

Derandomizing Isolation in Space-Bounded Settings

Dieter van Melkebeek and Gautam Prakriya

University of Wisconsin-Madison

6 July 2017

Isolation

Isolation

- ▶ Computational Problem

Π : instance $x \rightarrow$ set of solutions for x .

Eg.: Satisfiability, Reachability...

Isolation

- ▶ Computational Problem

Π : instance $x \rightarrow$ set of solutions for x .

Eg.: Satisfiability, Reachability...

- ▶ An isolation for Π is a map f that satisfies the following for all instances x :

(1) $|\Pi(f(x))| \leq 1$

(2) $|\Pi(f(x))| = 0$ iff $|\Pi(x)| = 0$

Isolation

- ▶ Computational Problem

Π : instance $x \rightarrow$ set of solutions for x .

Eg.: Satisfiability, Reachability...

- ▶ An isolation for Π is a map f that satisfies the following for all instances x :

(1) $|\Pi(f(x))| \leq 1$

(2) $|\Pi(f(x))| = 0$ iff $|\Pi(x)| = 0$

[(3) $\Pi(f(x)) \subseteq \Pi(x)$]

With (3) f is called a *pruning*.

Motivation

- ▶ In algorithms:
 - Avoiding cancellations in algebraic settings
 - Coordination in parallel algorithms
 - ...
- ▶ In complexity theory:
 - Used to show problems become no easier when restricted to instances with unique solutions.

Motivation

- ▶ In algorithms:
 - Avoiding cancellations in algebraic settings
 - Coordination in parallel algorithms
 - ...
- ▶ In complexity theory:
 - Used to show problems become no easier when restricted to instances with unique solutions.
- ▶ All known generic isolations are randomized.

Space-Bounded Setting

- ▶ NL: languages accepted by non-deterministic logspace machines.
- ▶ UL: languages accepted by unambiguous logspace machines, i.e., non-deterministic machines with at most one accepting path on every input.

Space-Bounded Setting

- ▶ NL: languages accepted by non-deterministic logspace machines.
- ▶ UL: languages accepted by unambiguous logspace machines, i.e., non-deterministic machines with at most one accepting path on every input.
- ▶ Open: $NL \stackrel{?}{\subseteq} UL$

Space-Bounded Setting

- ▶ NL: languages accepted by non-deterministic logspace machines.
- ▶ UL: languages accepted by unambiguous logspace machines, i.e., non-deterministic machines with at most one accepting path on every input.
- ▶ Open: $NL \stackrel{?}{\subseteq} UL$
 - $\iff REACH \in UL.$
 - $\iff REACH$ has a logspace isolation.

Results

Theorem 1

$$\text{NL} \subseteq \text{UTISP}(\text{poly}(n), O(\log^{3/2} n)).$$

Compare to

$\text{NL} \subseteq \text{DSPACE}(O(\log^2 n))$ [Savitch].

$\text{Reach} \in \text{DTISP}(\text{poly}(n), n/2^{\sqrt{\log n}})$ [Barnes et al.].

Results

Theorem 1

$$\text{NL} \subseteq \text{UTISP}(\text{poly}(n), O(\log^{3/2} n)).$$

Compare to

$$\text{NL} \subseteq \text{DSPACE}(O(\log^2 n)) \text{ [Savitch].}$$

$$\text{Reach} \in \text{DTISP}(\text{poly}(n), n/2^{\sqrt{\log n}}) \text{ [Barnes et al.]}$$

Theorem 2

$\text{NL} \subseteq \text{L/poly}$ if there is a logspace *pruning* for REACH.

Results

Theorem 1

$$\text{NL} \subseteq \text{UTISP}(\text{poly}(n), O(\log^{3/2} n)).$$

Compare to

$$\text{NL} \subseteq \text{DSPACE}(O(\log^2 n)) \text{ [Savitch].}$$

$$\text{Reach} \in \text{DTISP}(\text{poly}(n), n/2^{\sqrt{\log n}}) \text{ [Barnes et al.]}$$

Theorem 2

$\text{NL} \subseteq \text{L/poly}$ if there is a logspace *pruning* for REACH.

Similar results for LogCFL.

