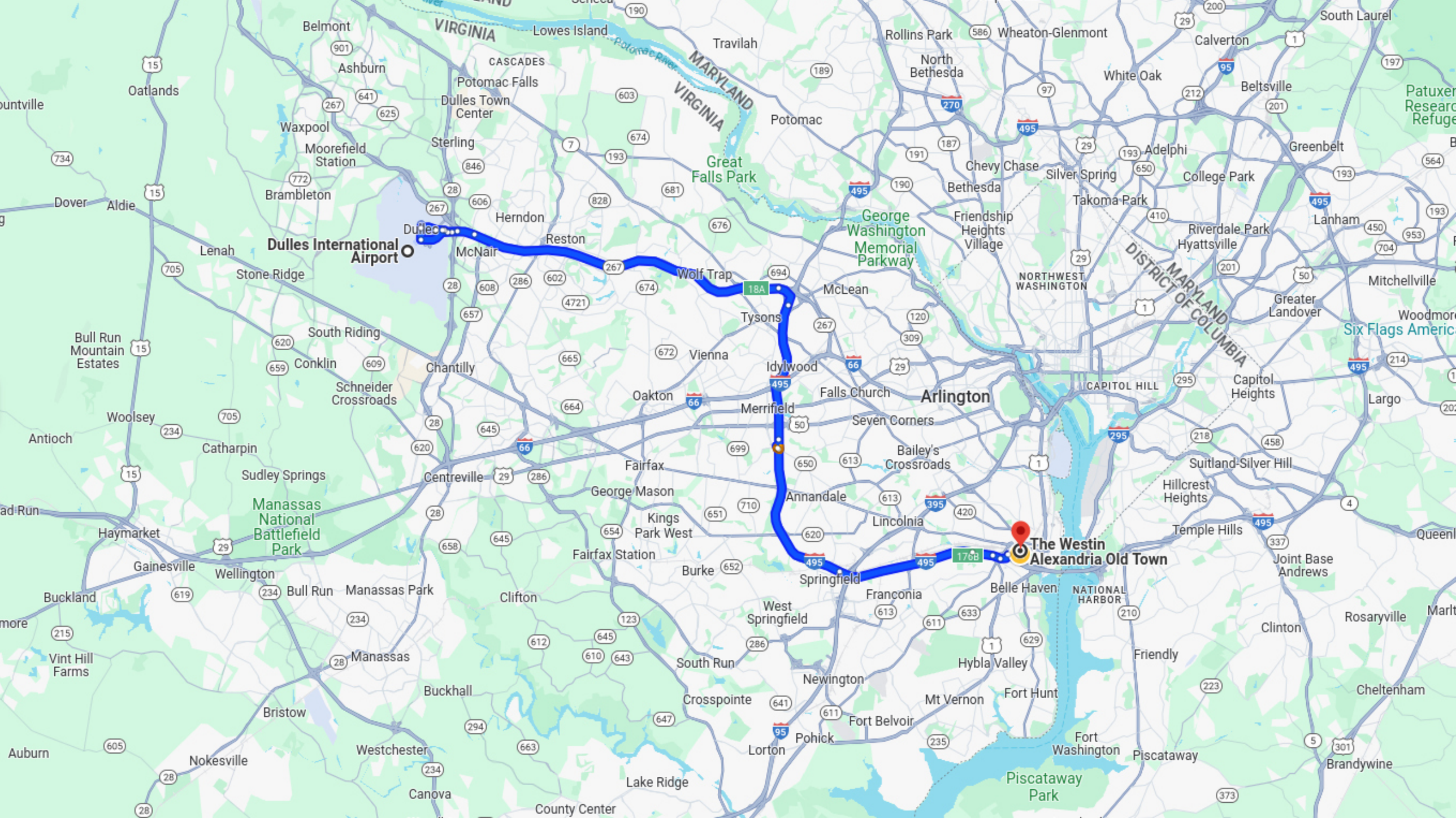
On Dynamic Graph Algorithms with Predictions

