



Optimized Priority Scheduling for Faster Scalable Belief Propagation

Abnash Bassi^{*}, Gilead Posluns[†], Mark C. Jeffrey[†]

Probabilistic Graphical Models (PGM) 2026

^{*} **Carnegie
Mellon
University**

[†]  **UNIVERSITY OF
TORONTO**

We're all impatient...how can we compute faster?

We're all impatient...how can we compute faster?

Parallelism!

Instead of performing compute **sequentially**



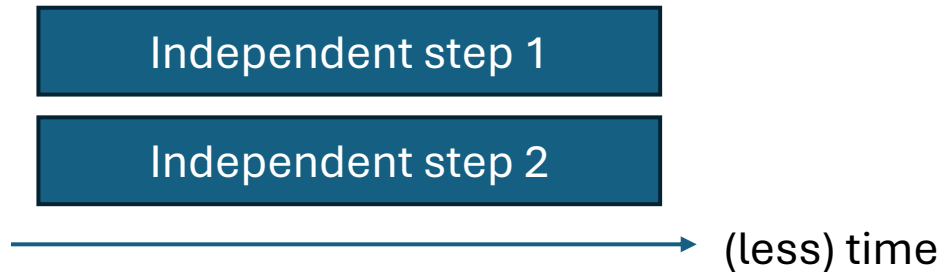
We're all impatient...how can we compute faster?

Parallelism!

Instead of performing compute **sequentially**



we can **overlap** independent compute to save time



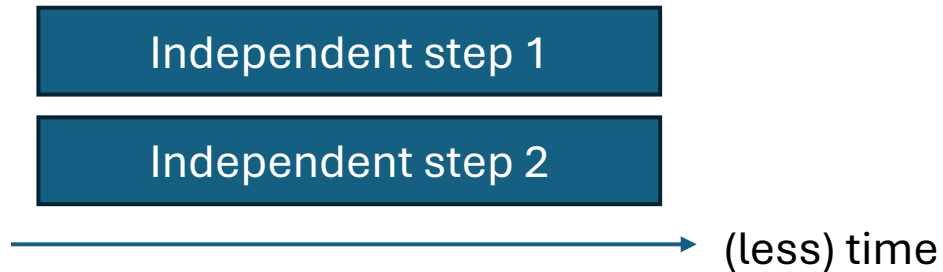
We're all impatient...how can we compute faster?

Parallelism!

Instead of performing compute **sequentially**



we can **overlap** independent compute to save time



However, the challenge is unlocking various forms of parallelism

Some applications are more challenging to parallelize than others

(Relatively) easy

Tensor algebra e.g., Matrix-
Matrix multiplication

Not so easy

Belief propagation

Why?

1. Fine-grained data dependences
2. Data dependences not known in advance

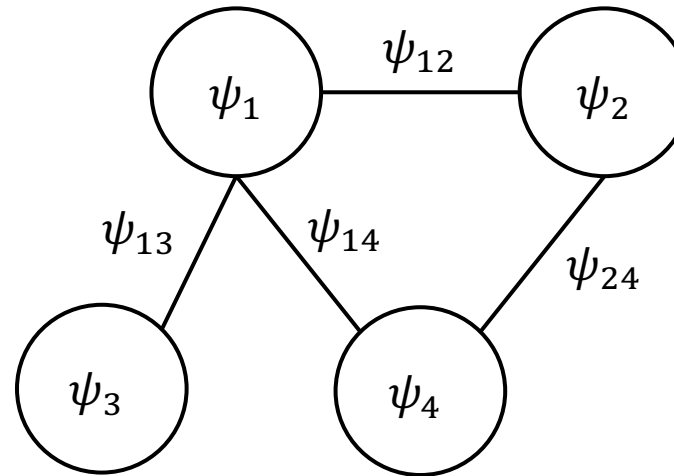
Our Contributions

- Address BP's parallelism bottleneck by adopting a state-of-the-art priority scheduler and a lazy-update optimization
- Identify one winning algorithm across inputs
- First known work to scale parallel BP up to 384 threads $\rightarrow >1.6\times$ per input type and geomean $2.66\times$ speedup

Belief Propagation Fundamentals

We consider a pairwise MRF

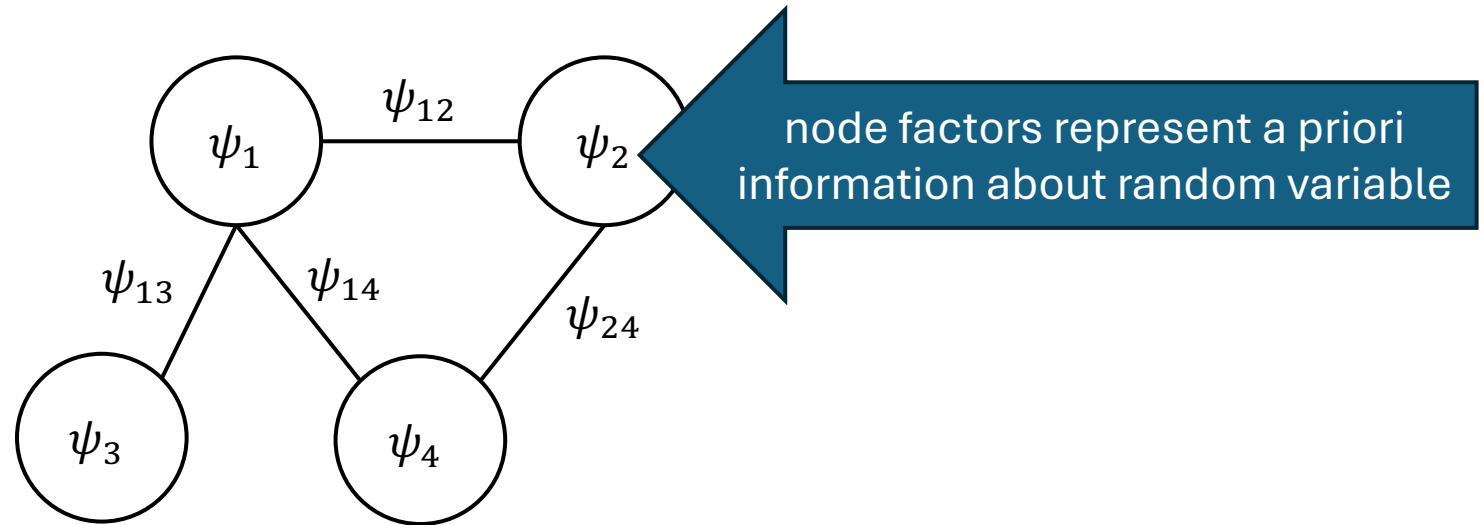
for random variables X_1, X_2, X_3, X_4 *



