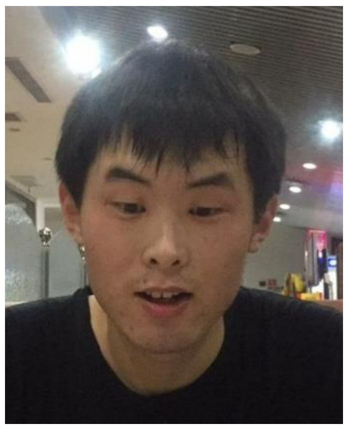
Tight Dynamic Problem Lower Bounds from Generalized BMM and OMv



Ce Jin
MIT

Yinzhan Xu
MIT

STOC 2022

Dynamic problems

- Maintain some data D (graphs, sequences, …)
- Support small updates to D (insertions, deletions, …)
- Answer queries about D (connectivity?...)

- Best possible update time and query time?

Dynamic problems

- Maintain some data D (graphs, sequences, …)
- Support small <u>updates</u> to D (insertions, deletions, …)
- Answer <u>queries</u> about D (connectivity?...)

- Best possible update time and query time?
- Unconditional Lower Bounds are stuck at $\mathrm{polylog}(n)$ ☹

Dynamic problems

- Maintain some data D (graphs, sequences, …)
- Support small <u>updates</u> to D (insertions, deletions, …)
- Answer <u>queries</u> about D (connectivity?...)

- Best possible update time and query time?
- Unconditional Lower Bounds are stuck at $\mathrm{polylog}(n)$ ☹
- **Higher LBs from <u>Fine-Grained Conjectures</u>!**
  - A long line of work
    - [Pătraşcu STOC'10]
    - [Abboud and Vassilevska Williams FOCS'14]
    - [Henzinger, Krinninger, Nanongkai, and Saranurak STOC'15]

    …

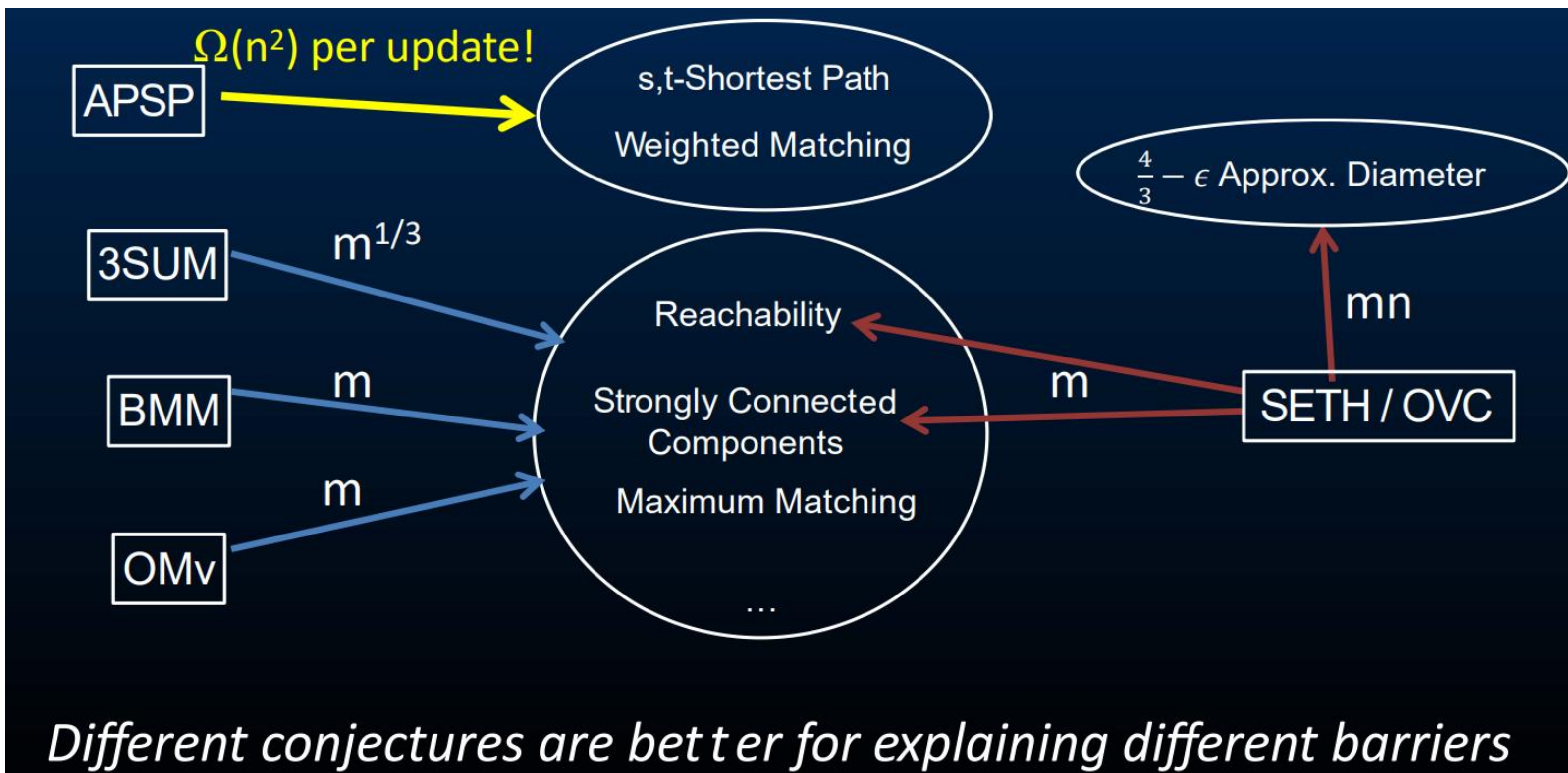
Dynamic problems

- Maintain some data D (graphs, sequences, ...)
- Support small <u>updates</u> to D (insertions, deletions, ...)
- Answer <u>queries</u> about D (connectivity?...)

- Best possible update time and query time?

polylog(n) ☹

s!