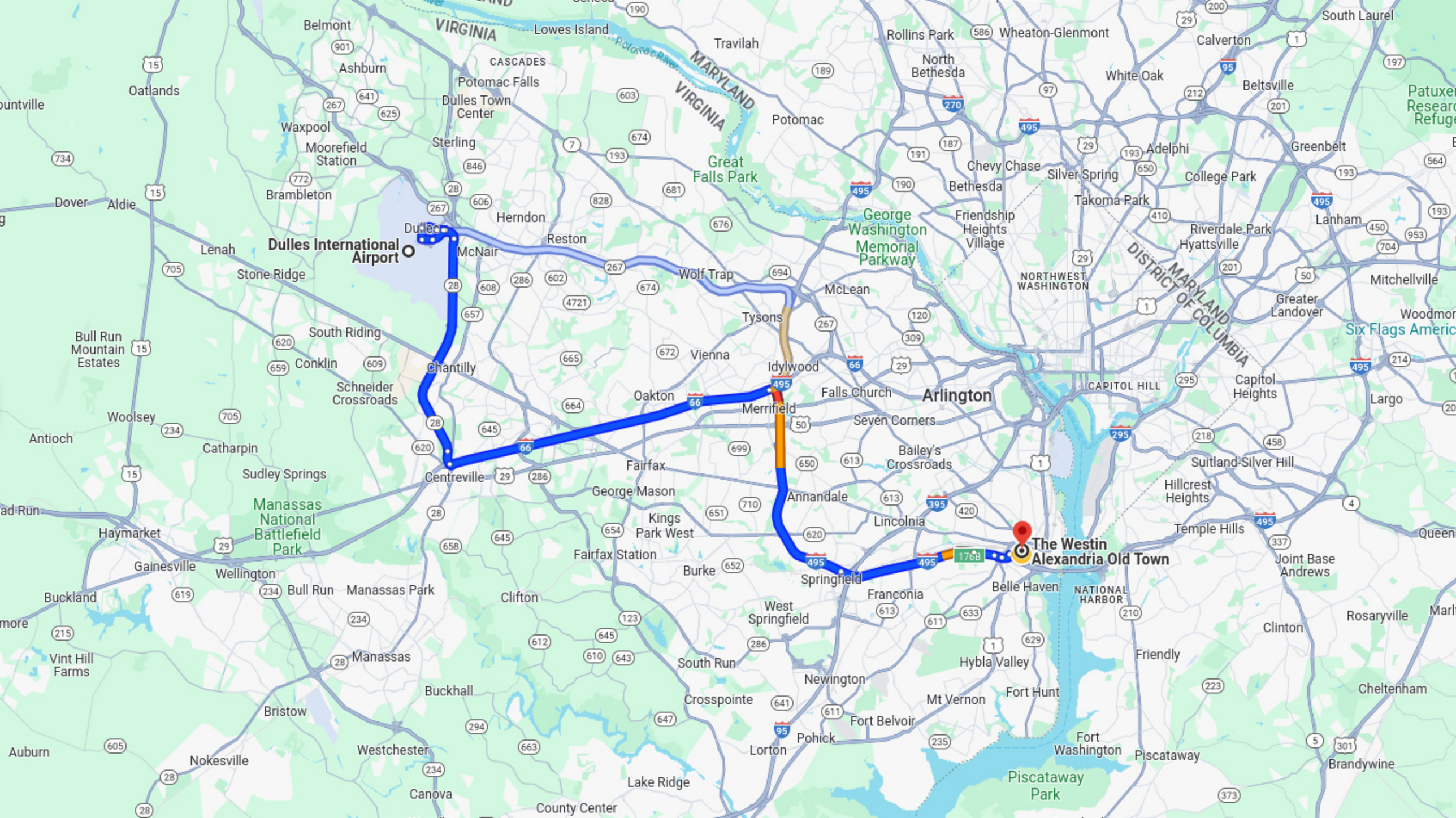
Jan van den Brand
Georgia Tech

Sebastian Forster
University of Salzburg

Yasamin Nazari
VU Amsterdam

Adam Polak
Bocconi University


Dulles International Airport
The Westin Alexandria Old Town
Dulles
McNair
Reston
Herndon
Wolf Trap
McLean
Tysons
Vienna
Idylwood
Falls Church
Arlington
Merrifield
Seven Corners
Bailey's Crossroads
Annandale
Lincolnia
Springfield
Franconia
Belle Haven
Fairfax
Centreville
Chantilly
Oakton
Fairfax Station
Burke
Clifton
Manassas
Manassas Park
Gainesville
Haymarket
Sudley Springs
Catharpin
Bull Run Mountain Estates
South Riding
Conklin
Schneider Crossroads
Woolsey
Antioch
Lenah
Stone Ridge
Dover
Aldie
Oatlands
Ashburn
Belmont
Lowes Island
Potomac Falls
Dulles Town Center
Waxpool
Moorefield Station
Sterling
Brambleton
Great Falls Park
Travilah
Potomac
North Bethesda
Rollins Park
Wheaton-Glenmont
White Oak
Calverton
South Laurel
Beltsville
Greenbelt
Adelphi
Chevy Chase
Silver Spring
Bethesda
Takoma Park
College Park
Friendship Heights Village
George Washington Memorial Parkway
NORTHWEST WASHINGTON
DISTRICT OF COLUMBIA
MARYLAND
Riverdale Park
Hyattsville
Lanham
Greater Landover
Mitchellville
Capitol Heights
CAPITOL HILL
Largo
Suitland-Silver Hill
Hillcrest Heights
Temple Hills
Joint Base Andrews
NATIONAL HARBOR
Hybla Valley
Mt Vernon
Fort Hunt
Fort Washington
Piscataway
Piscataway Park
Friendly
Clinton
Rosaryville
Cheltenham
Brandywine
Six Flags America
Woodmore
Queenl
Marlt
Patuxent Research Refuge
Fort Belvoir
Pohick
Lorton
Crosspointe
Newington
South Run
West Springfield
Kings Park West
George Mason
Bristow
Buckhall
Nokesville
Westchester
Canova
Lake Ridge
County Center
Auburn
Vint Hill Farms
Buckland
Bull Run
Wellington
Manassas National Battlefield Park
CASCADES
VIRGINIA
MARYLAND
Potomac River


Dulles International Airport
The Westin Alexandria Old Town
Arlington
Centreville
Chantilly
Dulles
McNair
Herndon
Reston
Wolf Trap
Tysons
McLean
Vienna
Idylwood
Falls Church
Merrifield
Oakton
Seven Corners
Bailey's Crossroads
Fairfax
George Mason
Annandale
Lincolnia
Springfield
Franconia
Kings Park West
Fairfax Station
Burke
Clifton
West Springfield
South Run
Newington
Crosspointe
Fort Belvoir
Lorton
Pohick
Mt Vernon
Fort Hunt
Hybla Valley
Belle Haven
Fort Washington
Piscataway
Piscataway Park
NATIONAL HARBOR
CAPITOL HILL
NORTHWEST WASHINGTON
DISTRICT OF COLUMBIA
MARYLAND
VIRGINIA
Great Falls Park
George Washington Memorial Parkway
Potomac River
Lowes Island
Potomac Falls
Dulles Town Center
Sterling
Ashburn
Belmont
Travilah
Potomac
Rollins Park
Wheaton-Glenmont
North Bethesda
White Oak
Calverton
South Laurel
Beltsville
Chevy Chase
Bethesda
Silver Spring
Takoma Park
Friendship Heights Village
Adelphi
Greenbelt
College Park
Riverdale Park
Hyattsville
Lanham
Greater Landover
Mitchellville
Capitol Heights
Largo
Suitland-Silver Hill
Hillcrest Heights
Temple Hills
Joint Base Andrews
Clinton
Friendly
Cheltenham
Rosaryville
Brandywine
Waxpool
Moorefield Station
Brambleton
Oatlands
Dover
Aldie
Lenah
Stone Ridge
Bull Run Mountain Estates
South Riding
Conklin
Schneider Crossroads
Woolsey
Antioch
Catharpin
Sudley Springs
Manassas National Battlefield Park
Haymarket
Gainesville
Wellington
Bull Run
Manassas Park
Buckland
Vint Hill Farms
Manassas
Buckhall
Bristow
Auburn
Nokesville
Westchester
Canova
Lake Ridge
County Center

Dynamic algorithms

Data structures, but for fancier queries

Operations:

- **updates**
  e.g., add edge, delete edge
- **queries**
  e.g., find a path from source to v,
  is the graph strongly connected?

Dynamic algorithms

Data structures, but for fancier queries

Operations:

- **updates**
  e.g., add edge, delete edge
- **queries**
  e.g., find a path from source to v,
  is the graph strongly connected?

