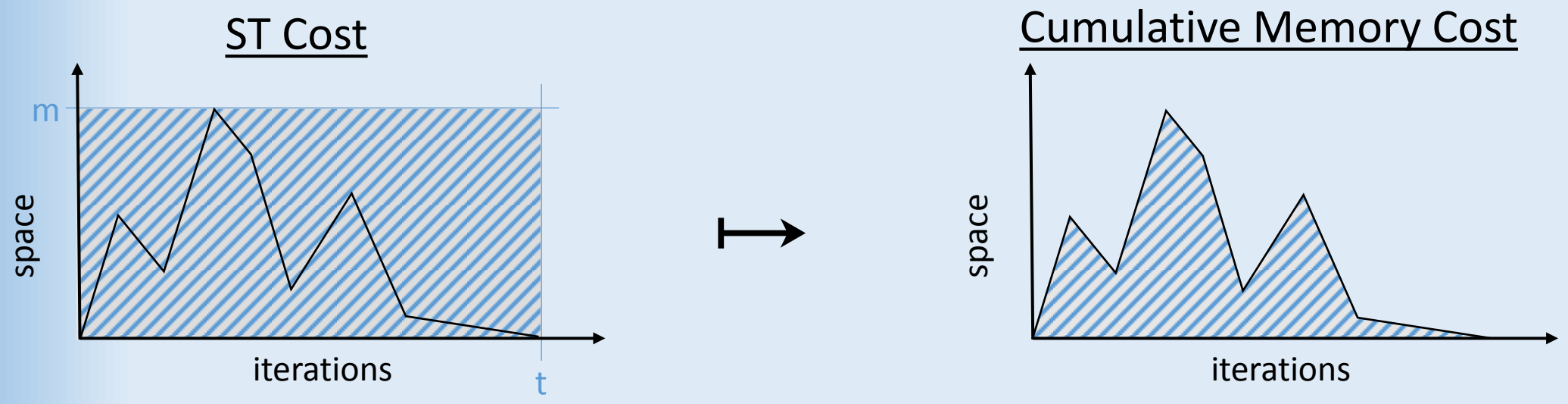
Cumulative Memory Complexity

- Fix an execution...



Cumulative Memory Complexity

- Fix an execution...
- Idea: Define the cost to be area under the “memory curve”.



Parallel Pebbling Game

- Intuition: Models Parallel Computation

- Iteratively place pebbles on the nodes of DAG *G*.
  - Initially no pebbles on *G*. Each node can have at most one pebble.

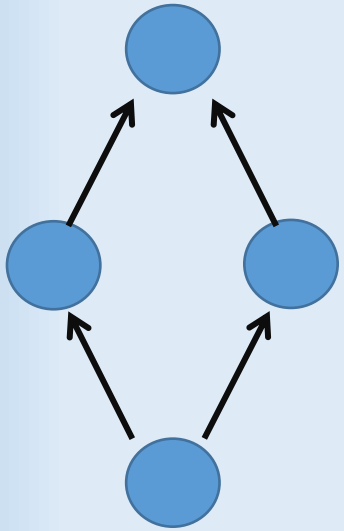
- Goal: Place a pebble on sink node(s) of *G*.

- Rules:
  1. Can place a pebble on *v* only if all of parents of *v* currently have a pebble.
     $\Rightarrow$ can always place a pebble on source nodes
  2. Can remove any pebble at any time.

A New Parallel Pebbling Game

Parallel Pebbling Game: Same as Black Pebbling, except can touch many pebbles per iteration.

Complexity: Cumulative Pebbling Complexity (CPC).

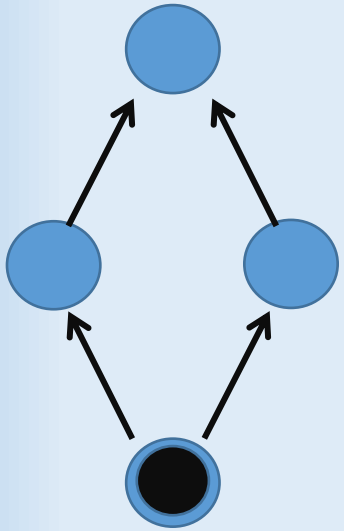


CPC-cost =

A New Parallel Pebbling Game

Parallel Pebbling Game: Same as Black Pebbling, except can touch many pebbles per iteration.

Complexity: Cumulative Pebbling Complexity (CPC).

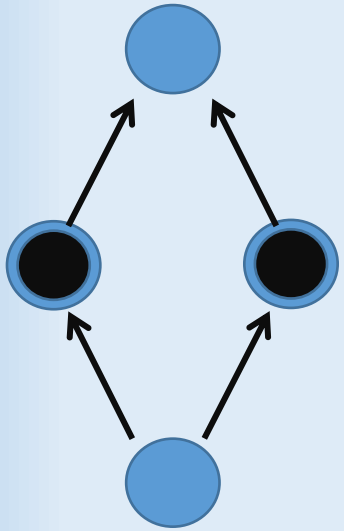


CPC-cost = 1+

A New Parallel Pebbling Game

Parallel Pebbling Game: Same as Black Pebbling, except can touch many pebbles per iteration.