Min-Isolation

Min-isolating weight assignment

For $G = (V, E)$, a weight assignment $w : V \rightarrow \mathbb{N}$ is *min-isolating* if for all $s, t \in V$, there is at most one min-weight s - t path.

The Isolation Lemma of Mulmuley, Vazirani and Vazirani implies
For any graph G , a uniformly random choice of weights from $\{1, \dots, n^3\}$ is min-isolating with high probability.

Min-Isolation

Min-isolating weight assignment

For $G = (V, E)$, a weight assignment $w : V \rightarrow \mathbb{N}$ is *min-isolating* if for all $s, t \in V$, there is at most one min-weight s - t path.

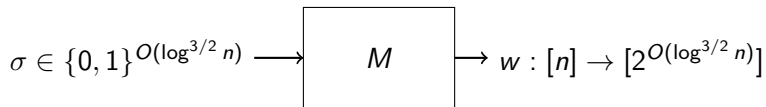
The Isolation Lemma of Mulmuley, Vazirani and Vazirani implies
For any graph G , a uniformly random choice of weights from $\{1, \dots, n^3\}$ is min-isolating with high probability.

- ▶ Gal and Wigderson: $\text{NL} \subseteq \mathcal{R} \cdot \text{promise-UL}$
- ▶ Reinhardt and Allender: $\text{NL} \subseteq \mathcal{R} \cdot (\text{co-UL} \cap \text{UL})$.

Derandomization

Key Lemma

There exists a logspace machine M

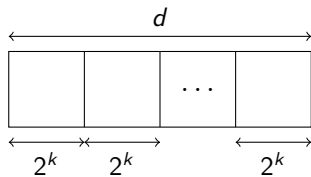


s.t. for all G on n vertices,

$$\Pr_{\sigma}[w \text{ is min-isolating for } G] \geq 1 - 1/n.$$

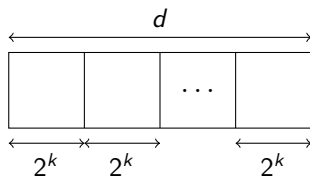
Proof of Key Lemma

Layered DAG:



Proof of Key Lemma

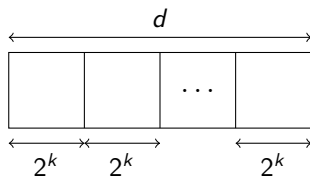
Layered DAG:



- ▶ Iteratively build $w_0, w_1, \dots, w_{\log d}$.
- ▶ Invariant: w_k is
 - ▶ min-isolating for blocks of length 2^k .
 - ▶ non-zero only on internal vertices.

Proof of Key Lemma

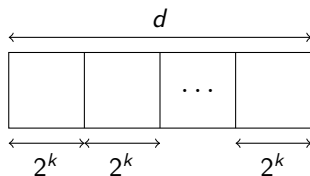
Layered DAG:



- ▶ Iteratively build $w_0, w_1, \dots, w_{\log d}$.
- ▶ Invariant: w_k is
 - ▶ min-isolating for blocks of length 2^k .
 - ▶ non-zero only on internal vertices.
- ▶ Start with $w_0 \equiv 0$. Goal: $w_{\log d}$.

Proof of Key Lemma

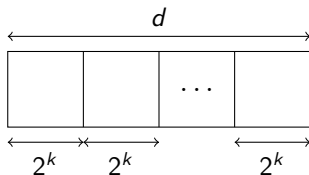
Layered DAG:



- ▶ Iteratively build $w_0, w_1, \dots, w_{\log d}$.
- ▶ Invariant: w_k is
 - ▶ min-isolating for blocks of length 2^k .
 - ▶ non-zero only on internal vertices.
- ▶ Start with $w_0 \equiv 0$. Goal: $w_{\log d}$.
- ▶ Relevant Parameters:
 - ▶ Number of random bits R_k for w_k .
 - ▶ Maximum path weight W_k for w_k .