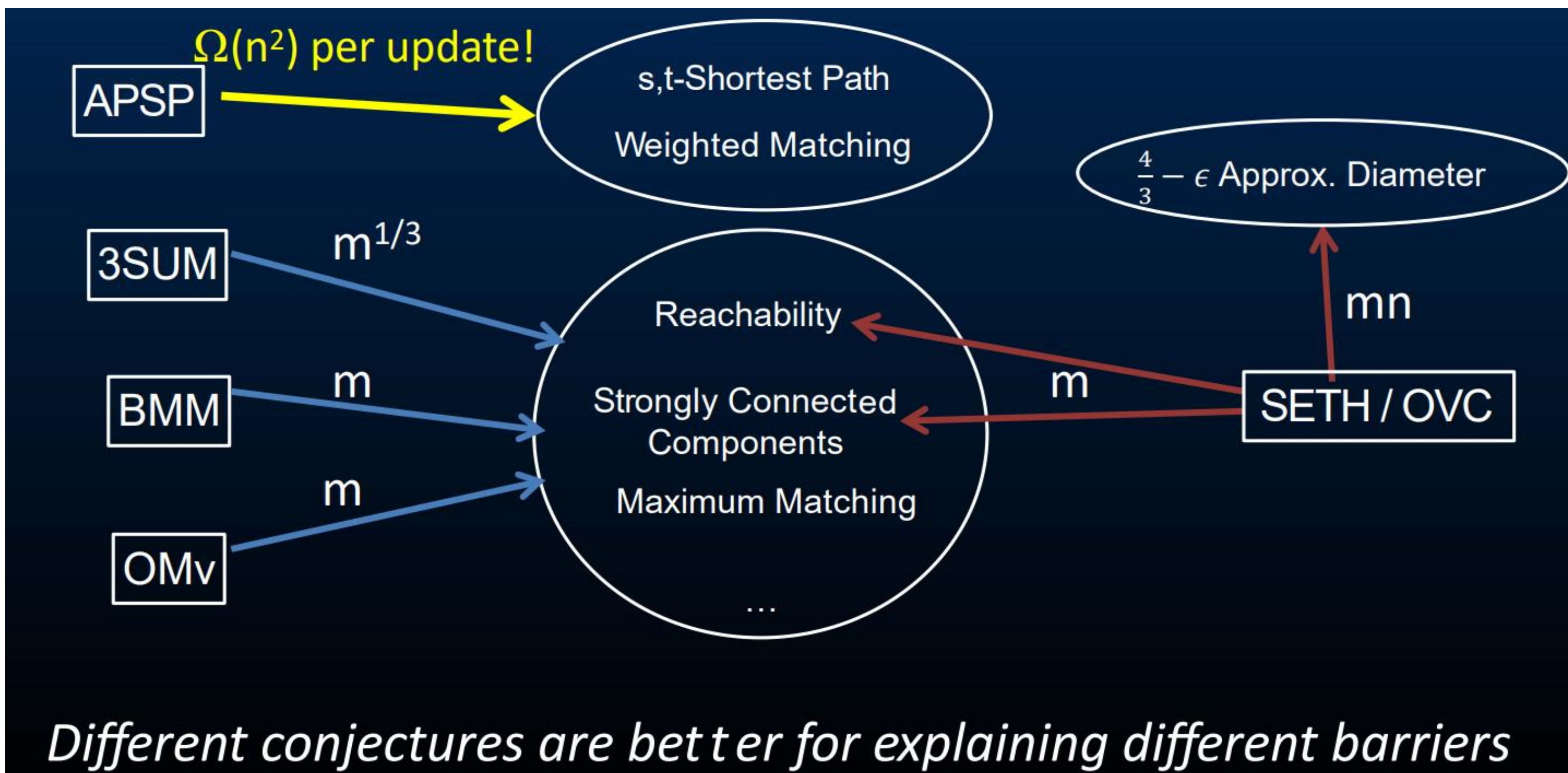


(from Virginia's lecture notes)

Dynamic problems

- Maintain some data D (graphs, sequences, …)
- Support small <u>updates</u> to D (insertions, deletions, …)
- Answer <u>queries</u> about D (connectivity?...)

- Best possible update time and query time?



(from Virginia's lecture notes)

This work:
A collection of new Lower Bounds based on generalized versions of **BMM** and **OMv**

Combinatorial BMM hypothesis

- Input: $n \times n$ Boolean matrices A B
- Output: Boolean matrix product AB

Combinatorial BMM hypothesis

- Input: $n \times n$ Boolean matrices A B
- Output: Boolean matrix product AB
- Conjecture: No $O(n^{3-\varepsilon})$-time "combinatorial" algorithm exists

Combinatorial BMM hypothesis

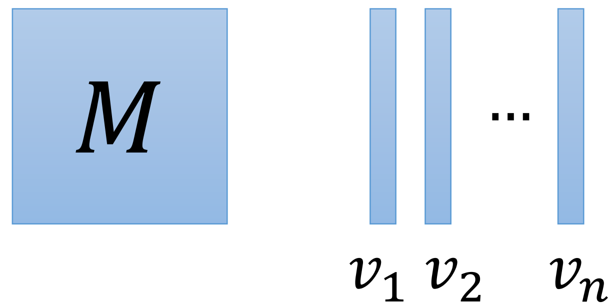
- Input: $n \times n$ Boolean matrices A B

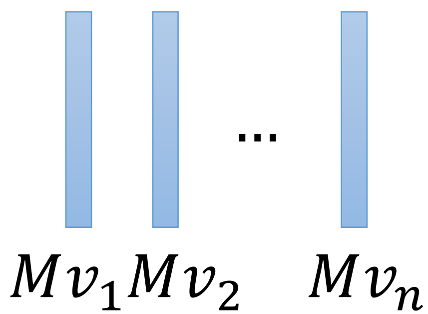
- Output: Boolean matrix product AB

- Conjecture: No $O(n^{3-\varepsilon})$-time "combinatorial" algorithm exists
  - Current best: $\boldsymbol{n^3}(\log\log n)^{O(1)}/(\boldsymbol{\log n})^{\boldsymbol{4}}$ [Yu'15]

Algorithms avoiding Fast Matrix Multiplication (e.g. Strassen's)

OMv hypothesis [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

- Input: Boolean 

- Output: 

(in an online fashion)