* using notation from Aksenov et al., 2020

We consider a pairwise MRF

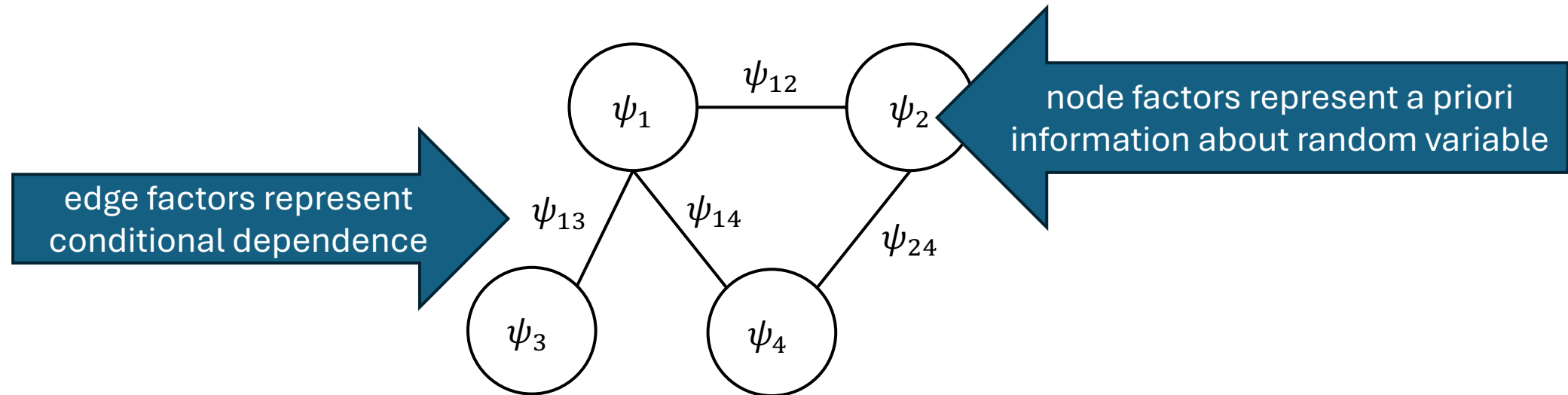
for random variables X_1, X_2, X_3, X_4 *



* using notation from Aksenov et al., 2020

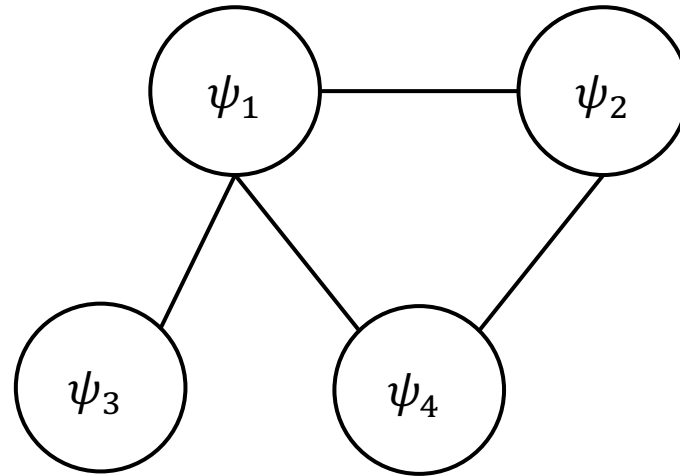
We consider a pairwise MRF

for random variables X_1, X_2, X_3, X_4 *



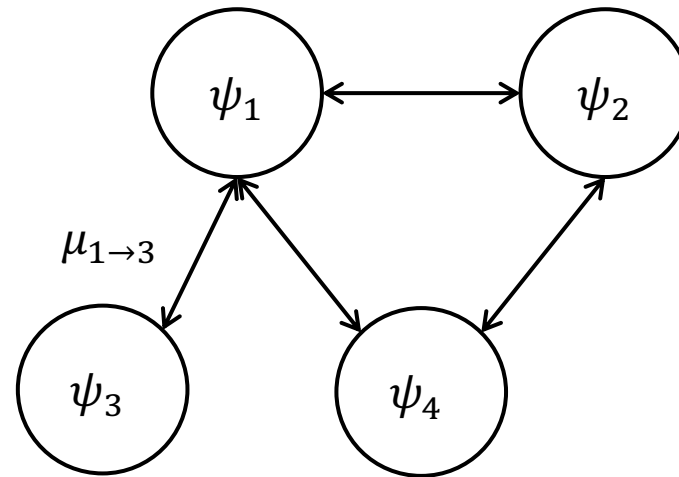
* using notation from Aksenov et al., 2020

What is (Residual*) Belief Propagation?



* (Elidan et al., 2006)

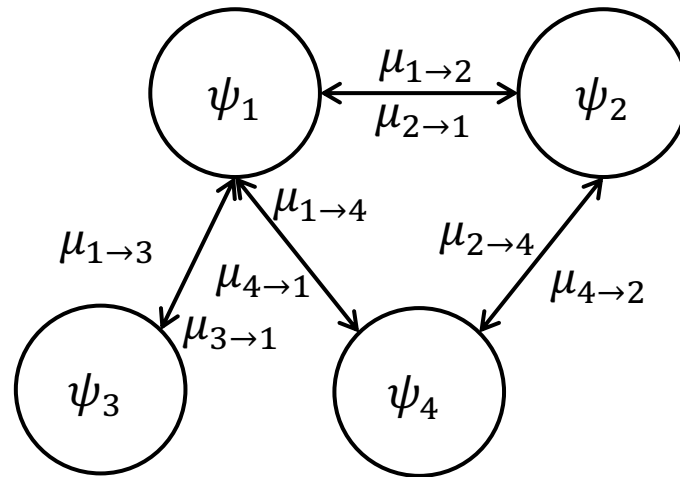
What is (Residual) Belief Propagation?



Pass messages between nodes

* (Elidan et al., 2006)

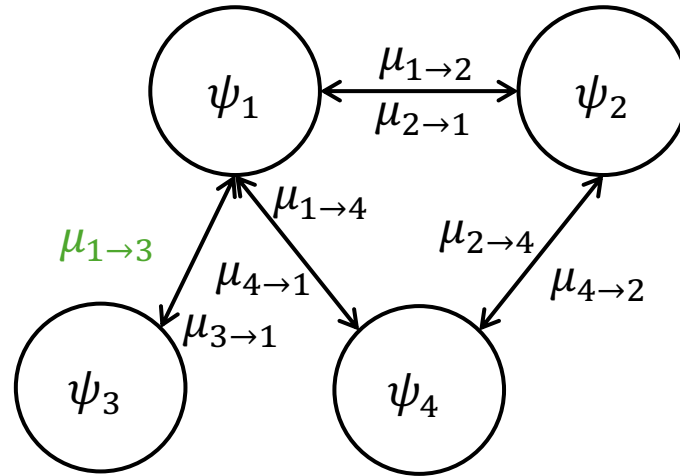
What is (Residual) Belief Propagation?