recompute from scratch

sublinear update time

polylog update time

Classical algorithms

- worst-case guarantees
- overly pessimistic

Classical algorithms

- worst-case guarantees
- overly pessimistic

Machine learning

- powerful for typical inputs
- no guarantees, can go crazy

Classical algorithms

- worst-case guarantees
- overly pessimistic

Machine learning

- powerful for typical inputs
- no guarantees, can go crazy

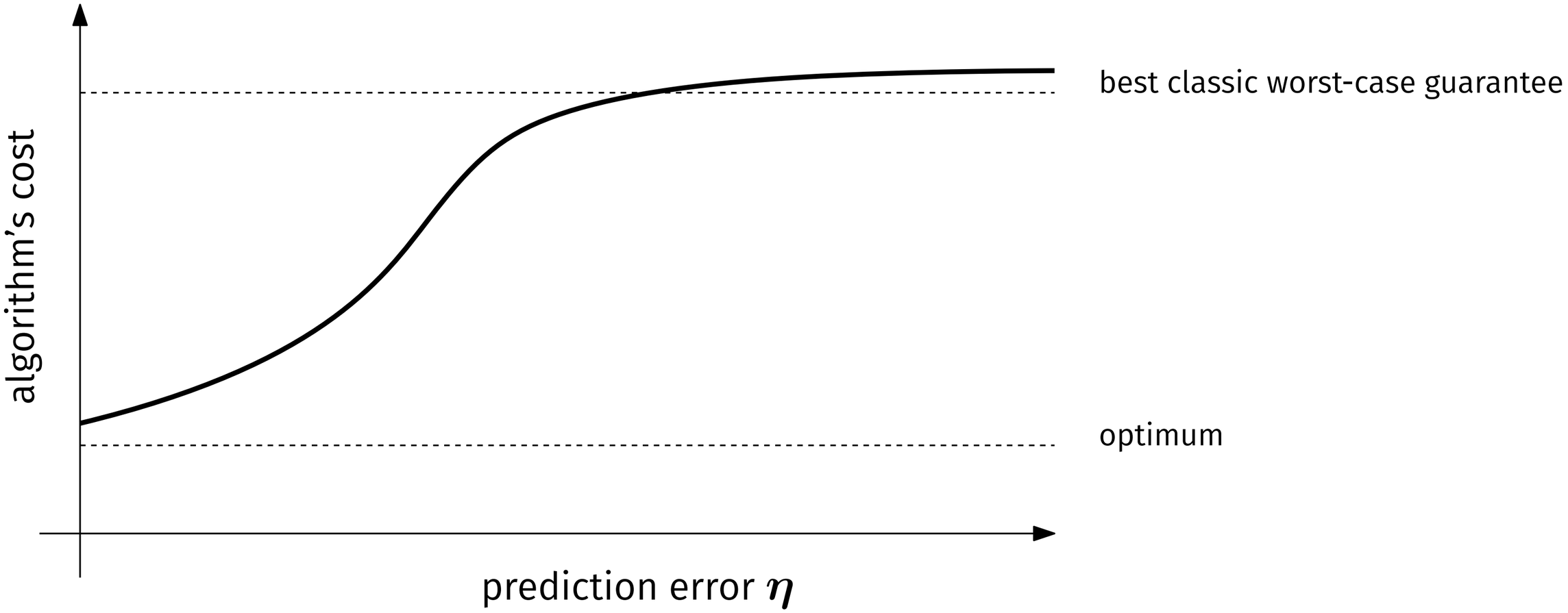
Best of both worlds: **learning-augmented algorithms** (algorithms with predictions)

Learning-augmented algorithms

Input + black-box **predictions** (possibly inacurate, with some **error** η)

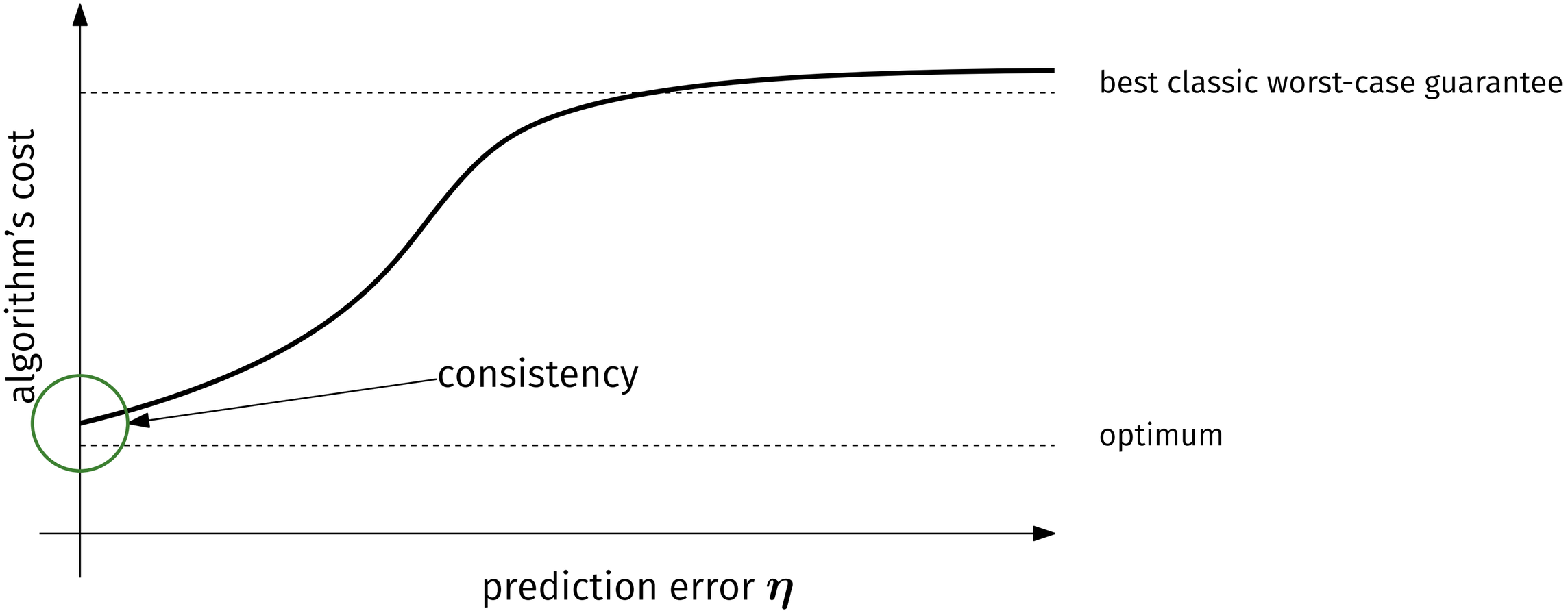
Learning-augmented algorithms

Input + black-box **predictions** (possibly inacurate, with some **error** η)