Complexity: Cumulative Pebbling Complexity (CPC).

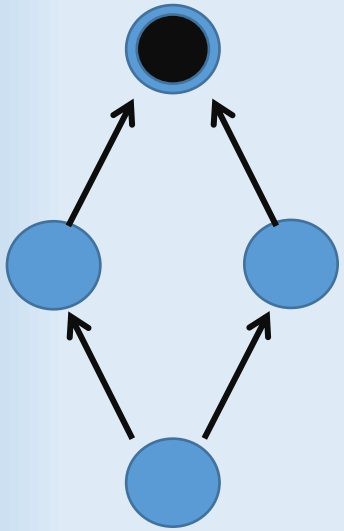


CPC-cost = 1+ 2+

A New Parallel Pebbling Game

Parallel Pebbling Game: Same as Black Pebbling, except can touch many pebbles per iteration.

Complexity: Cumulative Pebbling Complexity (CPC).

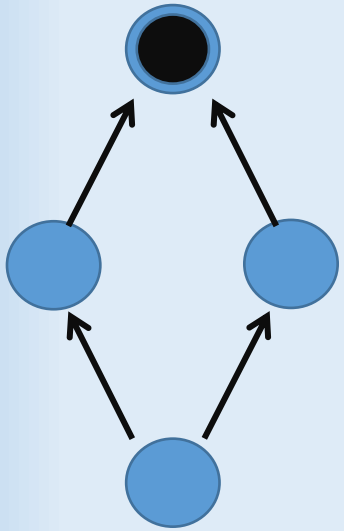


CPC-cost = 1+ 2+ 1 = 4

A New Parallel Pebbling Game

Parallel Pebbling Game: Same as Black Pebbling, except can touch many pebbles per iteration.

Complexity: Cumulative Pebbling Complexity (CPC).



CPC-cost = 1+ 2+ 1 = 4

CPC(Graph G) := min CPC(Pebbling of G)

“We Can Only Pebble A Graph Function”

- View a mode of operation as DAG. (hash graph”, “graph function”)

“We Can Only Pebble A Graph Function”

- View a mode of operation as DAG. (hash graph”, “graph function”)

- Theorem [AS15]
  - Let $\mathcal{H}: \{0,1\}^{2w} \rightarrow \{0,1\}^{w}$ be a RO and G be a DAG.
  - Let f be the function given by (G, $\mathcal{H}$).

$$\Longrightarrow CMC(f) \geq CPC(G)/4$$

“We Can Only Pebble A Graph Function”

- View a mode of operation as DAG. (hash graph”, “graph function”)

- Theorem [AS15]
  - Let $\mathcal{H}: \{0,1\}^{2w} \rightarrow \{0,1\}^{w}$ be a RO and G be a DAG.
  - Let f be the function given by (G, $\mathcal{H}$).

$$\Big\} \Longrightarrow CMC(f) \geq CPC(G)/4$$

- New Goals:
  - Security Proofs: find <u>constant indegree</u> graph with high CPC.
  - Attacks: find low CPC pebbling strategies.

Some Previous Results About CPC

MHF	Upper Bound	Lower Bound
Argon2i-A Balloon Hashing	$\tilde{O}(n^{1.75})$ [AB16]	–
Argon2i-B	$O(n^{1.8})$ [AB17]	–
Catena	$O(n^{1.67})$ [AB16]	–
AS15	–	$\Omega(n^2/\log^{10}(n))$ [AS15]
Any iMHF	$O\left(\frac{n^2 \mathrm{loglog}(n)}{\log(n)}\right)$[AB16]	–
SCRYPT (dMHF)	$O(n^2)$	–

New Results

MHF	Upper Bound	Lower Bound
Argon2i-A Balloon Hashing	$\tilde{O}(n^{1.71})$ [This Work] ~~$\tilde{O}(n^{1.75})$~~ [AB16]	$\tilde{\Omega}(n^{1.\bar{6}})$ [This Work]
Argon2i-B	$O(n^{1.8})$ [AB17]	$\tilde{\Omega}(n^{1.\bar{6}})$ [This Work]
Catena	$\tilde{O}(n^{1.618})$ [This Work] ~~$O(n^{1.67})$~~ [AB16]	$\tilde{\Omega}(n^{1.5})$ [This Work]
AS15	–	$\Omega(n^2/\log^{10}(n))$ [AS15]
This Work	–	$\Omega(n^2/\log(n))$
Any iMHF	$O\left(\frac{n^2 \text{loglog}(n)}{\log(n)}\right)$ [AB16]	–
SCRYPT (dMHF)	$O(n^2)$	$\Omega(n^2)$ [Next Talk]

Depth-Robust Graphs

A directed ac

Depth-Robust Graphs

A directed ac

- G called “(e,d) -reducible” iff G is not (e,d)-depth-robust.