Pass messages between nodes

* (Elidan et al., 2006)

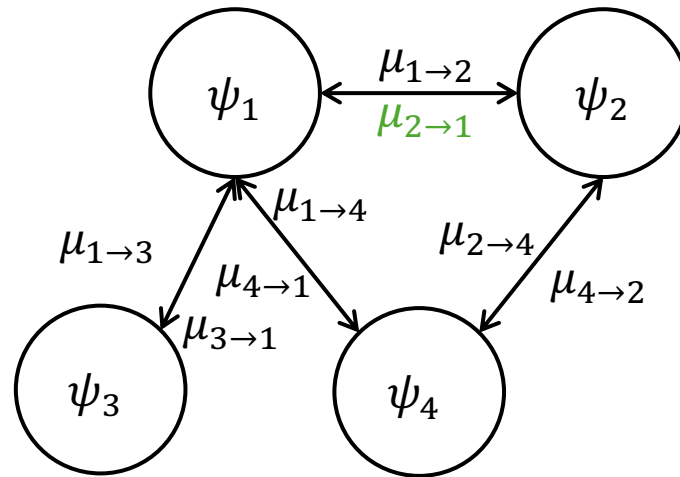
What is (Residual) Belief Propagation?



Compute priority as residual
(refer to paper for formulation)

* (Elidan et al., 2006)

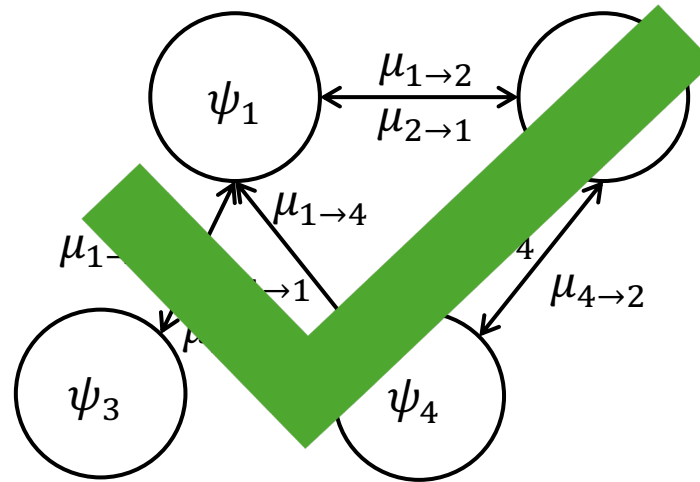
What is (Residual) Belief Propagation?



Repeatedly update messages until we converge

* (Elidan et al., 2006)

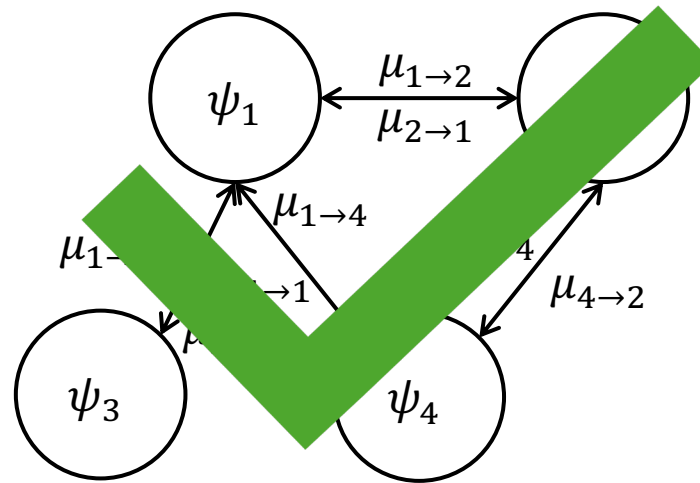
What is (Residual) Belief Propagation?



Repeatedly update messages until we converge

* (Elidan et al., 2006)

What is (Residual) Belief Propagation?



Using BP, we can run inference, i.e., compute conditional probability, on MRFs

* (Elidan et al., 2006)

What is *Relaxed* Residual Belief Propagation?

Instead of always choosing the highest priority message, relax!



Choose one of the top k highest priority messages^{*}

^{*} (Aksenov et al., 2020)

Limitations of Prior BP Approaches

What BP algorithm should you use for speed?

Input	Prior Work					Relaxed			
	Synch	Coarse-G	Splash (10)	RS (2)	Bucket	Residual	Weight-Decay	Priority	Smart Splash (2)
Tree	2.538x	0.265x	1.648x	2.252x	1.692x	1.391x	1.282x	1.239x	2.121x
Ising	3.009x	0.801x	5.393x	11.731x	3.311x	6.720x	6.276x	5.759x	14.175x
Potts	—	0.624x	1.041x	11.855x	—	7.454x	5.978x	5.850x	15.235x
LDPC	17.735x	1.166x	—	5.150x	3.044x	13.393x	5.615x	—	10.519x

Table 1: Algorithm speedups with respect to the sequential residual algorithm. Higher is better.

* (Aksenov et al., 2020)

What BP algorithm should you use for speed?

Input	Prior Work					Relaxed			
	Synch	Coarse-G	Splash (10)	RS (2)	Bucket	Residual	Weight-Decay	Priority	Smart Splash (2)
Tree	2.538x	0.265x	1.648x	2.252x	1.692x	1.391x	1.282x	1.239x	2.121x
Ising	3.009x	0.801x	5.393x	11.731x	3.311x	6.720x	6.276x	5.759x	14.175x
Potts	—	0.624x	1.041x	11.855x	—	7.454x	5.978x	5.850x	15.235x
LDPC	17.735x	1.166x	—	5.150x	3.044x	13.393x	5.615x	—	10.519x

Table 1: Algorithm speedups with respect to the sequential residual algorithm. Higher is better.

* (Aksenov et al., 2020)

What BP algorithm should you use for speed?

Input	Prior Work					Relaxed			
	Synch	Coarse-G	Splash (10)	RS (2)	Bucket	Residual	Weight-Decay	Priority	Smart Splash (2)
Tree	2.538x	0.265x	1.648x	2.252x	1.692x	1.391x	1.282x	1.239x	2.121x
Ising	3.009x	0.801x	5.393x	11.731x	3.311x	6.720x	6.276x	5.759x	14.175x
Potts	—	0.624x	1.041x	11.855x	—	7.454x	5.978x	5.850x	15.235x
LDPC	17.735x	1.166x	—	5.150x	3.044x	13.393x	5.615x	—	10.519x

Table 1: Algorithm speedups with respect to the sequential residual algorithm. Higher is better.