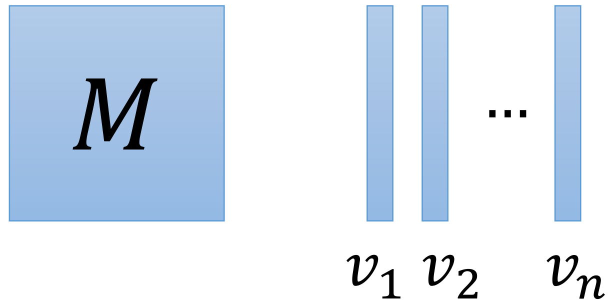
OMv hypothesis [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

- Input: Boolean M $v_1\ v_2\ \ldots\ v_n$
- Output: $Mv_1\ Mv_2\ \ldots\ Mv_n$ (in an online fashion)
- Conjecture: No $O(n^{3-\varepsilon})$-time algorithm exists

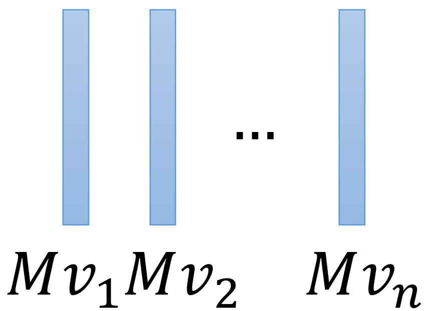
Not only "combinatorial"

OMv hypothesis [Henzinger-Krinninger-Nanongkai-Saranurak STOC'15]

- Input: Boolean



- Output:



(in an online fashion)

Not only "combinatorial"

- Conjecture: No $O(n^{3-\varepsilon})$-time algorithm exists
  - Current best: $n^3/2^{\Omega\left(\sqrt{\log n}\right)}$ time [Larsen-Williams'17]

Dynamic Range-Mode Query

- Maintain an integer array $A[1], A[2], \ldots, A[n]$
- Support Insertions and Deletions
- Query l, r: what is the most frequent element in $A[l], A[l+1], \ldots, A[r]$?
  (breaking ties arbitrarily)

Dynamic Range-Mode Query

- Maintain an integer array $A[1], A[2], \ldots, A[n]$
- Support Insertions and Deletions
- Query l, r: what is the <u>most frequent element</u> in $A[l], A[l+1], \ldots, A[r]$?
  (breaking ties arbitrarily)

- Combinatorial algorithms (folklore):
  - Dynamic range-mode: $\boldsymbol{\widetilde{O}(n^{2/3})}$ query time & update time

Dynamic Range-Mode Query

- Maintain an integer array $A[1], A[2], \ldots, A[n]$
- Support Insertions and Deletions
- Query l, r: what is the <u>most frequent element</u> in $A[l], A[l+1], \ldots, A[r]$?
  (breaking ties arbitrarily)

- Combinatorial algorithms (folklore):
  - Dynamic range-mode**:** $\boldsymbol{\widetilde{O}(n^{2/3})}$ query time & update time
  - Static range-mode: $\boldsymbol{\widetilde{O}(n^{0.5})}$ query time (after $\widetilde{O}(n^{1.5})$-time preprocessing)

Dynamic Range-Mode Query

- Maintain an integer array $A[1], A[2], \dots, A[n]$
- Support Insertions and Deletions
- Query l, r: what is the <u>most frequent element</u> in $A[l], A[l+1], \dots, A[r]$?
  (breaking ties arbitrarily)

- Combinatorial algorithms (folklore):
  - Dynamic range-mode: $\boldsymbol{\widetilde{O}(n^{2/3})}$ query time & update time
  - Static range-mode: $\boldsymbol{\widetilde{O}(n^{0.5})}$ query time (after $\widetilde{O}(n^{1.5})$-time preprocessing)
- (slight $n^{\Omega(1)}$ improvements using FMM are known [SX'20, VX'20, GPVX'21])

Dynamic Range-Mode Query

- Maintain an integer array $A[1], A[2], \ldots, A[n]$
- Support Insertions and Deletions
- Query l, r: what is the most frequent element in $A[l], A[l+1], \ldots, A[r]$?
  (breaking ties arbitrarily)

- Combinatorial algorithms (folklore):
  - Dynamic range-mode: $\boldsymbol{\tilde{O}(n^{2/3})}$ query time & update time
  - Static range-mode: $\boldsymbol{\tilde{O}(n^{0.5})}$ query time (after $\tilde{O}(n^{1.5})$-time preprocessing)
- (slight $n^{\Omega(1)}$ improvements using FMM are known [SX'20, VX'20, GPVX'21])
- Can these combinatorial algorithms be improved?

Dynamic Range-Mode Query: Lower Bounds

- *Static* Range-Mode: Tight combinatorial LB
  - Under BMM, no combinatorial algorithm can achieve $\boldsymbol{n^{0.5-\varepsilon}}$ query time and $n^{1.5-\varepsilon}$ preprocessing time [Chan-Durocher-Larsen-Morrison-Wilkinson’14]

Dynamic Range-Mode Query: Lower Bounds

- *Static* Range-Mode: Tight combinatorial LB
  - Under BMM, no combinatorial algorithm can achieve $\boldsymbol{n^{0.5-\varepsilon}}$ query time and $n^{1.5-\varepsilon}$ preprocessing time [Chan-Durocher-Larsen-Morrison-Wilkinson'14]
- *Dynamic* Range-Mode LB (NOT Tight ☹):
  - Under OMv, no algorithm can achieve $\boldsymbol{n^{0.5-\varepsilon}}$ query & update time and $\mathrm{poly}(n)$ preprocessing time.