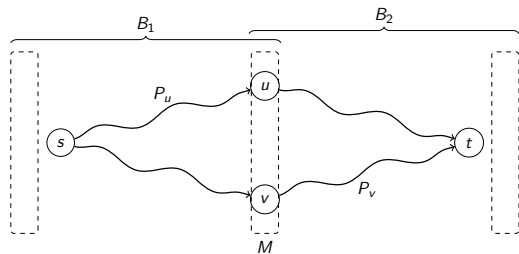
Proof of Key Lemma

Layered DAG:

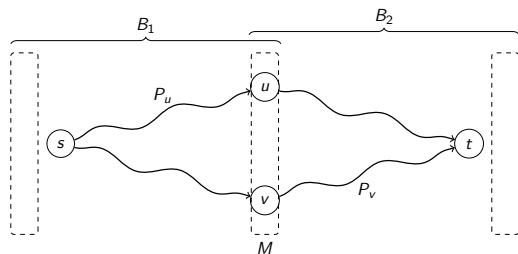


- ▶ Iteratively build $w_0, w_1, \dots, w_{\log d}$.
- ▶ Invariant: w_k is
 - ▶ min-isolating for blocks of length 2^k .
 - ▶ non-zero only on internal vertices.
- ▶ Start with $w_0 \equiv 0$. Goal: $w_{\log d}$.
- ▶ Relevant Parameters:
 - ▶ Number of random bits R_k for w_k .
 - ▶ Maximum path weight W_k for w_k .
- ▶ Space used by unambiguous algorithm for REACH is $O(R + \log W + \log n)$, where $R := R_{\log d}$ and $W := W_{\log d}$.

Iteration $k \rightarrow k + 1$

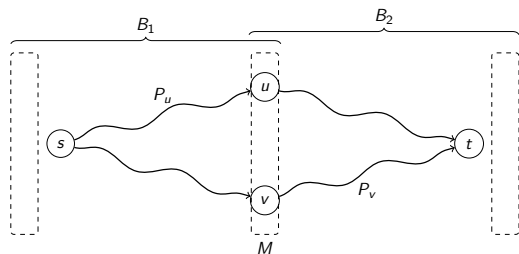


Iteration $k \rightarrow k + 1$



- ▶ w_{k+1} :
 - ▶ Same as w_k on vertices internal to blocks of length 2^k .
 - ▶ Additionally assigns weights to L_{k+1} , i.e., vertices in middle layers M of length 2^{k+1} blocks.

Iteration $k \rightarrow k + 1$



- ▶ w_{k+1} :
 - ▶ Same as w_k on vertices internal to blocks of length 2^k .
 - ▶ Additionally assigns weights to L_{k+1} , i.e., vertices in middle layers M of length 2^{k+1} blocks.
- ▶ Disambiguation requirement:

$$w_k(P_u) + w_{k+1}(u) \neq w_k(P_v) + w_{k+1}(v)$$

$$\forall s \in B_1, \forall t \in B_2, \forall u \neq v \in M, \forall \text{ blocks } B = B_1 \cup B_2.$$

Shifting $k \rightarrow k + 1$

$$w_k(P_u) + w_{k+1}(u) \neq w_k(P_v) + w_{k+1}(v)$$

Define

$$w_{k+1}(v) := \text{index}(v) \cdot (W_k + 1) \text{ for } v \in L_{k+1}.$$

$$\begin{array}{r}
 w_{k+1}(P_u) = \boxed{\text{index}(u)} \boxed{w_k(P_u)} \\
 \Downarrow \\
 w_{k+1}(P_v) = \boxed{\text{index}(v)} \boxed{w_k(P_v)}
 \end{array}$$

Shifting $k \rightarrow k + 1$

$$w_k(P_u) + w_{k+1}(u) \neq w_k(P_v) + w_{k+1}(v)$$

Define

$$w_{k+1}(v) := \text{index}(v) \cdot (W_k + 1) \text{ for } v \in L_{k+1}.$$

$$\begin{array}{c} w_{k+1}(P_u) = \boxed{\text{index}(u)} \quad w_k(P_u) \\ \Downarrow \\ w_{k+1}(P_v) = \boxed{\text{index}(v)} \quad w_k(P_v) \end{array}$$

► Parameters:

$$R_{k+1} = R_k \Rightarrow R_k = 0.$$

$$W_{k+1} \leq n^2 \cdot (W_k + 1) \Rightarrow W_k = n^{O(k)}.$$

$$R + \log W = O(\log^2 n).$$

Hashing $k \rightarrow k + 1$

$$w_k(P_u) + w_{k+1}(u) \neq w_k(P_v) + w_{k+1}(v)$$

Define

$$w_{k+1}(v) := h(v) \text{ for } v \in L_{k+1},$$

where $h : [n] \rightarrow [r]$ is picked uniformly at random from a universal hash family.