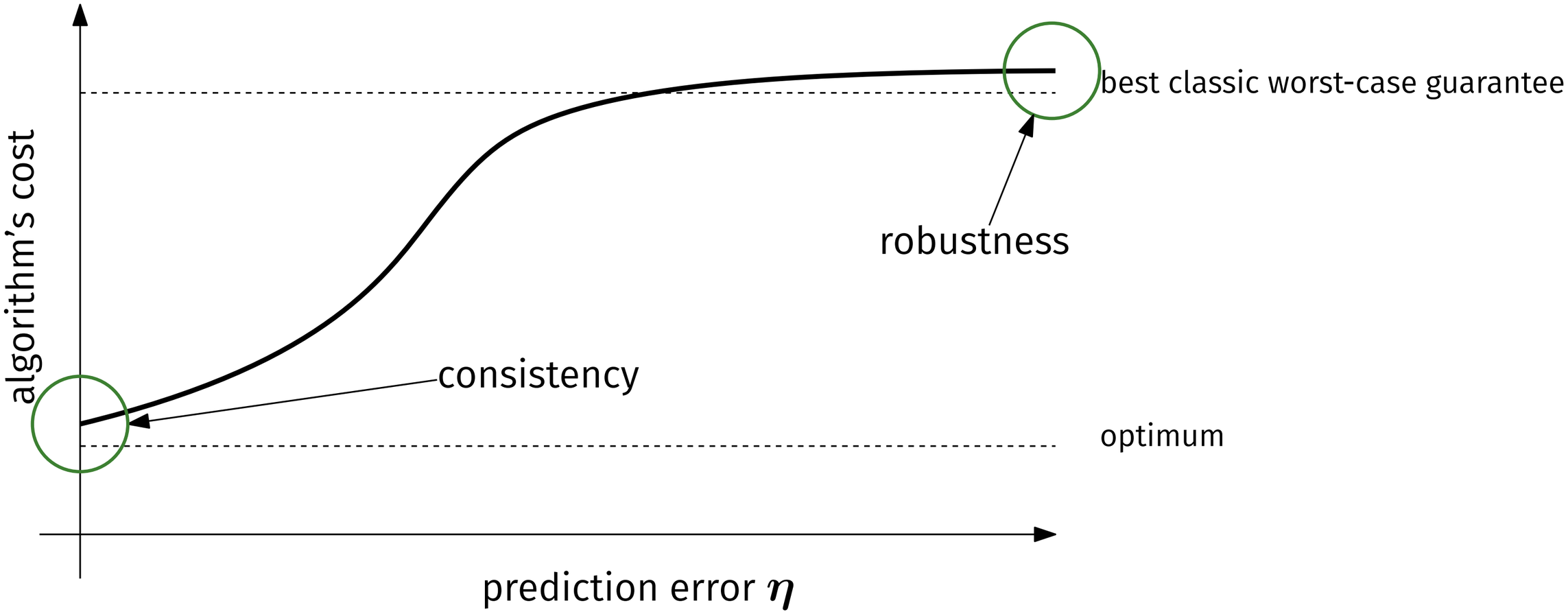
Learning-augmented algorithms

Input + black-box **predictions** (possibly inacurate, with some **error** η)



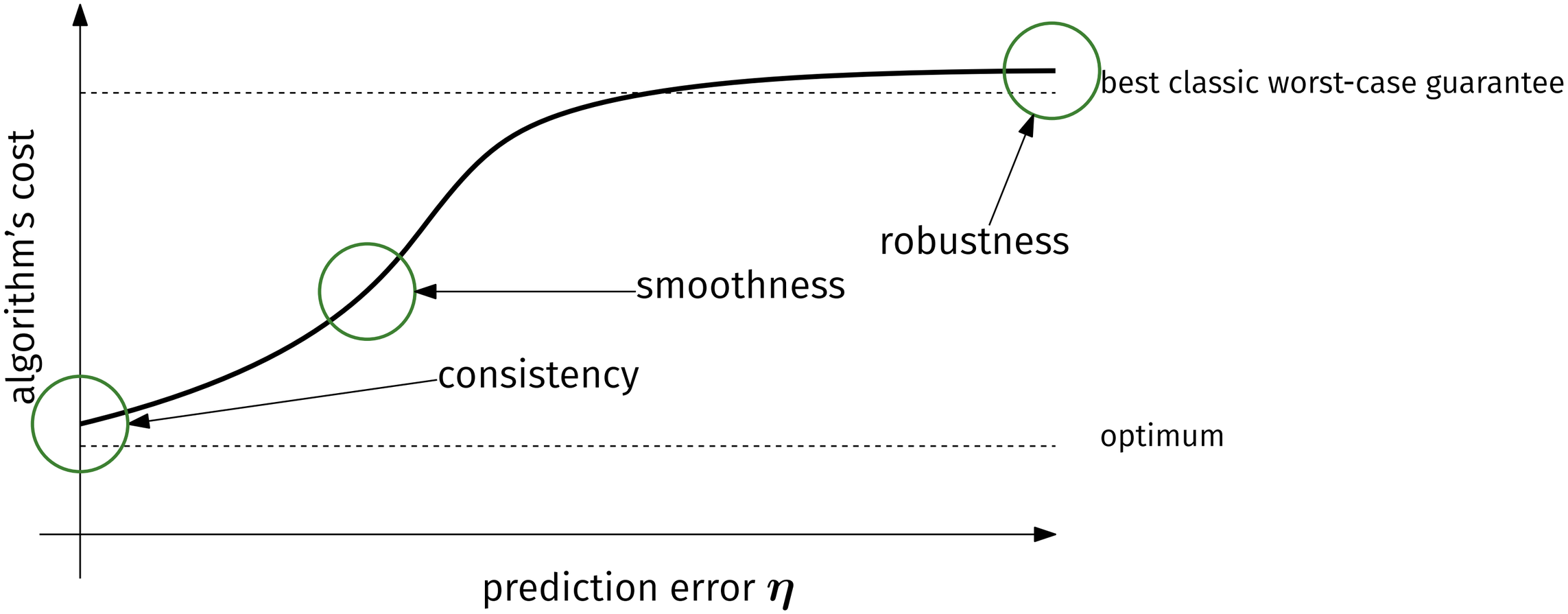
Learning-augmented algorithms

Input + black-box **predictions** (possibly inacurate, with some **error** η)



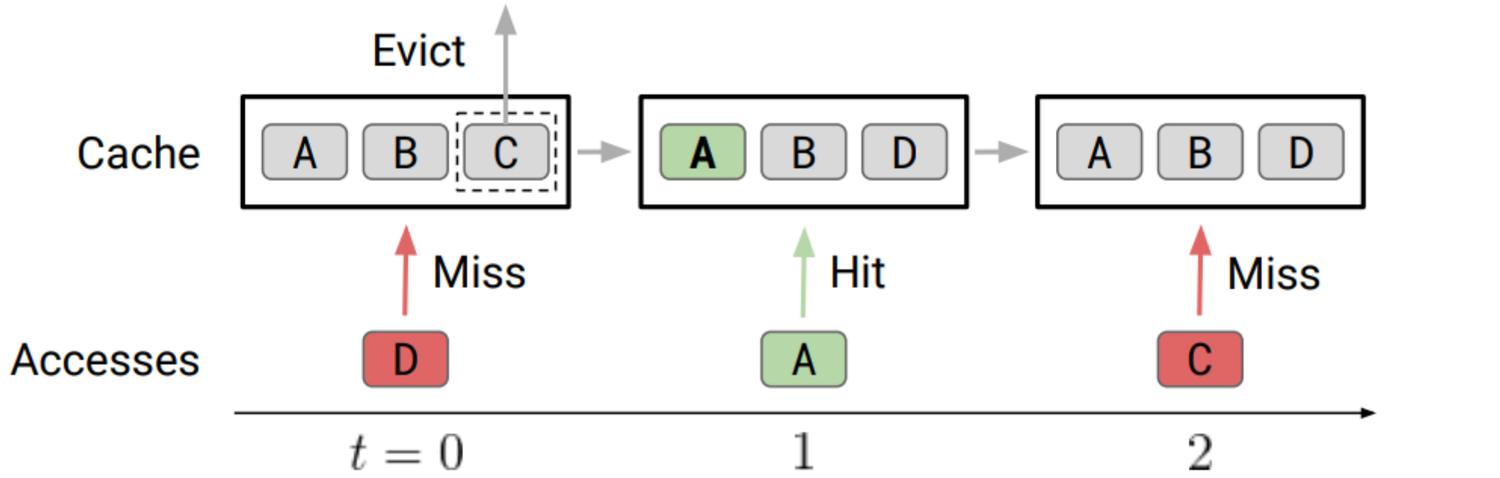
Learning-augmented algorithms

Input + black-box **predictions** (possibly inacurate, with some **error** η)