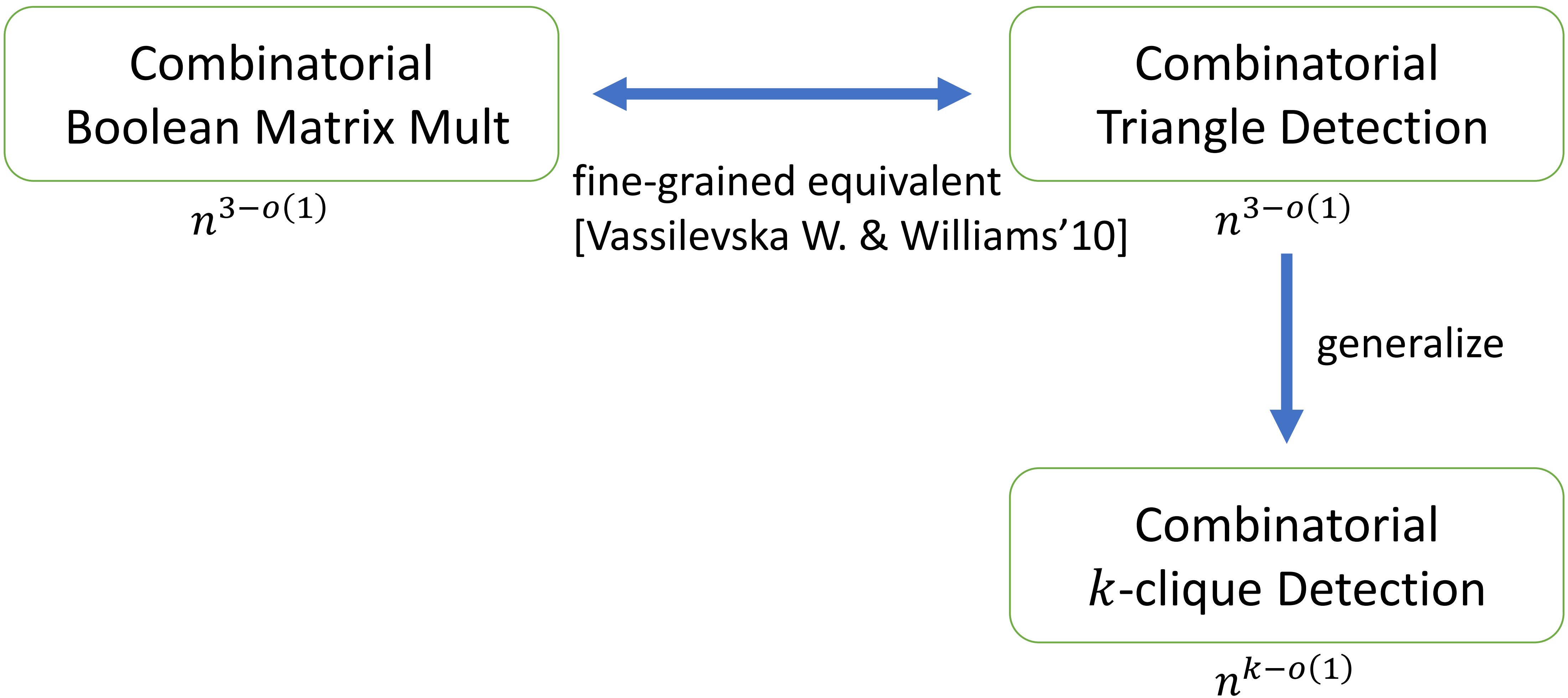
Dynamic Range-Mode Query: Lower Bounds

- *Static* Range-Mode: Tight combinatorial LB
  - Under BMM, no combinatorial algorithm can achieve $\boldsymbol{n^{0.5-\varepsilon}}$ query time and $n^{1.5-\varepsilon}$ preprocessing time [Chan-Durocher-Larsen-Morrison-Wilkinson'14]
- *Dynamic* Range-Mode LB (NOT Tight ☹):
  - Under OMv, no algorithm can achieve $\boldsymbol{n^{0.5-\varepsilon}}$ query & update time and $\mathrm{poly}(n)$ preprocessing time.
- **Our result (tight!)**: No combinatorial algorithm can solve ***Dynamic*** Range-Mode in $\boldsymbol{n^{2/3-\varepsilon}}$ query & update time and $\mathrm{poly}(n)$ preprocessing time
  - Under **combinatorial 4-clique hypothesis**

Dynamic Range-Mode Query: Lower Bounds

- *Static* Range-Mode: Tight combinatorial LB
  - Under BMM, no combinatorial algorithm can achieve $\boldsymbol{n^{0.5-\varepsilon}}$ query time and $n^{1.5-\varepsilon}$ preprocessing time [Chan-Durocher-Larsen-Morrison-Wilkinson'14]
- *Dynamic* Range-Mode LB (NOT Tight ☹):
  - Under OMv, no algorithm can achieve $\boldsymbol{n^{0.5-\varepsilon}}$ query & update time and $\mathrm{poly}(n)$ preprocessing time.
- **Our result (tight!)**: No combinatorial algorithm can solve ***Dynamic*** Range-Mode in $\boldsymbol{n^{2/3-\varepsilon}}$ query & update time and $\mathrm{poly}(n)$ preprocessing time
  - Under **combinatorial 4-clique hypothesis**

Combinatorial k-clique hypothesis: No combinatorial algorithm can detect k-clique in an n-node graph in $O\left(n^{k-\varepsilon}\right)$ time



Combinatorial k-clique hypothesis: No combinatorial algorithm can detect k-clique in an n-node graph in $O\left(n^{k-\varepsilon}\right)$ time

A proof template

Static LB from
$(k-1)$-clique hypothesis

Use dynamic operations to efficiently enumerate the extra k-th node

Dynamic LB from
k-clique hypothesis

Chan et al.'14: *Static* Range Mode requires $n^{0.5-o(1)}$ time (from combinatorial 3-clique)

Our result: *Dynamic* Range Mode requires $n^{2/3-o(1)}$ time (from combinatorial 4-clique)

A proof template

Static LB from
$(k-1)$-clique hypothesis

Use dynamic operations to efficiently enumerate the extra k-th node

Dynamic LB from
k-clique hypothesis

Chan et al.'14: *Static* Range Mode requires $n^{0.5-o(1)}$ time (from combinatorial 3-clique)

Our result: *Dynamic* Range Mode requires $n^{2/3-o(1)}$ time (from combinatorial 4-clique)

Tight combinatorial LBs for more dynamic problems:

- Dynamic 2D Orthogonal Range Color Counting $n^{2/3-o(1)}$ time ($k = 4$)
- Dynamic d-Dimensional Orthogonal Range Mode $n^{1-\frac{1}{2d+1}-o(1)}$ time ($k = 2d + 2$)
- Dynamic 2-Pattern Document Retrieval $n^{2/3-o(1)}$ time ($k = 4$)
- ...