Hashing $k \rightarrow k + 1$

$$w_k(P_u) + w_{k+1}(u) \neq w_k(P_v) + w_{k+1}(v)$$

Define

$$w_{k+1}(v) := h(v) \text{ for } v \in L_{k+1},$$

where $h : [n] \rightarrow [r]$ is picked uniformly at random from a universal hash family.

- ▶ h satisfies disambiguation requirement for a fixed choice of block, vertices s, t, u, v w.p. $1 - 1/r$.

Hashing $k \rightarrow k + 1$

$$w_k(P_u) + w_{k+1}(u) \neq w_k(P_v) + w_{k+1}(v)$$

Define

$$w_{k+1}(v) := h(v) \text{ for } v \in L_{k+1},$$

where $h : [n] \rightarrow [r]$ is picked uniformly at random from a universal hash family.

- ▶ h satisfies disambiguation requirement for a fixed choice of block, vertices s, t, u, v w.p. $1 - 1/r$.
- ▶ h satisfies requirement for all blocks, vertices s, t, u, v w.p. $1 - 1/n$ for large enough $r = n^{\Theta(1)}$.

Hashing $k \rightarrow k + 1$

$$w_k(P_u) + w_{k+1}(u) \neq w_k(P_v) + w_{k+1}(v)$$

Define

$$w_{k+1}(v) := h(v) \text{ for } v \in L_{k+1},$$

where $h : [n] \rightarrow [r]$ is picked uniformly at random from a universal hash family.

- ▶ h satisfies disambiguation requirement for a fixed choice of block, vertices s, t, u, v w.p. $1 - 1/r$.
- ▶ h satisfies requirement for all blocks, vertices s, t, u, v w.p. $1 - 1/n$ for large enough $r = n^{\Theta(1)}$.
- ▶ Parameters:

$$R_{k+1} = R_k + O(\log n) \Rightarrow R_k = O(k \log n).$$

$$W_{k+1} = W_k + n^{O(1)} \Rightarrow W_k = k \cdot n^{O(1)}.$$

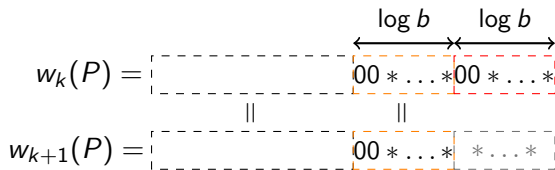
$$R + \log W = O(\log^2 n).$$

Shifting and Hashing $k \rightarrow k'$

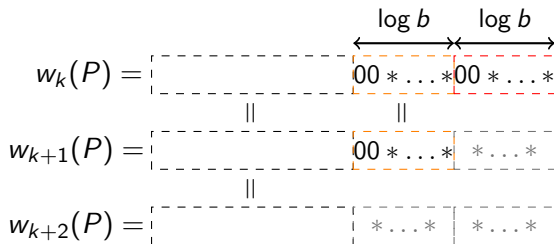
$$w_k(P) = \left[\dots \left[00 * \dots * \right] \left[00 * \dots * \right] \right]$$

The diagram illustrates the structure of the word $w_k(P)$. It is represented as a sequence of digits enclosed in dashed boxes. The rightmost part of the word is divided into two segments, each of length $\log b$, as indicated by the arrows above. The first segment is highlighted with an orange dashed border and contains the digits "00 * ... *". The second segment is highlighted with a red dashed border and also contains the digits "00 * ... *".