Predictions can improve **competitive ratio** of **online** algorithms

E.g.: **caching** [Lykouris, Vassilvitskii, ICML'18]



source: arxiv.org/abs/2006.16239

Prediction: *When currently requested item will be requested again?*

Result: $O(\min(\log k, \sqrt{\eta/OPT}))$-competitive algorithm (without predictions $\Theta(\log k)$ is tight)

Predictions can improve **running time** of **static** algorithms

E.g.: **max weight bipartite matching** [Dinitz, Im, Lavastida, Moseley, Vassilvitskii, NeurIPS'21]

Primal:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{subject to} \quad & \sum_{e \in N(v)} x_e = 1 \quad \forall\, v \in V \\ & x_e \geqslant 0 \qquad\quad \forall\, e \in E \end{aligned}$$

Dual:

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} y_v \\ \text{subject to} \quad & y_u + y_v \leqslant c_{u,v} \quad \forall\, (u,v) \in E \end{aligned}$$

Prediction: *dual LP solution*

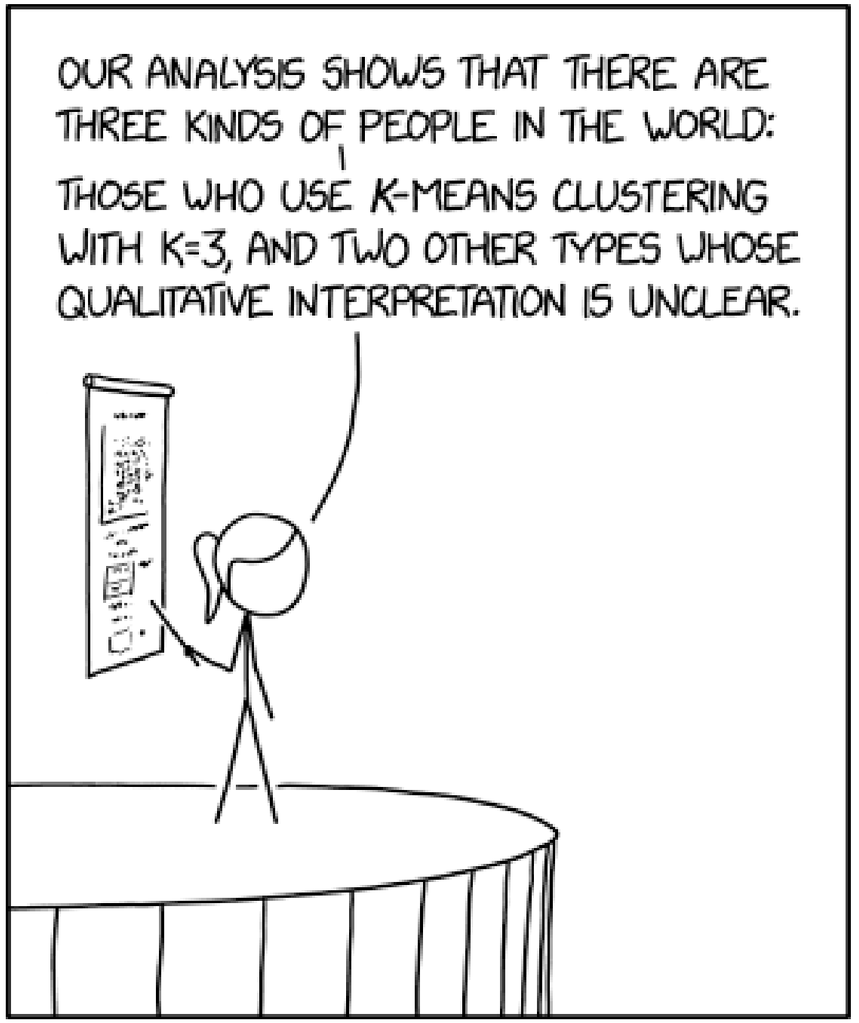
Result: $O(m\sqrt{n} \cdot \min(\sqrt{n}, \eta))$ time algorithm

(without predictions $O(mn)$ time Hungarian algorithm often used in practice)

Predictions can improve **approximation ratio** of polynomial time **approximation** algorithms

E.g.: **k-means clustering**

[Ergun, Feng, Silwal, Woodruff, Zhou, ICLR'22]
[Gamlath, Lattanzi, Norouzi-Fard, Svensson, COLT'22]