* (Aksenov et al., 2020)

What BP algorithm should you use for speed?

Input	Prior Work					Relaxed			
	Synch	Coarse-G	Splash (10)	RS (2)	Bucket	Residual	Weight-Decay	Priority	Smart Splash (2)
Tree	2.538x	0.265x	1.648x	2.252x	1.692x	1.391x	1.282x	1.239x	2.121x
Ising	3.009x	0.801x	5.393x	11.731x	3.311x	6.720x	6.276x	5.759x	14.175x
Potts	—	0.624x	1.041x	11.855x	—	7.454x	5.978x	5.850x	15.235x
LDPC	17.735x	1.166x	—	5.150x	3.044x	13.393x	5.615x	—	10.519x

Table 1: Algorithm speedups with respect to the sequential residual algorithm. Higher is better.

* (Aksenov et al., 2020)

What BP algorithm should you use for speed?

Input	Prior Work					Relaxed			
	Synch	Coarse-G	Splash (10)	RS (2)	Bucket	Residual	Weight-Decay	Priority	Smart Splash (2)
Tree	2.538x	0.265x	1.648x	2.252x	1.692x	1.391x	1.282x	1.239x	2.121x
Ising	3.009x	0.801x	5.393x	11.731x	3.311x	6.720x	6.276x	5.759x	14.175x
Potts	—	0.624x	1.041x	11.855x	—	7.454x	5.978x	5.850x	15.235x
LDPC	17.735x	1.166x	—	5.150x	3.044x	13.393x	5.615x	—	10.519x

Table 1: Algorithm speedups with respect to the sequential residual algorithm. Higher is better.

* (Aksenov et al., 2020)

What BP algorithm should you use for speed?

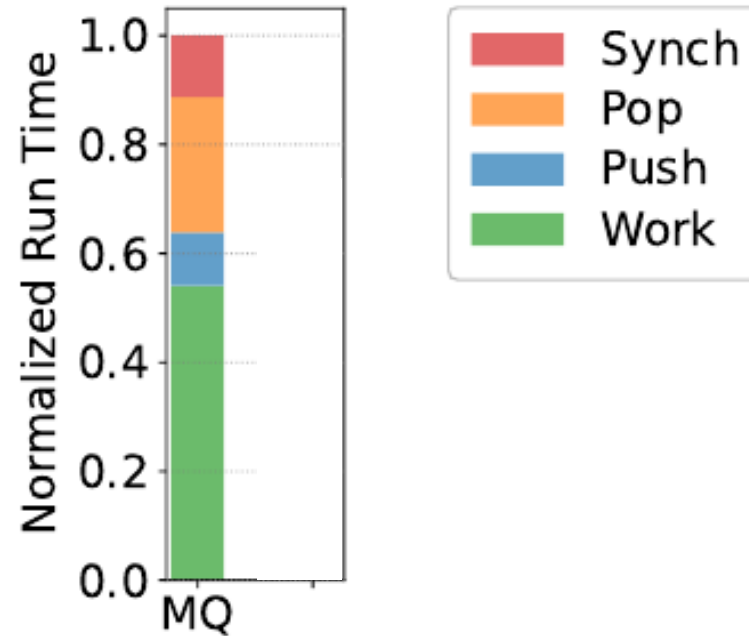
Input	Prior Work					Relaxed			
	Synch	Coarse-G	Splash (10)	RS (2)	Bucket	Residual	Weight-Decay	Priority	Smart Splash (2)
Tree	2.538x	0.265x	1.648x	2.252x	1.692x	1.391x	1.282x	1.239x	2.121x
Ising	3.009x	0.801x	5.393x	11.731x	3.311x	6.720x	6.276x	5.759x	14.175x
Potts	—	0.624x	1.041x	11.855x	—	7.454x	5.978x	5.850x	15.235x
LDPC	17.735x	1.166x	—	5.150x	3.044x	13.393x	5.615x	—	10.519x

Table 1: Algorithm speedups with respect to the sequential residual algorithm. Higher is better.

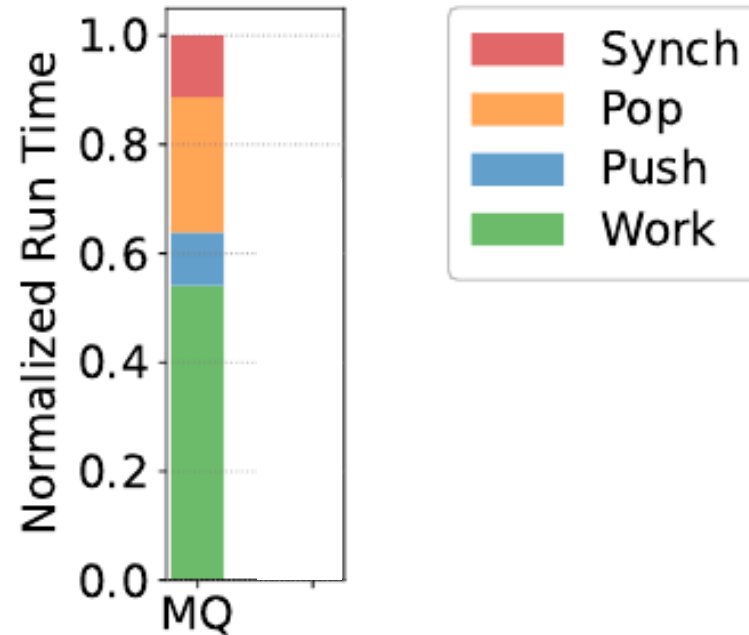
Prior work* has no winning BP algorithm.
It is non-trivial to select the right algorithm for your use case.

* (Aksenov et al., 2020)

Where does the time go?