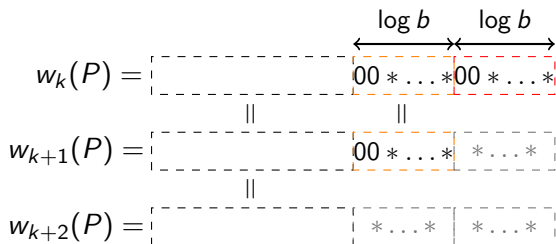
Shifting and Hashing $k \rightarrow k'$



Shifting and Hashing $k \rightarrow k'$



Shifting and Hashing $k \rightarrow k'$

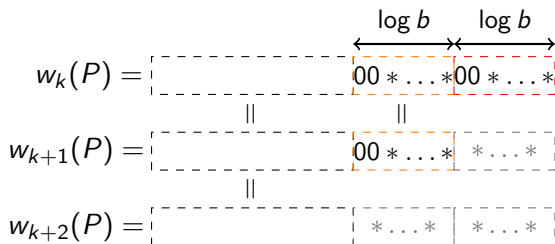


- ▶ For $i \in [k' - k]$,

$$w_{k+i}(v) := h(v) \cdot b^{i-1} \text{ for } v \in L_{k+i},$$

where $b = n^{\Theta(1)}$ and h is picked from a universal hash family.

Shifting and Hashing $k \rightarrow k'$



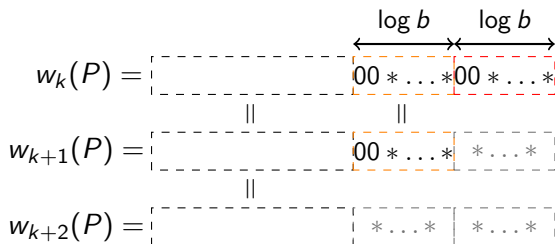
- ▶ For $i \in [k' - k]$,

$$w_{k+i}(v) := h(v) \cdot b^{i-1} \text{ for } v \in L_{k+i},$$

where $b = n^{\Theta(1)}$ and h is picked from a universal hash family.

- ▶ $R_{k'} = R_k + O(\log n)$.
 $W_{k'} = W_k + n^{O(k'-k)}$.

Shifting and Hashing $k \rightarrow k'$



- ▶ For $i \in [k' - k]$,

$$w_{k+i}(v) := h(v) \cdot b^{i-1} \text{ for } v \in L_{k+i},$$

where $b = n^{\Theta(1)}$ and h is picked from a universal hash family.

- ▶ $R_{k'} = R_k + O(\log n)$.
 $W_{k'} = W_k + n^{O(k'-k)}$.
- ▶ Repeating with $\sqrt{\log d}$ hash functions and $k' - k = \sqrt{\log d}$ gives $R + \log W = O(\log n \sqrt{\log d}) = O(\log^{3/2} n)$.

Proof of Theorem 1

Theorem 1

$NL \subseteq UTISP(\text{poly}(n), O(\log^{3/2} n))$.

Proof of Theorem 1

Theorem 1

$NL \subseteq UTISP(\text{poly}(n), O(\log^{3/2} n))$.

- ▶ Running through all possible choices of hash functions gives $NL \subseteq USPACE(O(\log^{3/2} n))$.

Proof of Theorem 1

Theorem 1

$NL \subseteq UTISP(\text{poly}(n), O(\log^{3/2} n))$.

- ▶ Running through all possible choices of hash functions gives $NL \subseteq USPACE(O(\log^{3/2} n))$.
- ▶ Reduce running time to $\text{poly}(n)$ by picking good hash functions one at a time.

Recap of main results

Theorem 1: $NL \subseteq UTISP(\text{poly}(n), O(\log^{3/2} n))$.

Theorem 2: $NL \subseteq L/\text{poly}$ if either of the following hold:

- ▶ there is a logspace *pruning* for REACH.
 - ▶ there is a logspace algorithm that produces a polynomially bounded weight assignment w and a number that is equal to the weight of min-weight s - t path if there is a s - t path.
- ▶ Similar results for LogCFL.

Recap of main results

Theorem 1: $NL \subseteq UTISP(\text{poly}(n), O(\log^{3/2} n))$.

Theorem 2: $NL \subseteq L/\text{poly}$ if either of the following hold:

- ▶ there is a logspace *pruning* for REACH.
 - ▶ there is a logspace algorithm that produces a polynomially bounded weight assignment w and a number that is equal to the weight of min-weight s - t path if there is a s - t path.
- ▶ Similar results for LogCFL.
- ▶ Open questions.

Thank You!

Questions?