Depth-Robust Graphs

A directed ac(n), Ω(n))-depth-robust with indegree O(log(n)).

- G called “(e,d) -reducible” iff G is <u>not</u> (e,d)-depth-robust.

History:

- First considered: Erdös, Graham, Szemerédi [EGS77]
- EGS graph: (Ω(n), Ω(n))-depth-robust with indegree O(log(n)).

New Construction: Technique

Lemma "Indegree Reduction": If G has indegree δ and is (e,d)-depth-robust then there exists H with indegree 2 and:

size(H) $\leq 2\delta$*size(G) H is (e,dδ)-depth-robust

heorem: Let G=(V,E) be (e,d)-depth-robust then $\mathrm{CPC}(G) > ed$.

Corollary: There is a DAG G with maximum indegree $\delta = 2$ and $\mathrm{E_R}(G) = \Omega\left(\frac{n^2}{\log n}\right)$. Furthermore, there is a sequential pebbling algorithm N with cost $\mathrm{E_R}(N) = O\left(\frac{n^2}{\log n}\right)$.

New Construction: Technique

Let G=(V,E) be (e,d)-depth-robust then CPC $G\ GG\ G > eedd$.

Lemma "Indegree Reduction": If G has indegree δ and is (e,d)-depth-robust then there exists H with indegree 2 and:

size(H) $\leq 2\delta$*size(G) H is (e,dδ)-depth-robust

Theorem: Let G=(V,E) be (e,d)-depth-robust then $\mathrm{CPC}(G) > ed$.

Corollary: There is a DAG G with maximum indegree $\delta = 2$ and $\mathrm{E_R}(G) = \Omega\left(\frac{n^2}{\log n}\right)$. Furthermore, there is a sequential pebbling algorithm N with cost $\mathrm{E_R}(N) = O\left(\frac{n^2}{\log n}\right)$.

New Construction: Technique

2 and ER $G\ GG\ G$ =Ω $n\ 2$ log n $n\ 2$ log n $n\ 2\ nn\ n\ 2\ 2\ n\ 2$ $n\ 2$ log n log n log log $n\ nn$ log n $n\ 2$ log n $n\ 2$ log n .
Furthermore, there is a sequential pebbling algorithm N with cost ER $N\ NN\ N$ =OO $n\ 2$ log n $n\ 2$ log n $n\ 2\ nn\ n\ 2\ 2\ n\ 2$ $n\ 2$ log n log n log log $n\ nn$ log n $n\ 2$ log n $n\ 2$ log n .

Let G=(V,E) be (e,d)-depth-robust then CPC $G\ GG\ G > eedd$.

Lemma "Indegree Reduction": If G has indegree δ and is (e,d)-depth-robust then there exists H with indegree 2 and:

size(H) $\leq 2\delta$*size(G) H is (e,dδ)-depth-robust

Theorem: Let G=(V,E) be (e,d)-depth-robust then $\mathrm{CPC}(G) > ed$.

Corollary: There is a DAG G with maximum indegree $\delta = 2$ and

New Bounds For Known Constructions

- New Lower Bounds
  - Lower bound Depth-Robustness of Balloon Hashing and Argon2i-{A,B}
  - Second Technique: “Dispersal” Property of a DAG
    1. Lower Bound CPC(G) in terms of Dispersal properties of G
    2. Analyze Dispersal properties of Catena (Dragonfly and Butterfly versions)

- New Upper Bounds
  - Idea: Recursive version of AB16 Algorithm
  - Analyze Depth-Robustness “curves” for Argon2i, Catena & Balloon Hashing

The CPC of Depth-Robust Graphs

- **Theorem**: Let G=(V,E) be (e,d)-depth-robust then $\mathrm{CPC}(G) > ed$.

"The proof of this result is really an incredibly simple and beautiful two-line argument….I generally view simplicity as a positive, and this is the right proof. But there is such as thing as too simple..."