A proof template

Static LB from
$(k-1)$-clique hypothesis

Use dynamic operations to efficiently enumerate the extra k-th node

Dynamic LB from
k-clique hypothesis

Chan et al.'14: *Static* Range Mode requires $n^{0.5-o(1)}$ time (from combinatorial 3-clique)

Our result: *Dynamic* Range Mode requires $n^{2/3-o(1)}$ time (from combinatorial 4-clique)

Main takeaway:
(Combinatorial) k-clique hypothesis is useful for dynamic lower bounds!

Previous dynamic LBs mostly used $k = 3$ (BMM).
(exception: [Gutenberg, Vassilevska Williams, and Wein'20] reduction from 4-clique to dynamic shortest path)

Dynamic Subgraph Connectivity

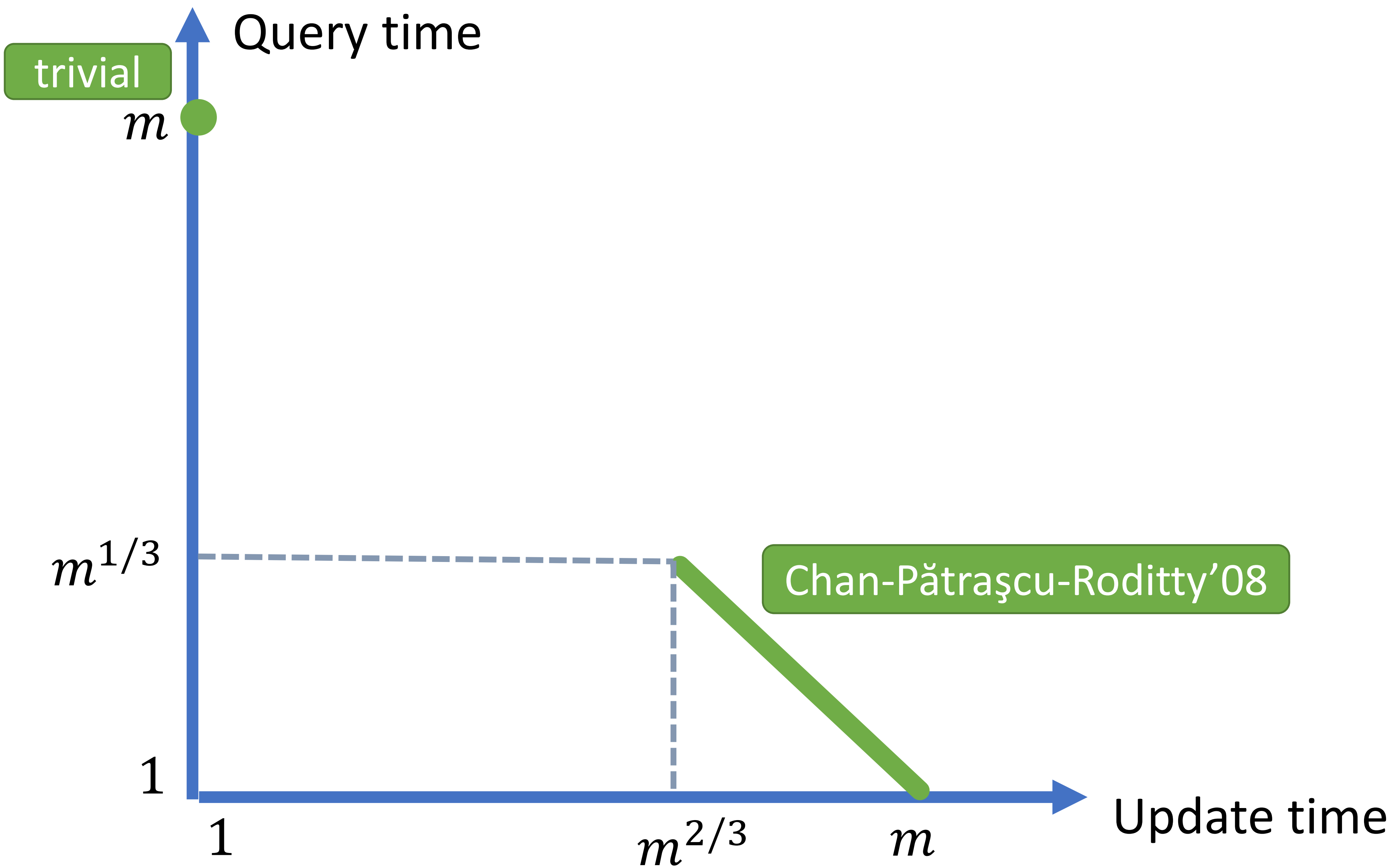
- Preprocess a static undirected graph G with m edges
- Maintain a dynamic <u>vertex subset S</u> ("on" vertices)
  - Turn on u: $S \leftarrow S \cup \{u\}$
  - Turn off u: $S \leftarrow S \setminus \{u\}$
  - Query u, v: are u and v connected in the induced subgraph $G[S]$?