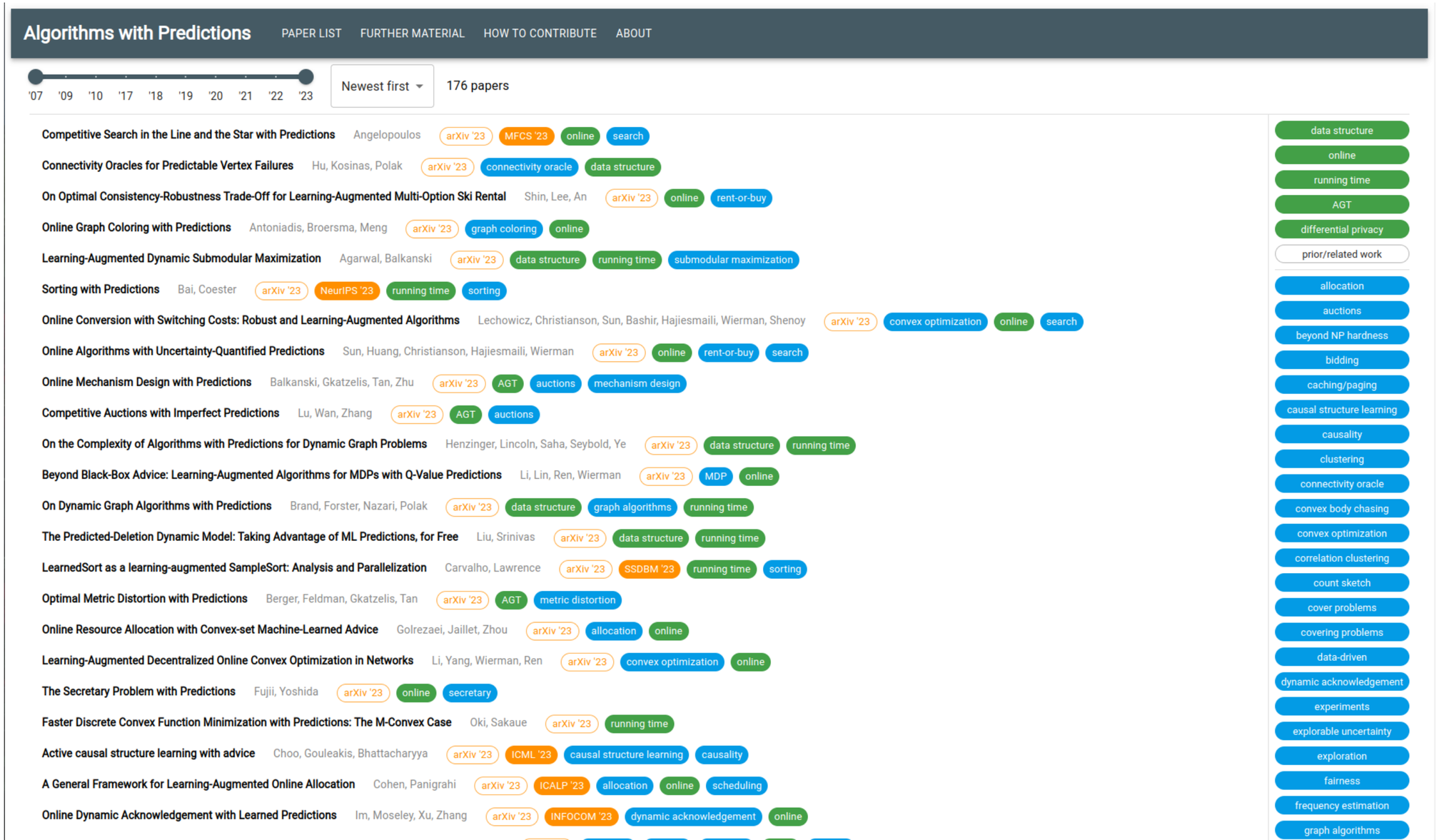


Prediction: *(noisy) clustering*

Result: $(1+\varepsilon)$**-approximation** algorithm

(without predictions $\Theta(1)$ is tight)

https://algorithms-with-predictions.github.io/



How predictions can improve **dynamic** algorithms?

How predictions can improve **dynamic** algorithms?

- Let’s predict **future**
- Let’s improve **running time**

How predictions can improve **dynamic** algorithms?

- Let’s predict **future**
- Let’s improve **running time**

Perfect and **full** knowledge about future = **offline** dynamic algorithms

How predictions can improve **dynamic** algorithms?

- Let’s predict **future**
- Let’s improve **running time**

Perfect and **full** knowledge about future = **offline** dynamic algorithms

Ideal result: With predictions we can do what is (provably) impossible without them

How predictions can improve **dynamic** algorithms?

- Let’s predict **future**
- Let’s improve **running time**

Perfect and **full** knowledge about future = **offline** dynamic algorithms

Ideal result: With predictions we can do what is (provably) impossible without them

OMv

How predictions can improve **dynamic** algorithms?

- Let’s predict **future**
- Let’s improve **running time**

Perfect and **full** knowledge about future = **offline** dynamic algorithms

Ideal result: With predictions we can do what is (provably) impossible without them

OMv

Open problem: Prove a stronger-than-offline **lower bound** against **online dynamic** algorithms under a **static hypothesis** (e.g., SETH, 3SUM)

How predictions can improve **dynamic** algorithms?

- Let's predict **future**
- Let's improve **running time**

Perfect and **full** knowledge about future = **offline** dynamic algorithms

Ideal result: With predictions we can do what is (provably) impossible without them

OMv

Open problem: Prove a stronger-than-offline **lower bound** against **online dynamic** algorithms under a **static hypothesis** (e.g., SETH, 3SUM)

see also [Bringmann, Grønlund, Künnemann, Larsen, ITCS'24]

Online matrix-vector multiplication hypothesis [HKNS'15]

Input:

- M — Boolean $n \times n$ matrix , given **offline**
- $v_1, \ldots, v_n$ — Boolean $n \times 1$ vectors, given one by one **online**

Output: $Mv_1, Mv_2, \ldots, Mv_n$

Hypothesis: requires $n^{3-o(1)}$ time

Online matrix-vector multiplication hypothesis [HKNS'15]

Input:

- M — Boolean $n \times n$ matrix , given **offline**
- $v_1, \ldots, v_n$ — Boolean $n \times 1$ vectors, given one by one **online**

Output: $Mv_1, Mv_2, \ldots, Mv_n$

Hypothesis: requires $n^{3-o(1)}$ time

Does not hold offline — compute $M \cdot [v_1, \ldots, v_n]$ in $O(n^\omega)$ time

Online matrix-vector multiplication **with predictions** [this work]

Input:

- M — Boolean $n \times n$ matrix , given **offline**
- $\hat{v}_1, \ldots, \hat{v}_n$ — **predicted** vectors, given **offline**
- $v_1, \ldots, v_n$ — Boolean $n \times 1$ vectors, given one by one **online**

Output: $Mv_1, Mv_2, \ldots, Mv_n$

Online matrix-vector multiplication **with predictions** [this work]

Input:

- M — Boolean $n \times n$ matrix , given **offline**
- $\hat{v}_1, \ldots, \hat{v}_n$ — **predicted** vectors, given **offline**
- $v_1, \ldots, v_n$ — Boolean $n \times 1$ vectors, given one by one **online**

Output: $Mv_1, Mv_2, \ldots, Mv_n$

Algorithm:

- Preprocessing in $O(n^\omega)$ time

over integers!

$$M \cdot [\hat{v}_1, \ldots, \hat{v}_n]$$

- Each request in $O(n\eta_i)$ time, $\eta_i = ||v_i - \hat{v}_i||_1 = ||v_i - \hat{v}_i||_0$

$$Mv_i = M(v_i - \hat{v}_i + \hat{v}_i) = M(v_i - \hat{v}_i) + M\hat{v}_i$$

$O(n\eta_i)$ $O(n)$

Online matrix-vector multiplication **with predictions** [this work]

Input:

- M — Boolean $n \times n$ matrix , given **offline**
- $\hat{v}_1, \ldots, \hat{v}_n$ — **predicted** vectors, given **offline**
- $v_1, \ldots, v_n$ — Boolean $n \times 1$ vectors, given one by one **online**

Output: $Mv_1, Mv_2, \ldots, Mv_n$

Algorithm:

- Preprocessing in $O(n^\omega)$ time

over integers!

$$M \cdot [\hat{v}_1, \ldots, \hat{v}_n]$$

- Each request in $O(n\eta_i)$ time, $\eta_i = ||v_i - \hat{v}_i||_1 = ||v_i - \hat{v}_i||_0$

$$Mv_i = M(v_i - \hat{v}_i + \hat{v}_i) = M(v_i - \hat{v}_i) + M\hat{v}_i$$

$O(n\eta_i)$ $O(n)$

Total time: $O(n^\omega + n\eta)$, $\eta = \sum_{i=1}^{n} \eta_i$

Incremental transitive closure (a.k.a. all-pairs reachability)

Update: add directed edge (u, v)
Query: is there a path from a to b?

Incremental transitive closure (a.k.a. all-pairs reachability)

Update: add directed edge (u, v)
Query: is there a path from a to b?