Proof by Picture

Given: DAG G = (V,E)
is (e,d)-Depth-Robust

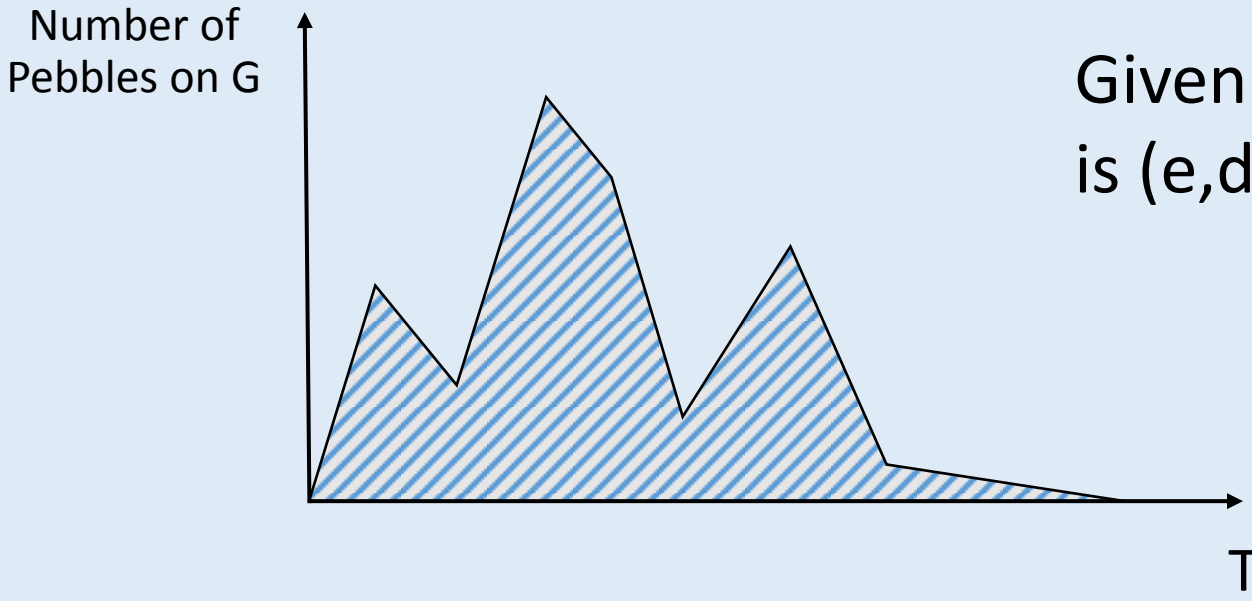
Proof by Picture

Given: DAG G = (V,E)
is (e,d)-Depth-Robust

Fix an optimal pebbling of G: $P = (P_1, P_2, P_3, \dots)$

$P_i \subseteq V$

Proof by Picture

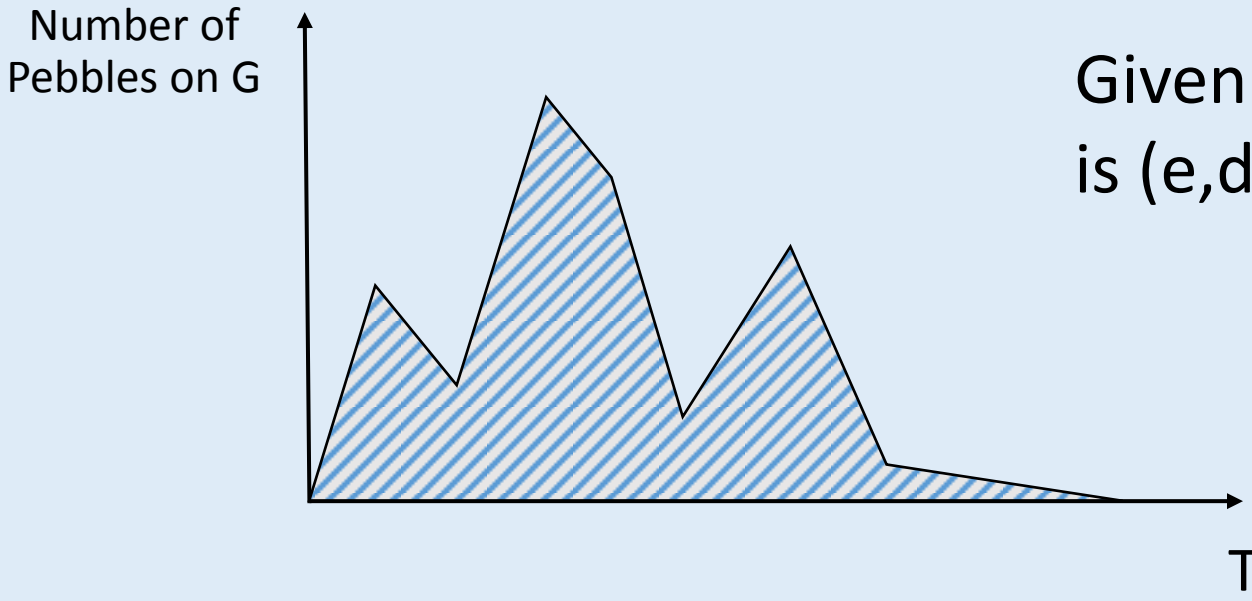


Given: DAG G = (V,E)
is (e,d)-Depth-Robust

Fix an optimal pebbling of G: $P = (P_1, P_2, P_3, \dots)$

$P_i \subseteq V$