Dynamic Subgraph Connectivity

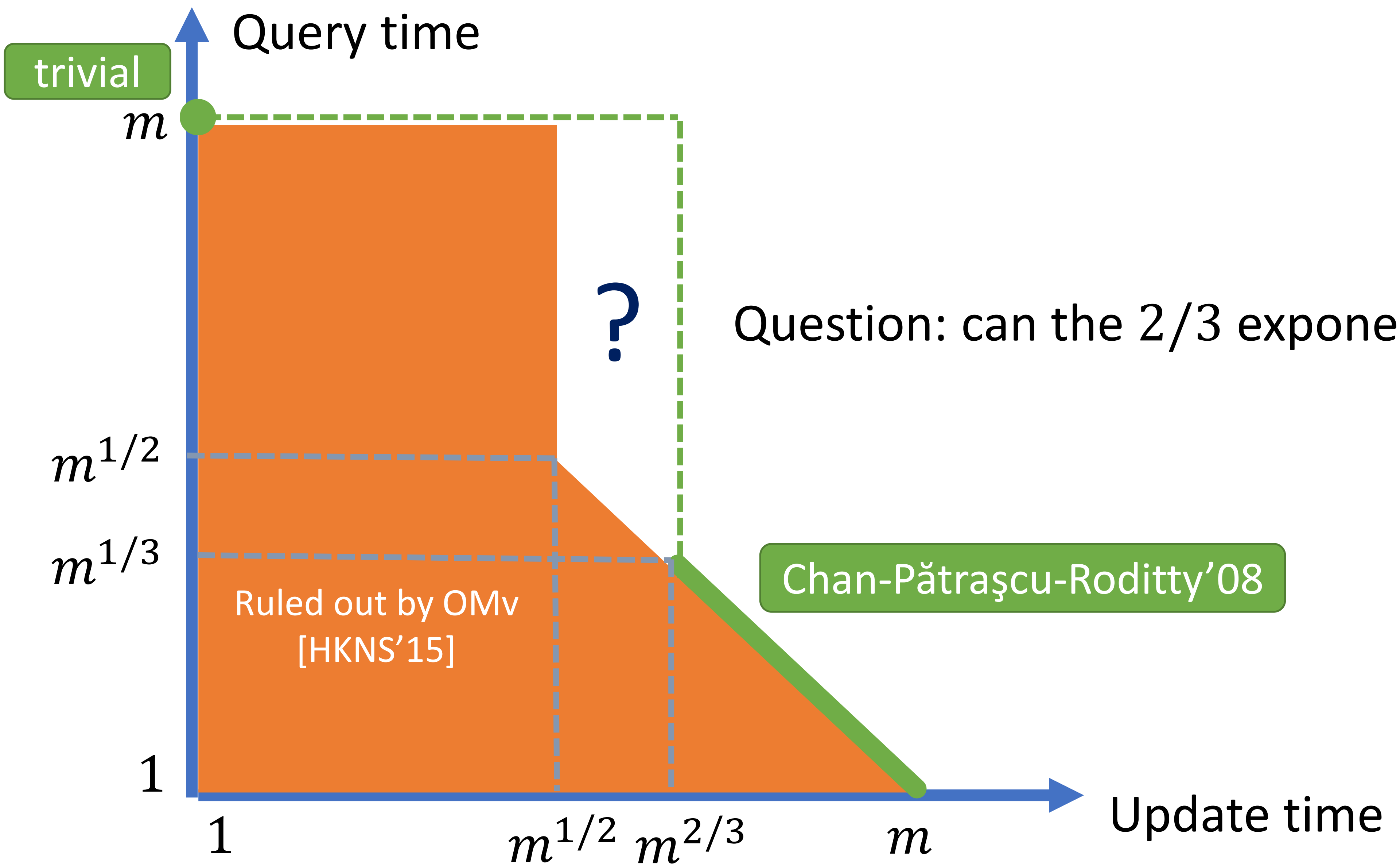
- Preprocess a static undirected graph G with m edges
- Maintain a dynamic vertex subset S ("on" vertices)
  - Turn on u: $S \leftarrow S \cup \{u\}$
  - Turn off u: $S \leftarrow S \setminus \{u\}$
  - Query u, v: are u and v connected in the induced subgraph $G[S]$?

- Combinatorial algorithm by Chan, Pătraşcu, and Roditty (FOCS'08) in
  - $\boldsymbol{\tilde{O}(m^{2/3})}$ update time (amortized)
  - $\tilde{O}(m^{1/3})$ query time
  - ($\tilde{O}(m^{4/3})$ preprocessing time)
- Can the $2/3$ exponent be improved?

Dynamic Subgraph Connectivity

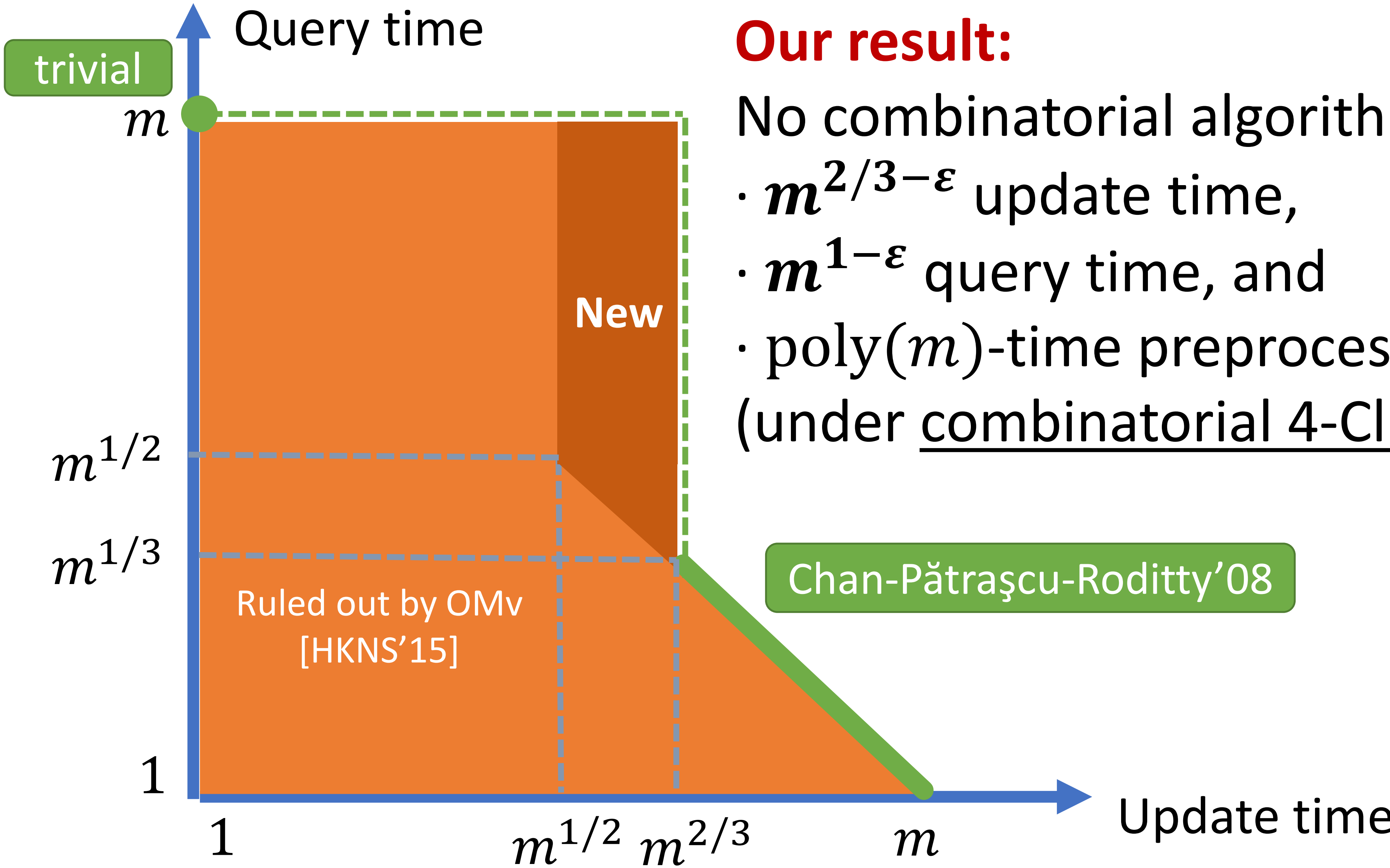


Dynamic Subgraph Connectivity



Question: can the 2/3 exponent be improved?