Upper bound: $O(nm)$ ammortized time (lazily maintain n single-source traversal trees)
Lower bound: n^2 updates and n^2 queries require $n^{3-o(1)}$ time, under OMv

Incremental transitive closure (a.k.a. all-pairs reachability)

Update: add directed edge (u, v)
Query: is there a path from a to b?

Upper bound: $O(nm)$ ammortized time (lazily maintain n single-source traversal trees)
Lower bound: n^2 updates and n^2 queries require $n^{3-o(1)}$ time, under OMv

Offline = all-pairs **bottleneck** paths ($O(n^{(3+\omega)/2}) \leqslant O(n^{2.687})$ time, via min-max matrix product)

$D[\boldsymbol{a}, \boldsymbol{b}] = \min_{\boldsymbol{P} \in \{\text{paths from } \boldsymbol{a} \text{ to } \boldsymbol{b}\}} \max_{\boldsymbol{e} \in \boldsymbol{P}} weight(\boldsymbol{e})$

Incremental transitive closure (a.k.a. all-pairs reachability)

Update: add directed edge (u, v)
Query: is there a path from a to b?

Upper bound: $O(nm)$ ammortized time (lazily maintain n single-source traversal trees)
Lower bound: n^2 updates and n^2 queries require $n^{3-o(1)}$ time, under OMv

Offline = all-pairs **bottleneck** paths ($O(n^{(3+\omega)/2}) \leqslant O(n^{2.687})$ time, via min-max matrix product)

Prediction: *sequence of insertions*. But, how to handle prediction errors?

Incremental transitive closure (a.k.a. all-pairs reachability)

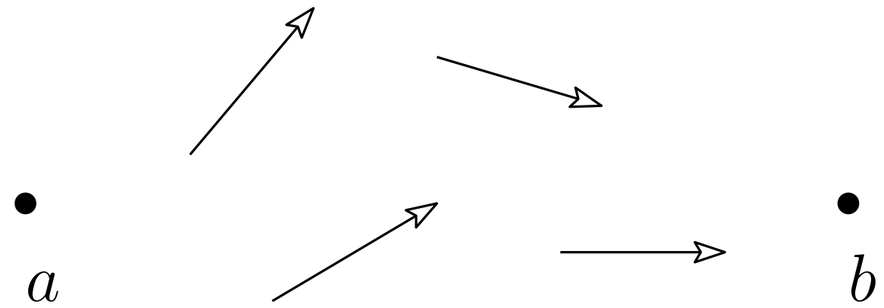
Update: add directed edge (u, v)
Query: is there a path from a to b?

Upper bound: $O(nm)$ ammortized time (lazily maintain n single-source traversal trees)
Lower bound: n^2 updates and n^2 queries require $n^{3-o(1)}$ time, under OMv

Offline = all-pairs **bottleneck** paths ($O(n^{(3+\omega)/2}) \leqslant O(n^{2.687})$ time, via min-max matrix product)

Prediction: *sequence of insertions*. But, how to handle prediction errors?

k edges "on the side" $\implies O(k^2)$ query time
$k \leqslant \ell_\infty$ error of predictions



Incremental transitive closure (a.k.a. all-pairs reachability)

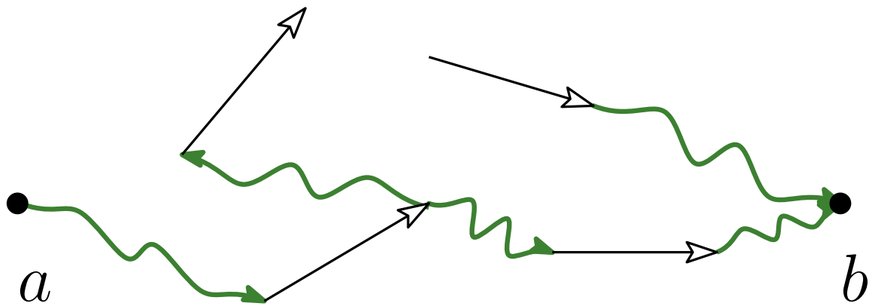
Update: add directed edge (u, v)
Query: is there a path from a to b?

Upper bound: $O(nm)$ ammortized time (lazily maintain n single-source traversal trees)
Lower bound: n^2 updates and n^2 queries require $n^{3-o(1)}$ time, under OMv

Offline = all-pairs **bottleneck** paths ($O(n^{(3+\omega)/2}) \leqslant O(n^{2.687})$ time, via min-max matrix product)

Prediction: *sequence of insertions*. But, how to handle prediction errors?

k edges "on the side" $\implies$ $O(k^2)$ query time
$k \leqslant \ell_\infty$ error of predictions



Incremental transitive closure (a.k.a. all-pairs reachability)

Update: add directed edge (u, v)
Query: is there a path from a to b?

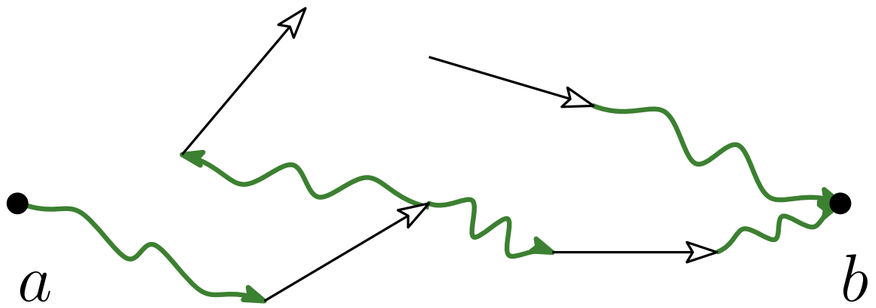
Upper bound: $O(nm)$ ammortized time (lazily maintain n single-source traversal trees)
Lower bound: n^2 updates and n^2 queries require $n^{3-o(1)}$ time, under OMv

Offline = all-pairs **bottleneck** paths ($O(n^{(3+\omega)/2}) \leqslant O(n^{2.687})$ time, via min-max matrix product)

Prediction: *sequence of insertions*. But, how to handle prediction errors?

k edges "on the side" $\implies$ $O(k^2)$ query time
$k \leqslant \ell_\infty$ error of predictions

$O(1)$ query time $\impliedby$ dynamic APBP



Incremental transitive closure (a.k.a. all-pairs reachability)

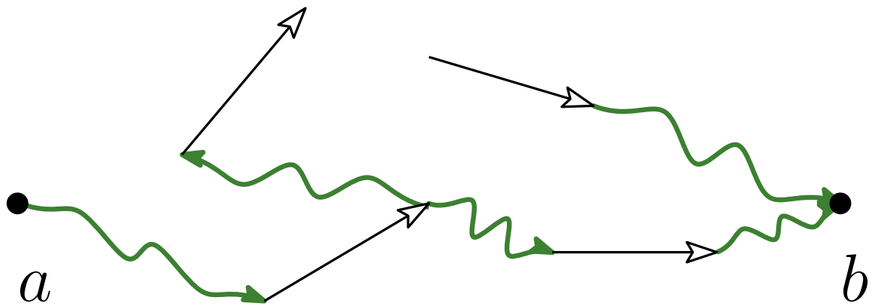
Update: add directed edge (u, v)
Query: is there a path from a to b?