Proof by Picture



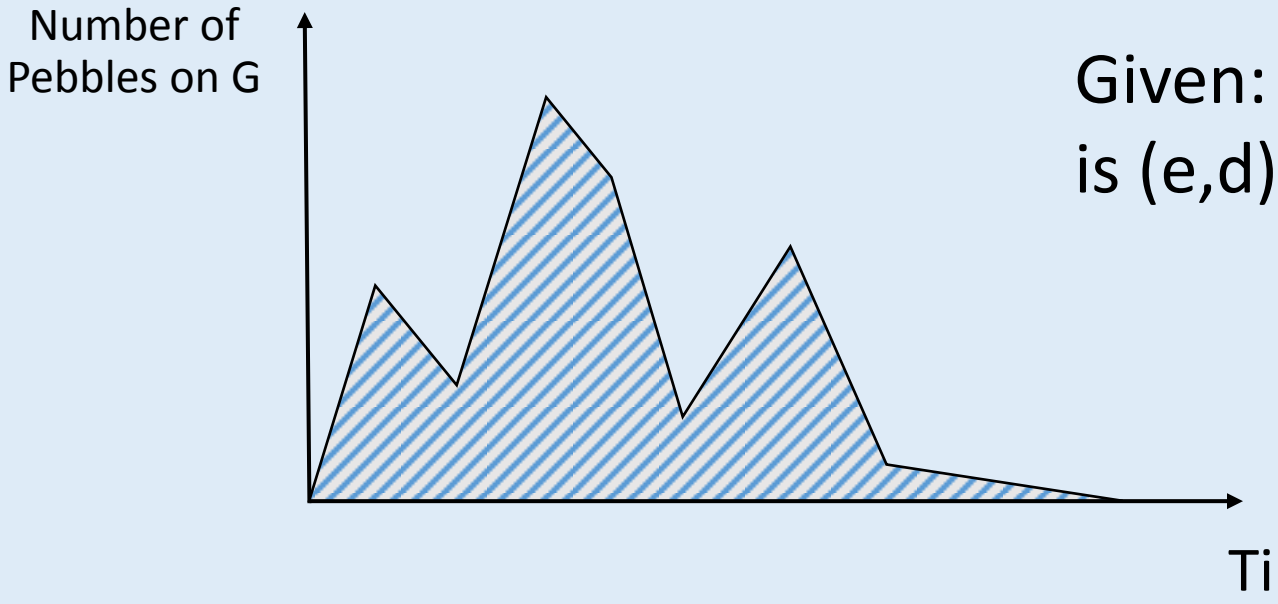
Given: DAG G = (V,E)
is (e,d)-Depth-Robust

Fact 1: Area under curve
= CPC (G)

Fix an optimal pebbling of G: $P = (P_1, P_2, P_3, \dots)$

$P_i \subseteq V$

Proof by Picture



Given: DAG G = (V,E)
is (e,d)-Depth-Robust

Fact 1: Area under curve = CPC (G)

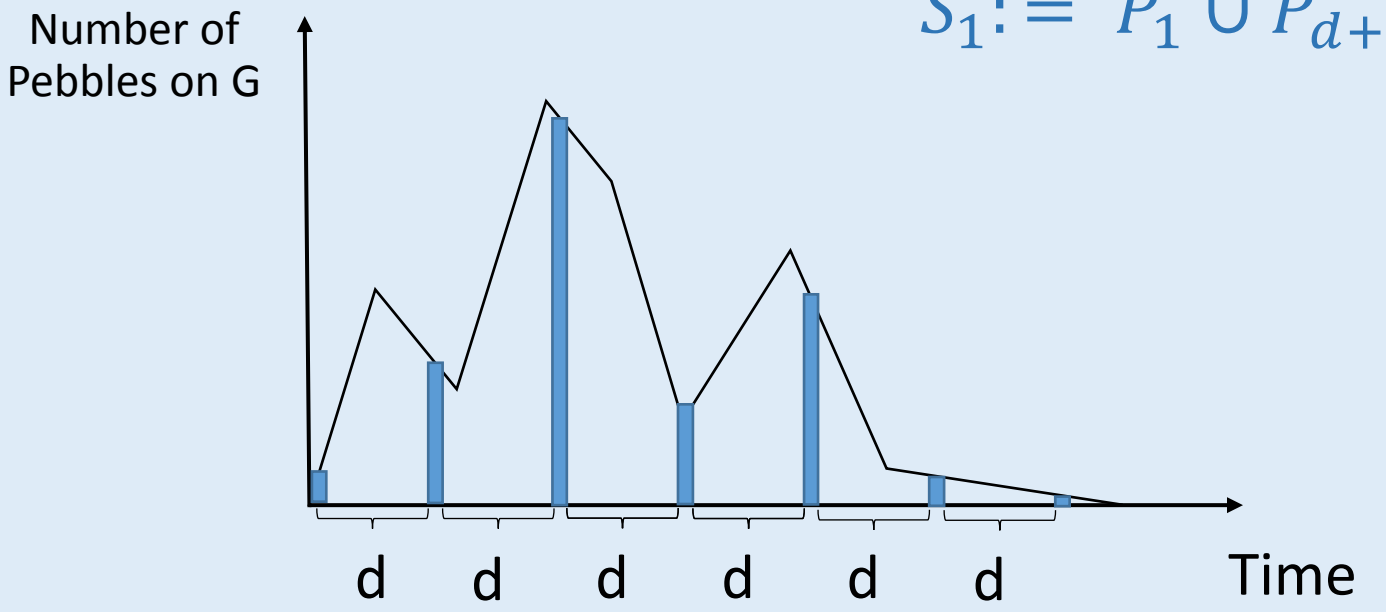
Fact 2: P pebbles every node in G at least once.

