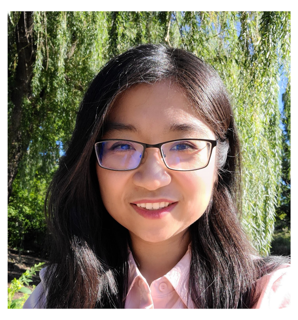
The Predicted-Updates Dynamic Model: Offline to Fully-Dynamic Transformations



Quanquan C. Liu
Yale University
quanquan.liu@yale.edu

Vaidehi Srinivas
Northwestern University
vaidehi@u.northwestern.edu

COLT '24
Session 4D-1
Wed. Jul. 03, 14:30

Dynamic Algorithms with Predictions

Large gap between fully-dynamic and offline-dynamic runtime for some problems

Ex. Triconnectivity

$O(n^{2/3})$ fully-dynamic update time

$O(T \text{ polylog } n)$ offline-dynamic runtime for T updates

↳ via slick divide-and-conquer algorithm

? Can we use predictions of future events to lift fast divide-and-conquer algorithms to the fully-dynamic setting?

A Yes! Can simultaneously achieve

1. Consistency: offline performace for high quality predictions
2. Robustness: no worse than fully dynamic performance for low quality preds.
3. Graceful degradation: performance deteriorates gracefully with prediction error

Same framework also lifts incremental and decremental algorithms to fully-dynamic setting.

Applications

Our framework gives improved runtimes with predictions to many well-studied problems, out of the box.

Problem	Best Fully Dynamic Runtimes		New Predicted-Update Runtimes (Theorems 6.4 to 6.6)	
Planar Digraph APSP	$\widetilde{O}(n^{2/3})$	[FR06, Kle05]	$\widetilde{O}(\sqrt{n})$	[DGWN22]
Triconnectivity	$O(n^{2/3})$	[GIS99]	$\widetilde{O}(1)$	[HR20, PSS17]
k-Edge Connectivity	$n^{o(1)}$	[JS22]	$\widetilde{O}(1)$	[CDK+21]
Dynamic DFS Tree	$\widetilde{O}(\sqrt{mn})$	[BCCK19]	$\widetilde{O}(n)$	[BCCK19, CDW+18]
Minimum Spanning Forest	$\widetilde{O}(1)$	[HDLT01]	$\widetilde{O}(1)$	[Epp94]
APSP	$\left(\frac{256}{k^2}\right)^{4/k}$-Approx $\widetilde{O}(n^k)$ update $\widetilde{O}(n^{k/8})$ query	[FGNS23]	$(2r-1)^k$-Approx $\widetilde{O}\left(m^{1/(k+1)}n^{k/r}\right)$	[CGH+20]
AP Maxflow/Mincut	$O(\log(n)\log\log n)$-Approx $\widetilde{O}\left(n^{2/3+o(1)}\right)$	[CGH+20]	$O\left(\log^{8k}(n)\right)$-Approx. $\widetilde{O}\left(n^{2/(k+1)}\right)$	[Gor19, GHS19]
MCF	$(1+\varepsilon)$-Approx $\widetilde{O}(1)$ update $\widetilde{O}(n)$ query	[CGH+20]	$O(\log^{8k}(n))$-Approx. $\widetilde{O}\left(n^{2/(k+1)}\right)$ update $\widetilde{O}(P^2)$ query	[Gor19, GHS19]
Strongly Connected Components	$\Omega(m^{1-\varepsilon})$ query or update	[AW14]	$\widetilde{O}(m)$	[Rod13]
Uniform Sparsest Cut	$2^{O(\log^{5/6}(n))}$-Approx $2^{O(\log^{5/6}(n))}$ update $O(\log^{1/6}(n))$ query	[GRST21]	$O\left(\log^{8k}(n)\right)$-Approx $\widetilde{O}\left(n^{2/(k+1)}\right)$ $O(1)$ query	[Gor19, GHS19]
Submodular Max	$1/4$-Approx $\widetilde{O}(k^2)$	[DFL+23]	0.3178-Approx $\widetilde{O}(\text{poly}(k))$	[FLN+22]