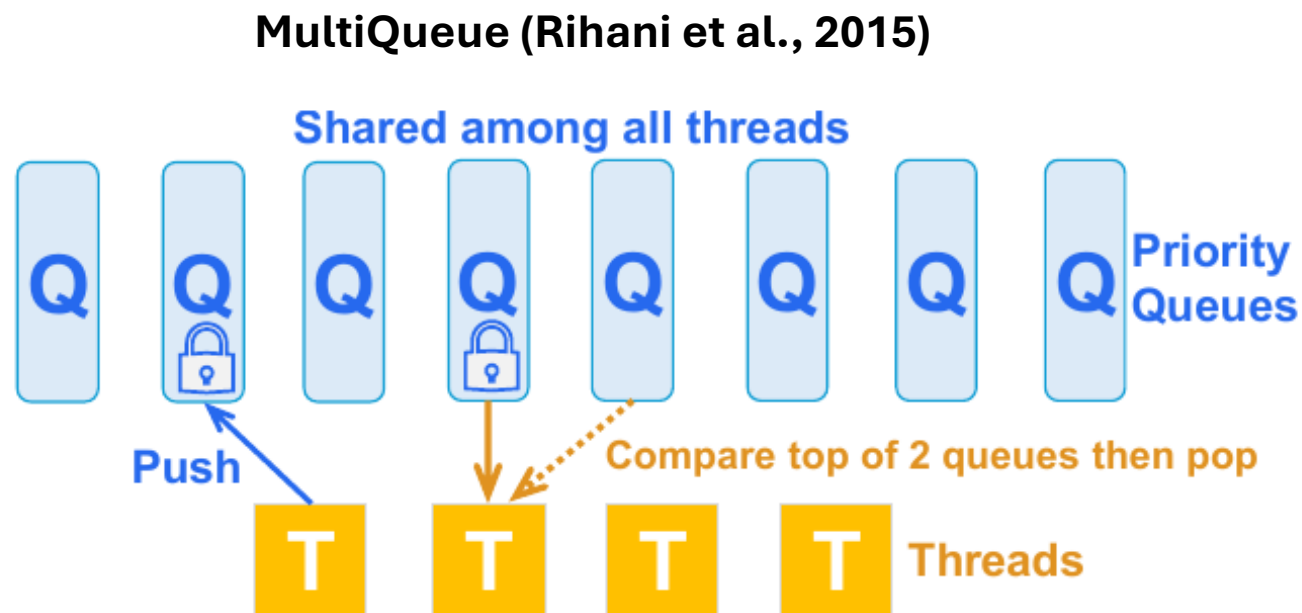
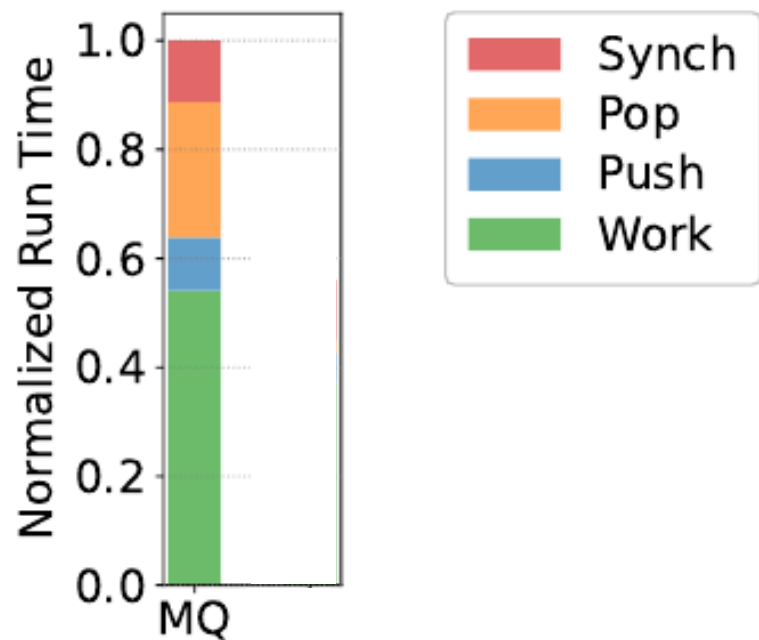


Where does the time go?

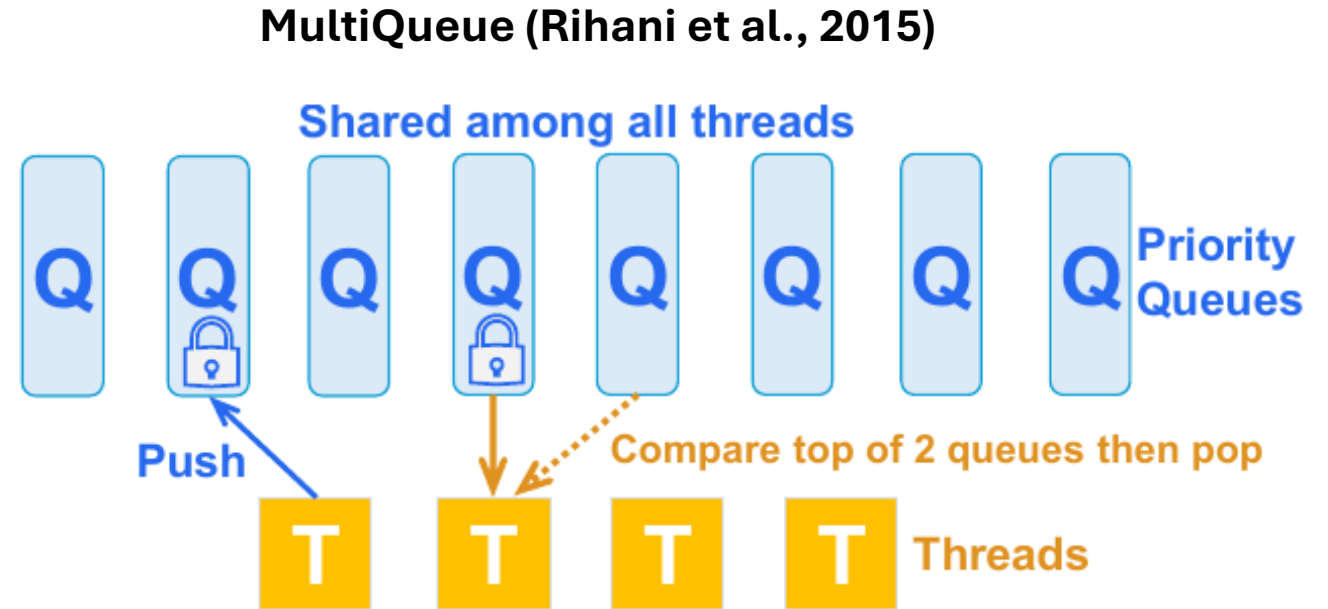
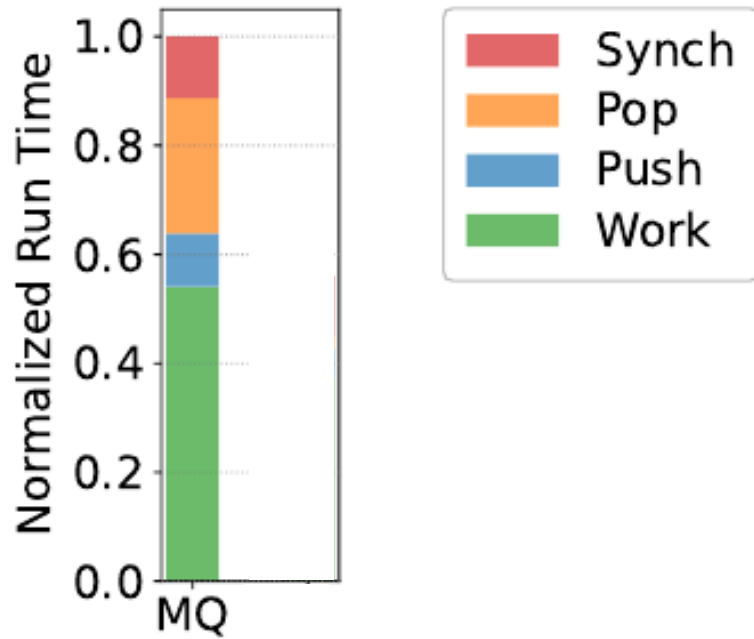


Prior work has high overhead for queue accesses

Why is there high queue access overhead?