Fix an optimal pebbling of G: $P = (P_1, P_2, P_3, \dots)$

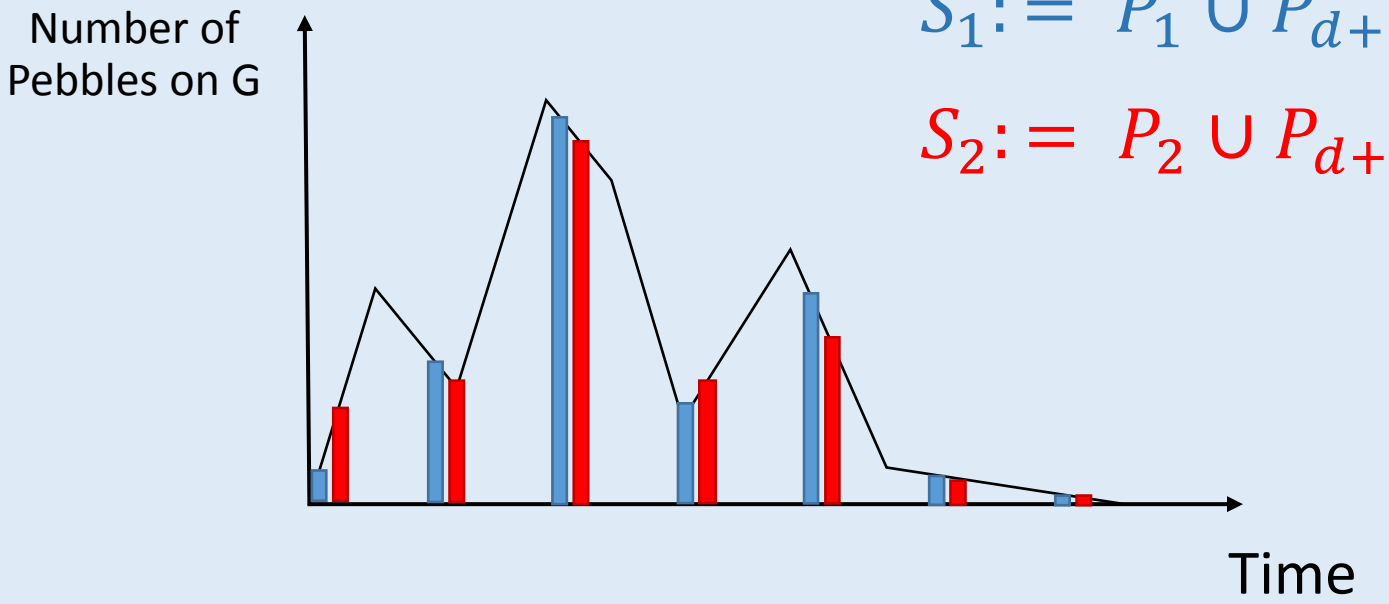
$P_i \subseteq V$

Proof by Picture

$S_1 := P_1 \cup P_{d+1} \cup P_{2d+1} \cup \ldots \subseteq V$



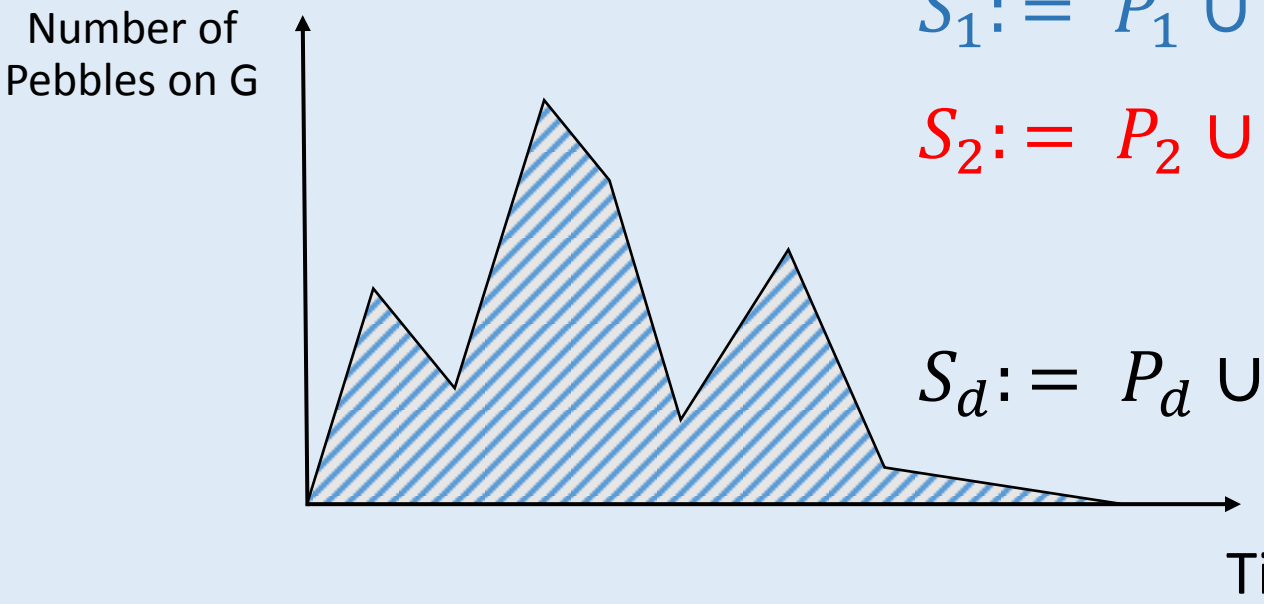
Proof by Picture



$$S_1 := P_1 \cup P_{d+1} \cup P_{2d+1} \cup \ldots \subseteq V$$

$$S_2 := P_2 \cup P_{d+2} \cup P_{2d+2} \cup \ldots \subseteq V$$

Proof by Picture



$S_1 := P_1 \cup P_{d+1} \cup P_{2d+1} \cup \ldots \subseteq V$

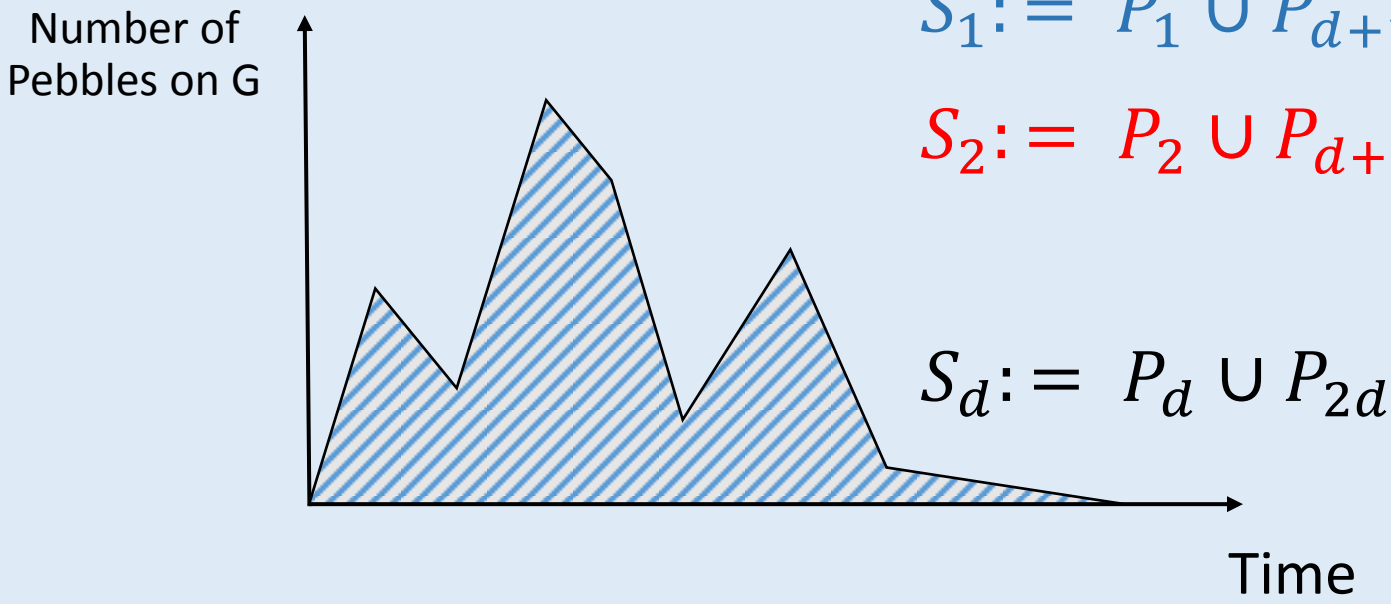
$S_2 := P_2 \cup P_{d+2} \cup P_{2d+2} \cup \ldots \subseteq V$

…

$S_d := P_d \cup P_{2d} \cup P_{3d} \cup \ldots \subseteq V$

$$\sum_{i=1}^{i=d} |S_i| \leq CPC(G)$$

Proof by Picture



$$S_1 := P_1 \cup P_{d+1} \cup P_{2d+1} \cup \ldots \subseteq V$$

$$S_2 := P_2 \cup P_{d+2} \cup P_{2d+2} \cup \ldots \subseteq V$$

…

$$S_d := P_d \cup P_{2d} \cup P_{3d} \cup \ldots \subseteq V$$

$$\Rightarrow \exists i \text{ such that } |S_i| \leq \frac{CPC(G)}{d}$$

Proof by Picture

Let graph G’ = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset \ \ \forall j$.