Dynamic Subgraph Connectivity



Our result:

No combinatorial algorithm can achieve

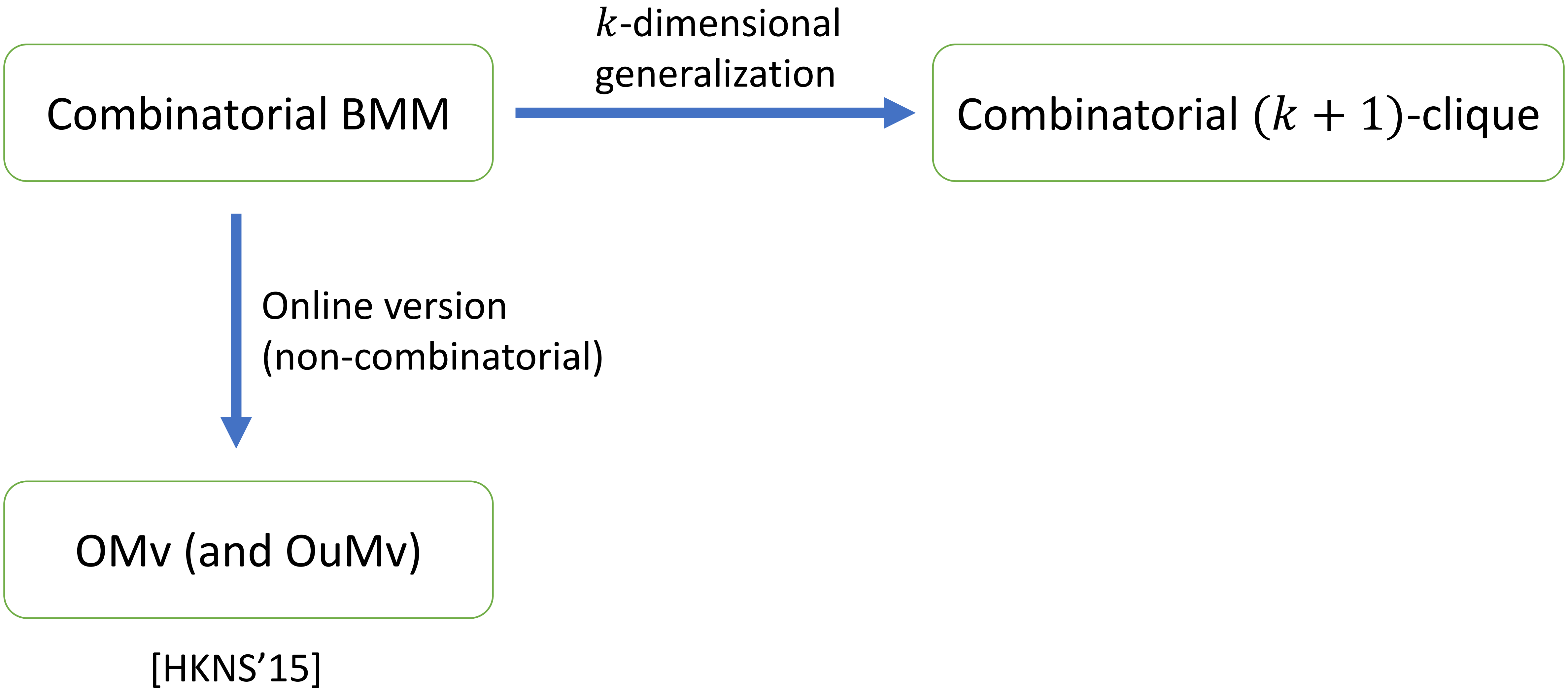
· $\boldsymbol{m^{2/3-\varepsilon}}$ update time,