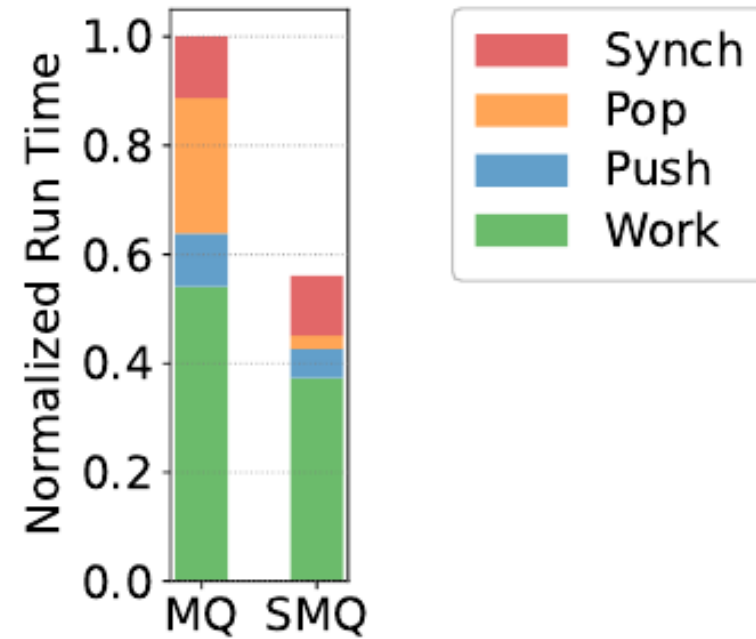


Why is there high queue access overhead?

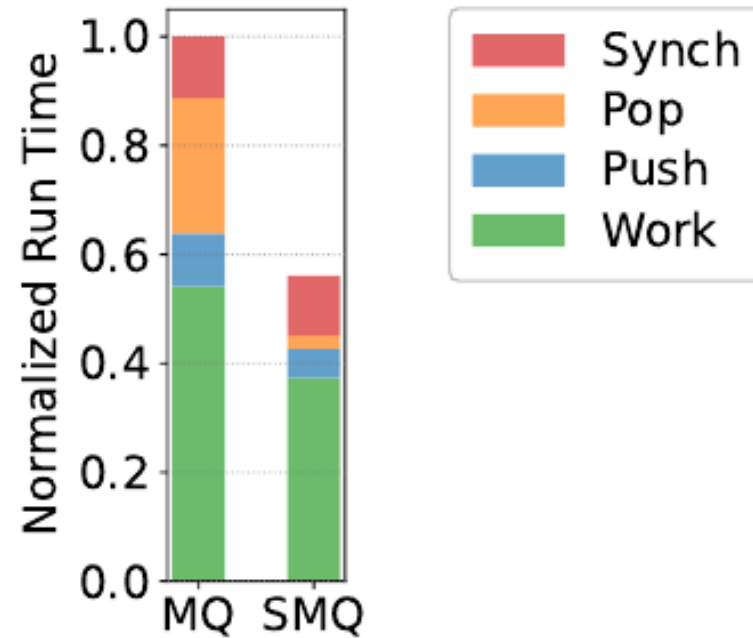


MultiQueue has less contention than one shared queue with a lock, but now *all* threads share the priority queues

Our approach cuts down on queue access overhead



Our approach cuts down on queue access overhead



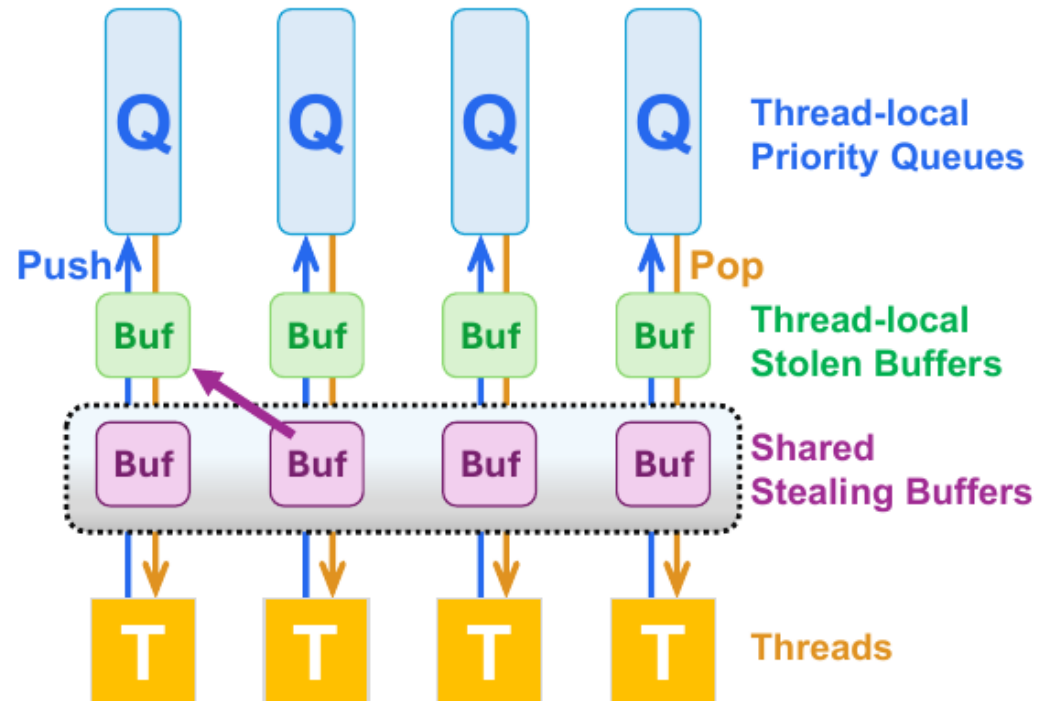
We can change our scheduler & apply optimizations to improve BP performance