Upper bound: $O(nm)$ ammortized time (lazily maintain n single-source traversal trees)
Lower bound: n^2 updates and n^2 queries require $n^{3-o(1)}$ time, under OMv

Offline = all-pairs **bottleneck** paths ($O(n^{(3+\omega)/2}) \leqslant O(n^{2.687})$ time, via min-max matrix product)

Prediction: *sequence of insertions*. But, how to handle prediction errors?

k edges "on the side" $\implies$ $O(k^2)$ query time
↳ $k \leqslant \ell_\infty$ error of predictions



$O(1)$ query time $\impliedby$ dynamic APBP
↳ incremental: $O(n^2)$
fully dynamic: $O(n^{2.5})$ [via APSP]

Incremental transitive closure (a.k.a. all-pairs reachability)

Update: add directed edge (u, v)
Query: is there a path from a to b?

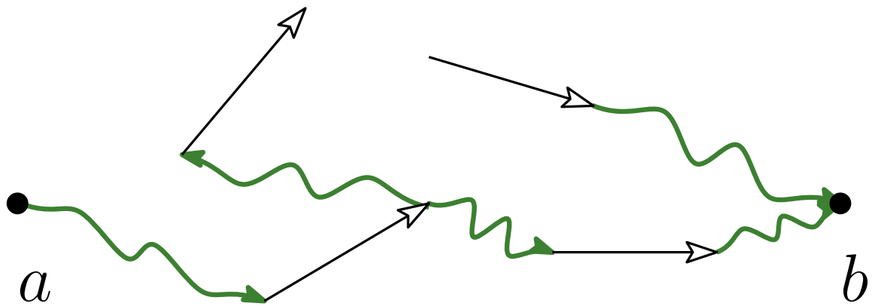
Upper bound: $O(nm)$ ammortized time (lazily maintain n single-source traversal trees)
Lower bound: n^2 updates and n^2 queries require $n^{3-o(1)}$ time, under OMv

Offline = all-pairs **bottleneck** paths ($O(n^{(3+\omega)/2}) \leqslant O(n^{2.687})$ time, via min-max matrix product)

Prediction: *sequence of insertions*. But, how to handle prediction errors?

k edges "on the side" $\implies$ $O(k^2)$ query time
↳ $k \leqslant \ell_\infty$ error of predictions

$O(1)$ query time $\impliedby$ dynamic APBP
↳ incremental: $O(n^2)$
fully dynamic: $O(n^{2.5})$ [via APSP]



← Open problem: Improve

Our results for graph problems

Our results for graph problems

Partially dynamic
(edge insertions **or** deletions)

Prediction: list of **updates**

- Transitive closure
- Aproximate APSP

Preprocessing	$O(n^{2.687})$
Update	$O(1)$
Query	$O(\eta^2)$

ℓ_∞ prediction error $\leftarrow\uparrow$

Our results for graph problems

Partially dynamic
(edge insertions **or** deletions)

Prediction: list of **updates**

- Transitive closure
- Aproximate APSP

Preprocessing	$O(n^{2.687})$
Update	$O(1)$
Query	$O(\eta^2)$

ℓ_∞ prediction error $\rightharpoondown\uparrow$

Fully dynamic
(edge insertions **and** deletions)

Prediction: list of **operations**

- Triangle detection
- Exact matching
- Single-source reachability
- many more…

Preprocessing	$O(n^{2.373})$
Update	$O(n^{1.373} + n\eta_i)$
Query	$O(n^{1.373} + n\eta_i)$

ℓ_1 error per operation $\rightharpoondown\uparrow$

Our results for graph problems

Partially dynamic
(edge insertions **or** deletions)

Prediction: list of **updates**

- Transitive closure
- Aproximate APSP

Preprocessing	$O(n^{2.687})$
Update	$O(1)$
Query	$O(\eta^2)$

ℓ_∞ prediction error $\leftharpoondown\uparrow$

Fully dynamic
(edge insertions **and** deletions)

Prediction: list of **operations**

- Triangle detection
- Exact matching
- Single-source reachability
- many more…

Preprocessing	$O(n^{2.373})$
Update	$O(n^{1.373} + n\eta_i)$
Query	$O(n^{1.373} + n\eta_i)$

ℓ_1 error per operation $\leftharpoondown\uparrow$

Fully dynamic
(edge insertions **and** deletions)

Prediction: **deletion** times
(given during insertions)

- All-pairs shortest paths (Exact APSP)

Preprcessing	—
Update	$\tilde{O}(n^2)$
Query	$\tilde{O}(n^2\eta_i)$

ℓ_1 error per operation $\leftharpoondown\uparrow$

Our results for graph problems

see also [Liu, Srinivas, 2023]

Partially dynamic
(edge insertions **or** deletions)

Prediction: list of **updates**

- Transitive closure
- Aproximate APSP

Preprocessing	$O(n^{2.687})$
Update	$O(1)$
Query	$O(\eta^2)$

ℓ_∞ prediction error

see also [Henzinger, Saha, Seybold, Ye, ITCS'24]

Fully dynamic
(edge insertions **and** deletions)

Prediction: list of **operations**

- Triangle detection
- Exact matching
- Single-source reachability
- many more…

Preprocessing	$O(n^{2.373})$
Update	$O(n^{1.373} + n\eta_i)$
Query	$O(n^{1.373} + n\eta_i)$

ℓ_1 error per operation

Fully dynamic
(edge insertions **and** deletions)

Prediction: **deletion** times
(given during insertions)

- All-pairs shortest paths (Exact APSP)

Preprcessing	—
Update	$\tilde{O}(n^2)$
Query	$\tilde{O}(n^2\eta_i)$

ℓ_1 error per operation

Our results for graph problems

Partially dynamic
(edge insertions **or** deletions)

Prediction: list of **updates**

- Transitive closure
- Aproximate APSP

Preprocessing	$O(n^{2.687})$
Update	$O(1)$
Query	$O(\eta^2)$

ℓ_∞ prediction error ↛

see also [Henzinger, Saha, Seybold, Ye, ITCS'24]

Fully dynamic
(edge insertions **and** deletions)

Prediction: list of **operations**

- Triangle detection
- Exact matching
- Single-source reachability
- many more…

Preprocessing	$O(n^{2.373})$
Update	$O(n^{1.373} + n\eta_i)$
Query	$O(n^{1.373} + n\eta_i)$

ℓ_1 error per operation ↛

see also [Liu, Srinivas, 2023]

Fully dynamic
(edge insertions **and** deletions)

Prediction: **deletion** times
(given during insertions)

- All-pairs shortest paths (Exact APSP)

Preprcessing	—
Update	$\tilde{O}(n^2)$
Query	$\tilde{O}(n^2\eta_i)$

ℓ_1 error per operation ↛

Thank you!