· $\boldsymbol{m^{1-\varepsilon}}$ query time, and

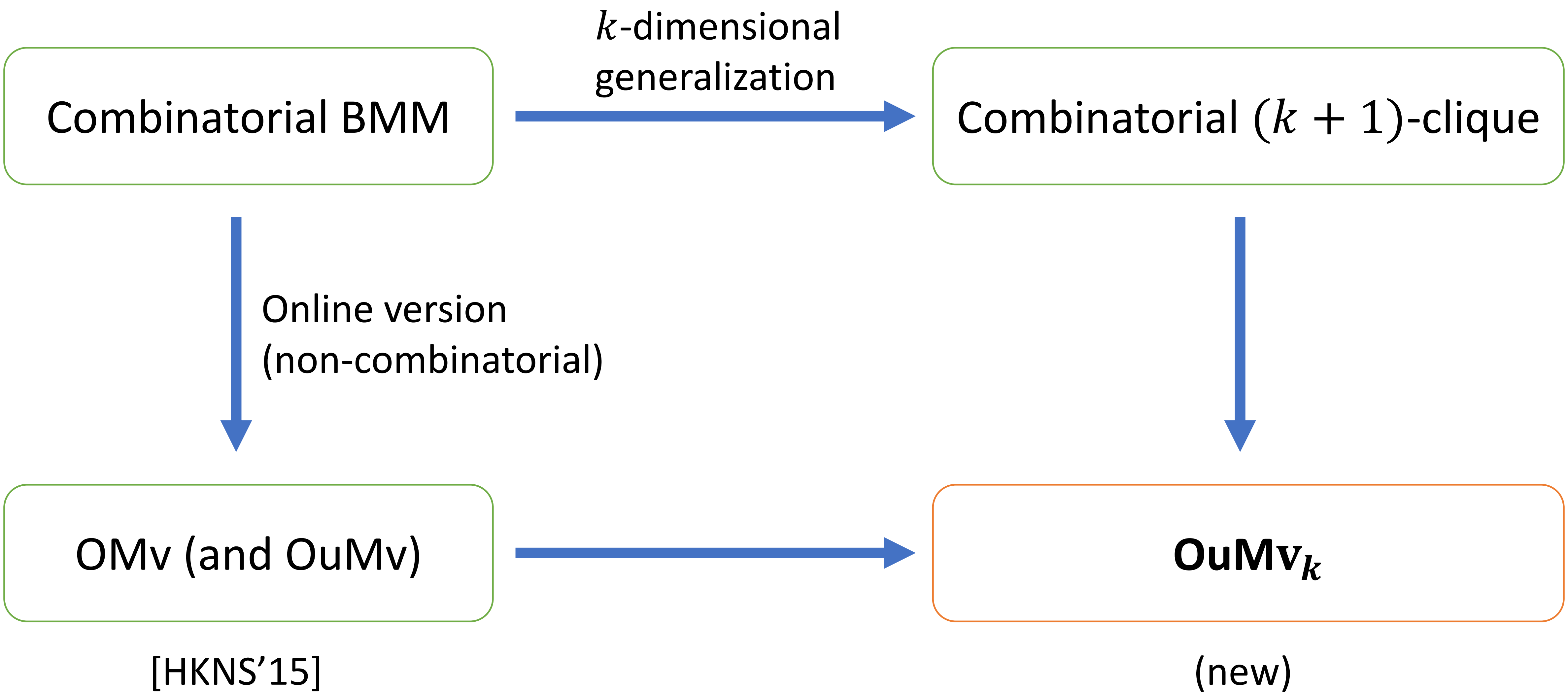
· $\mathrm{poly}(m)$-time preprocessing time

(under <u>combinatorial 4-Clique hypothesis</u>)

A new fine-grained conjecture



A new fine-grained conjecture



OuMv_k hypothesis

- Pre-process a subset $M \subseteq \{1,2,\ldots,n\}^k$
- Answer n online queries:
  - Given k sets $U^{(1)}, U^{(2)}, \ldots, U^{(k)} \subseteq \{1,2,\ldots,n\}$,
  - Is $\left(U^{(1)} \times U^{(2)} \times \cdots \times U^{(k)}\right) \cap M$ non-empty?
- Conjecture: No $O\left(n^{1+k-\varepsilon}\right)$-time algorithm exists

Not only “combinatorial”

- Naturally generalizes OuMv [HKNS’15] (which is OuMv_2)

$\text{OuM}\mathbf{v}_k$ hypothesis

- Pre-process a subset $M \subseteq \{1,2,\ldots,n\}^k$
- Answer n online queries:
  - Given k sets $U^{(1)}, U^{(2)}, \ldots, U^{(k)} \subseteq \{1,2,\ldots,n\}$,
  - Is $\left(U^{(1)} \times U^{(2)} \times \cdots \times U^{(k)}\right) \cap M$ non-empty?
- Conjecture: No $O\left(n^{1+k-\varepsilon}\right)$-time algorithm exists

Not only "combinatorial"

- Naturally generalizes OuMv [HKNS'15] (which is OuMv_2)
- Useful for dynamic geometry problems in $\mathbf{R}^k$
  - Obtain higher lower bounds as dimension k increases

Dynamic Skyline (Maximal) Points Counting

- Maintain a set P of n points in $\mathbf{R}^d$
- Insertion: $P \leftarrow P \cup \{x\}$
- Deletion: $P \leftarrow P \setminus \{x\}$
- Query: how many "skyline points" does P have?
  - $x \in P$ is a "skyline point"("maximal point") iff no other $y \in P$ dominates x (i.e. $y_i \geq x_i$ for all $i = 1,2, \dots, d$)

Dynamic Skyline (Maximal) Points Counting

- Maintain a set P of n points in $\mathbf{R}^d$
- Insertion: $P \leftarrow P \cup \{x\}$
- Deletion: $P \leftarrow P \setminus \{x\}$
- Query: how many "skyline points" does P have?
  - $x \in P$ is a "skyline point"("maximal point") iff no other $y \in P$ dominates x (i.e. $y_i \geq x_i$ for all $i = 1,2, \ldots, d$)

Chan'03 (adapted): A *semi-online* algorithm in $\mathbf{R}^{2k-1}$ with $\tilde{O}\left(n^{1-1/k}\right)$ update time.

Semi-online: When x is inserted, we are told when x will be deleted in the future

Dynamic Skyline (Maximal) Points Counting

- Maintain a set P of n points in $\mathbf{R}^d$
- Insertion: $P \leftarrow P \cup \{x\}$
- Deletion: $P \leftarrow P \setminus \{x\}$
- Query: how many "skyline points" does P have?
  - $x \in P$ is a "skyline point"("maximal point") iff no other $y \in P$ dominates x (i.e. $y_i \geq x_i$ for all $i = 1,2, \ldots, d$)

Chan'03 (adapted): A *semi-online* algorithm in $\mathbf{R}^{2k-1}$ with $\tilde{O}\left(n^{1-1/k}\right)$ update time.

Semi-online: When x is inserted, we are told when x will be deleted in the future

Our result: this is <u>tight</u> under **OuMv$_k$ hypothesis**

(The $k = 2$ case based on OMv was recently independently proved by Dallant & Iacono (2021))

Conclusion

- We used combinatorial k-clique hypothesis and OuMv_k hypothesis to prove tight fine-grained lower bounds for dynamic problems.

Open questions:

- Can Dynamic Subgraph Connectivity have update time better than $m^{2/3}$ using fast matrix multiplication?
- What is the optimal update time for Dynamic Skyline Points Counting in $\mathbf{R}^{2k}$? (semi-online algorithms allowed)

Conclusion

- We used combinatorial k-clique hypothesis and OuMv_k hypothesis to prove tight fine-grained lower bounds for dynamic problems.

Open questions:

- Can Dynamic Subgraph Connectivity have update time better than $m^{2/3}$ using fast matrix multiplication?
- What is the optimal update time for Dynamic Skyline Points Counting in $\mathbf{R}^{2k}$? (semi-online algorithms allowed)

- **Thanks!**