Our Approach

Combining a state-of-the-art priority scheduler
& lazy-updates with belief propagation

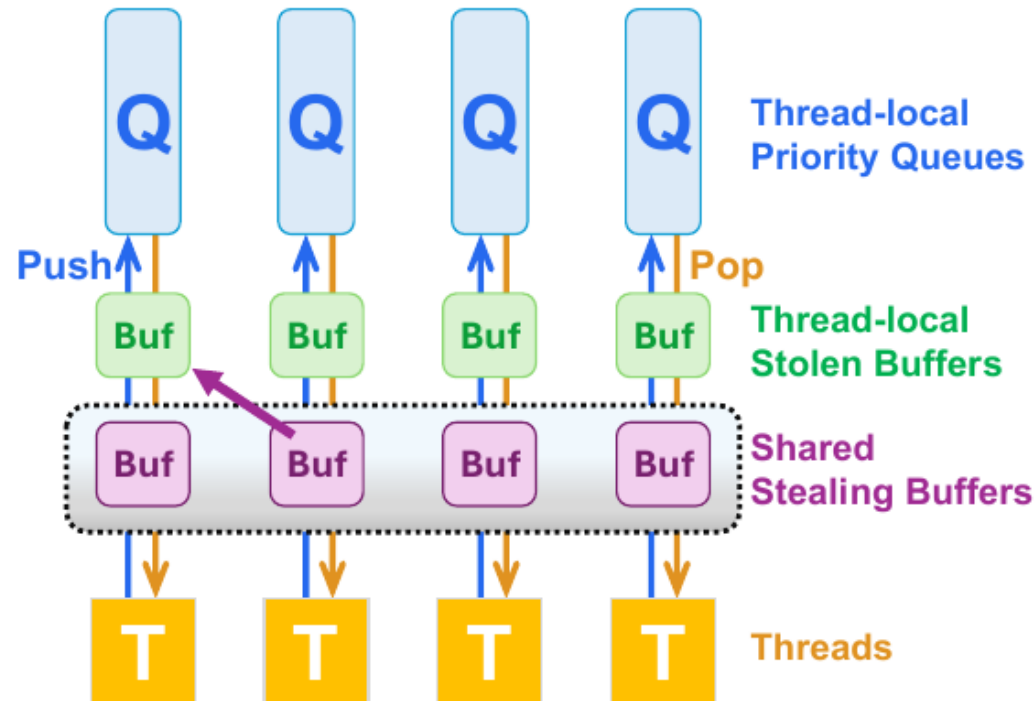
Are there other state-of-the-art schedulers?

Stealing MultiQueue (Postnikova et al., 2022)



Are there other state-of-the-art schedulers?

Stealing MultiQueue (Postnikova et al., 2022)



More thread-local accesses should reduce time on queue accesses

What if we could avoid unneeded updates?

- MQ & SMQ do not easily support priority updates
- Possible workaround: use push-pop queue, replace all updates with inserts, & discard stale pops
- Issue: increases queue pressure

What if we could avoid unneeded updates?

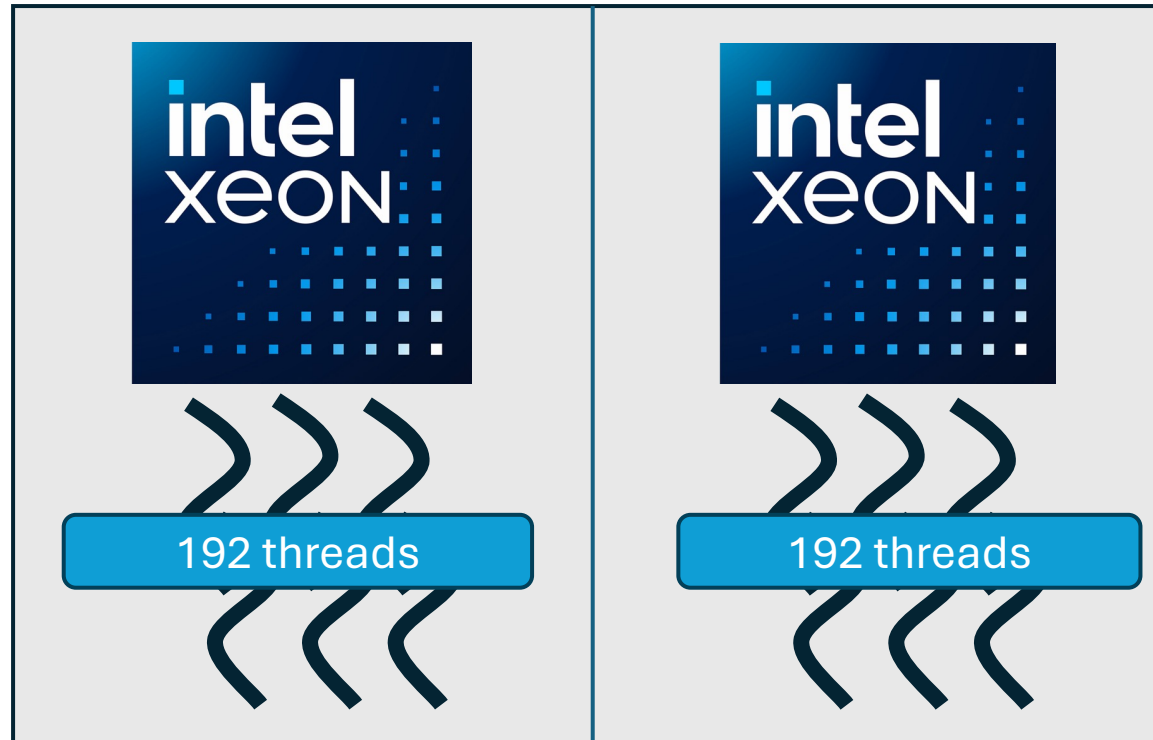
- MQ & SMQ do not easily support priority updates
- Possible workaround: use push-pop queue, replace all updates with inserts, & discard stale pops
- **Issue: increases queue pressure**

Postpone Pattern keeps queue small by deferring pushes if priority decreases:

- Store current node priorities in separate array
- Updating neighbour:
 - Update priorities array
 - Only push to queue if priority increased
- Stale pop:
 - Only push to queue if priority increased

Experimental Methodology

How do we demonstrate speed **and** scalability?