Results

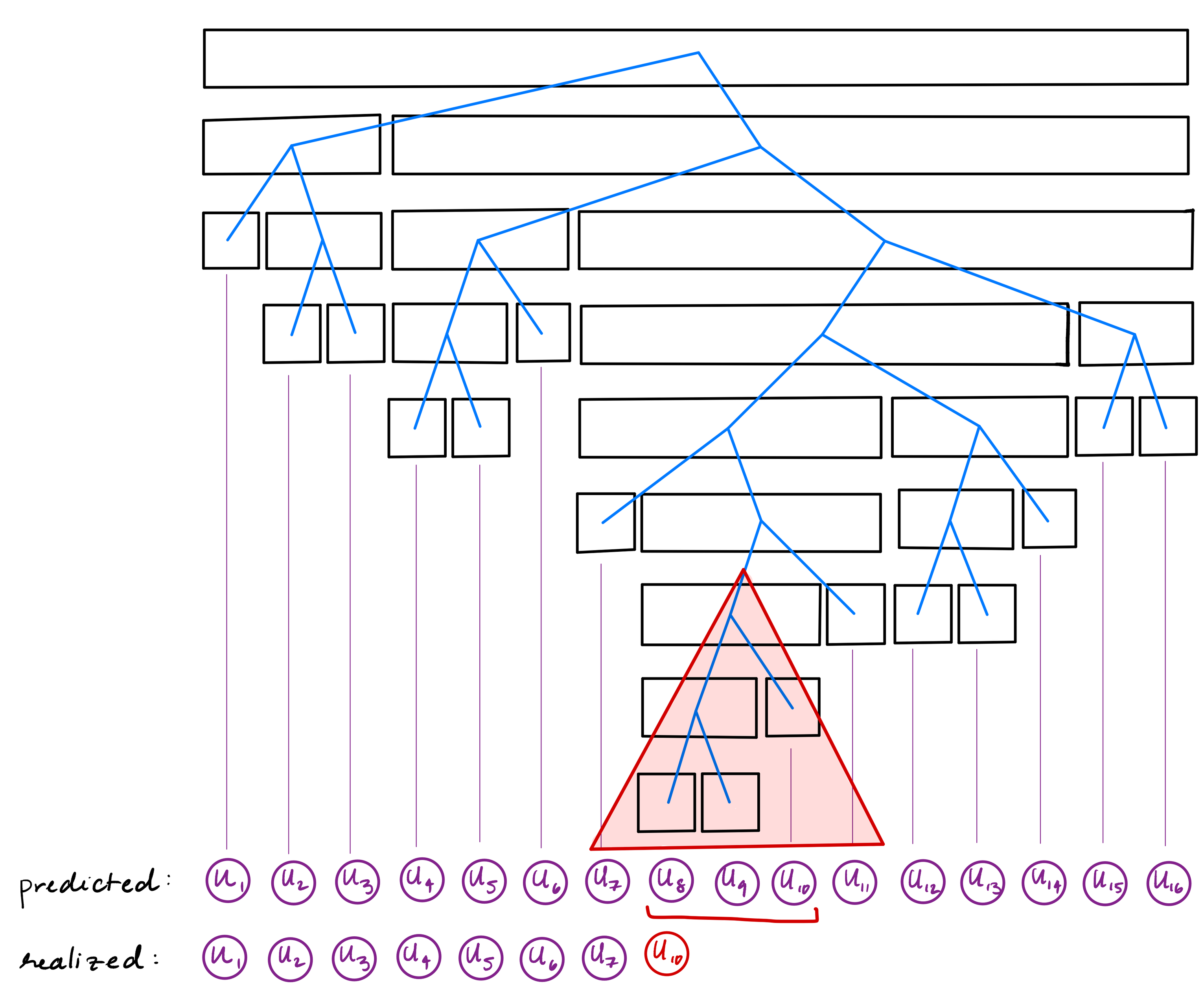
Informal Theorem: Given offline divide-and-conquer algorithm to compute $f(\cdot)$ that does $\tilde{O}(T)$ work, and fully-dynamic alg. B,

we can design a predicted-updates algorithm that does total work

$$\tilde{O}\left(\min\left\{T + \|\text{pred. error}\|_1,\ T \cdot \text{updatetime}(B)\right\}\right).$$

Key Technical Idea

Randomly splitting recursive subproblems ensures effects of prediction errors stay local.



! Small errors only require fixing a small subtree of the divide-and-conquer, in expectation.

Future Directions

- Better dependency on prediction error?
- Make this deterministic or robust to adaptive adversaries
- Worst-case vs. amortized bounds
- Better dynamic subroutines for static problems