P’ legally pebbles every node in G’ at least once.

Notice: Every d steps P’ has no pebbles on G’.

$\Rightarrow$ No path in G’ is longer than d-1.

$\Rightarrow \ e < \ |S_i| \leq \frac{CPC(G)}{d}$

$\Rightarrow \ ed < CPC(G)$.

Proof by Picture

Let graph G’ = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset \ \ \forall j$.

P’ legally pebbles every node in G’ at least once.

Notice: Every d steps P’ has no pebbles on G’.

⇒ No path in G’ is longer than d-1.

⇒ $e < |S_i| \leq \frac{CPC(G)}{d}$

⇒ $ed < CPC(G)$.

Fact 2: P pebbles every node in G at least once.

Proof by Picture

P’ legally pebbles every node in G’ at least once.

Let graph G’ = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset \ \ \forall j$.

$\Rightarrow$ P’ legally pebbles every node in G’ at least once.

Notice: Every d steps P’ has no pebbles on G’.

$\Rightarrow$ No path in G’ is longer than d-1.

$\Rightarrow \quad e < \ |S_i| \leq \frac{CPC(G)}{d}$

$\Rightarrow \quad ed < CPC(G).$

Proof by Picture

P’ legally pebbles every node in G’ at least once.

Let graph G’ = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset \ \ \forall j$.

$\Rightarrow$ P’ legally pebbles every node in G’ at least once.

Fact 3: Pebbling a path of length d takes at least d time.

Notice: Every d steps P’ has no pebbles on G’.

$\Rightarrow$ No path in G’ is longer than d-1.

$\Rightarrow \quad e < \ |S_i| \leq \frac{CPC(G)}{d}$

$\Rightarrow \quad ed < CPC(G).$

Proof by Picture

P’ legally pebbles every node in G’ at least once.

Let graph G’ = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset \ \ \forall j$.

Notice: Every d steps P’ has no pebbles on G’.

Notice: Every d steps P’ has no pebbles on G’.

$\Rightarrow$ No path in G’ is longer than d-1.

$\Rightarrow \quad e < \ |S_i| \leq \frac{CPC(G)}{d}$

$\Rightarrow \quad ed < CPC(G)$.

Fact 3: Pebbling a path of length d takes at least d time.

Proof by Picture

No path in G’ is longer than d-1.

P’ legally pebbles every node in G’ at least once.

Let graph G’ = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset \ \ \forall j$.

Notice: Every d steps P’ has no pebbles on G’.

$\Rightarrow$ No path in G’ is longer than d-1.

$\Rightarrow$ No path in G’ is longer than d-1.

$\Rightarrow \quad e < \ |S_i| \leq \frac{CPC(G)}{d}$

$\Rightarrow \quad ed < CPC(G).$

Proof by Picture

No path in G’ is longer than d-1.

P’ legally pebbles every node in G’ at least once.

Let graph G’ = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset \ \ \forall j$.

Notice: Every d steps P’ has no pebbles on G’.

⇒ No path in G’ is longer than d-1.

⇒ No path in G’ is longer than d-1.

But G is (e,d)-depth robust

⇒ $e < |S_i| \leq \frac{CPC(G)}{d}$

⇒ $ed < CPC(G)$.

Proof by Picture

$ee< \ S\ i \ \ S\ i\ SS\ S\ i\ ii\ S\ i \ \ S\ i \leq CPC(G)\ d\ CCPPCC(GG)\ CPC(G)\ d\ d\ d\ CPC(G)\ d$

No path in G' is longer than d-1.

P' legally pebbles every node in G' at least once.

Let graph G' = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset \ \ \forall j$.

Notice: Every d steps P' has no pebbles on G'.

$\Rightarrow \quad e < \ |S_i| \leq \frac{CPC(G)}{d}$

$\Rightarrow$ No path in G' is longer than d-1.

$\Rightarrow e < \ |S_i| \leq \frac{CPC(G)}$

Proof by Picture

$eedd<CCPPCC\ G\ GG\ G$.

$ee<\ \ S\ i\ \ \ S\ i\ SS\ S\ i\ ii\ S\ i\ \ \ S\ i\ \leq CPC(G)\ d\ CCPPCC(GG)\ CPC(G)\ d\ d\ d\ CPC(G)\ d$

No path in G' is longer than d-1.

P' legally pebbles every node in G' at least once.

Let graph G' = G with nodes in S_i removed.

- Let pebbling P' = P but with $P_{i+jd} = \emptyset\ \ \forall j$.

Notice: Every d steps P' has no pebbles on G'.

$\Rightarrow\ \ ed < CPC(G)$.

⇒ No path in G' is longer than d-1.

Thank You