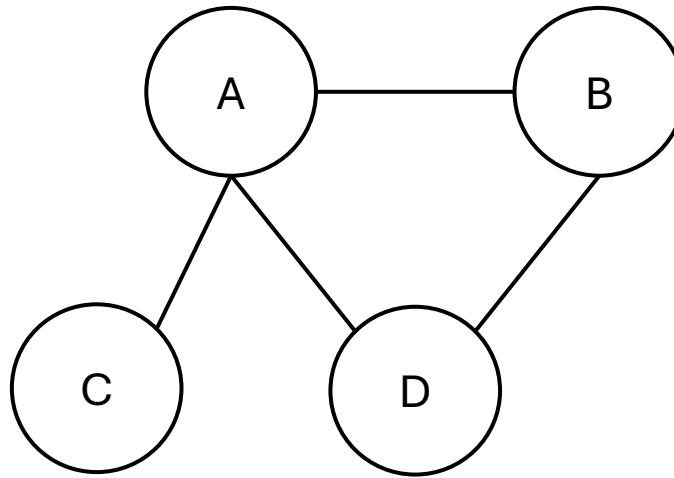


We measure performance on 384 threads vs. 70 in prior* BP work

* (Aksenov et al., 2020)

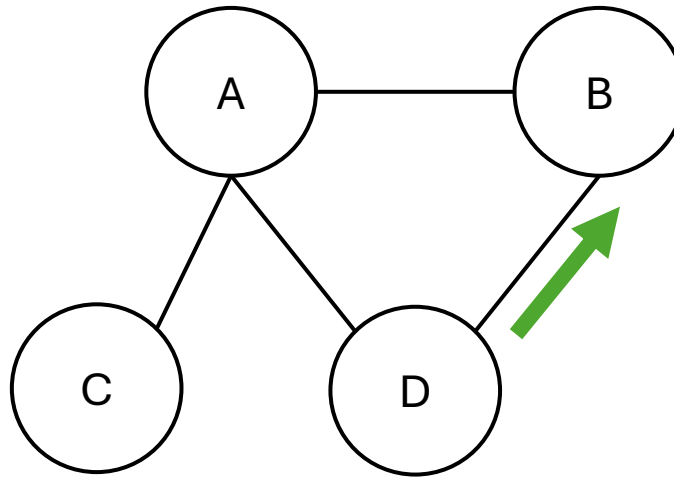
Which BP algorithms do we evaluate?

1. Single-thread baseline: Residual BP



Which BP algorithms do we evaluate?

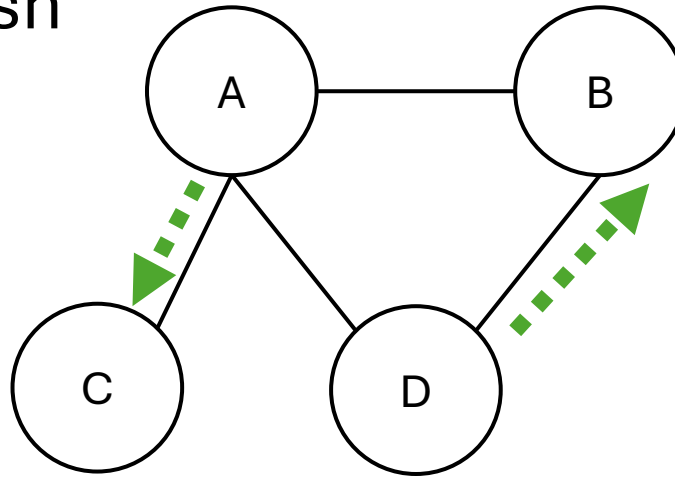
1. Single-thread baseline: Residual BP



Residual BP updates the message with the highest priority

Which BP algorithms do we evaluate?

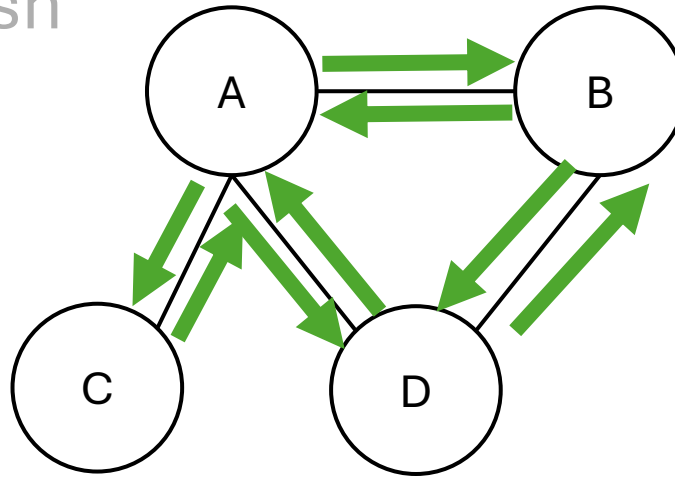
1. Single-thread baseline: Residual BP
2. Relaxed residual BP
3. Relaxed smart splash



Relaxed BP algorithms update one of the top k priority messages

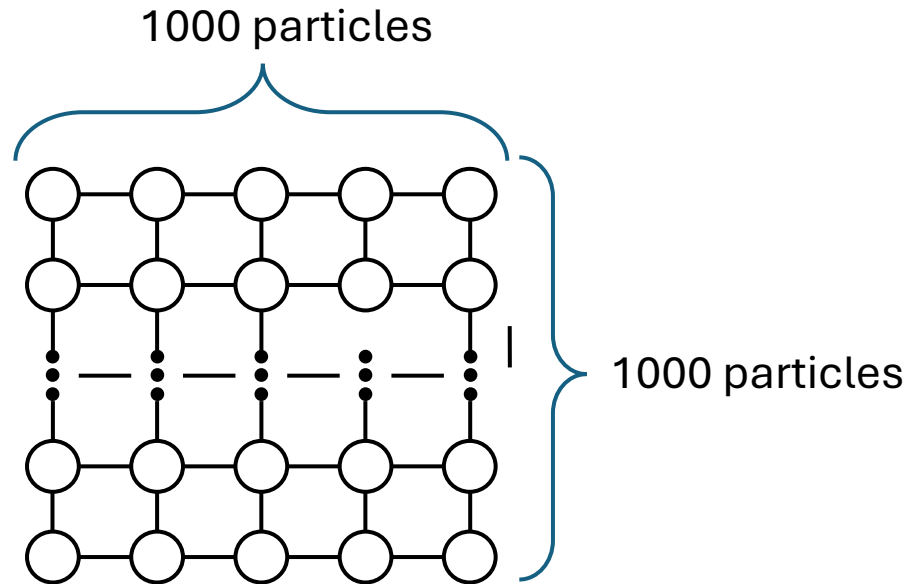
Which BP algorithms do we evaluate?

1. Single-thread baseline: Residual BP
2. Relaxed residual BP
3. Relaxed smart splash
4. Synchronous BP



Synchronous BP updates *all* messages

What input MRFs do we evaluate on?



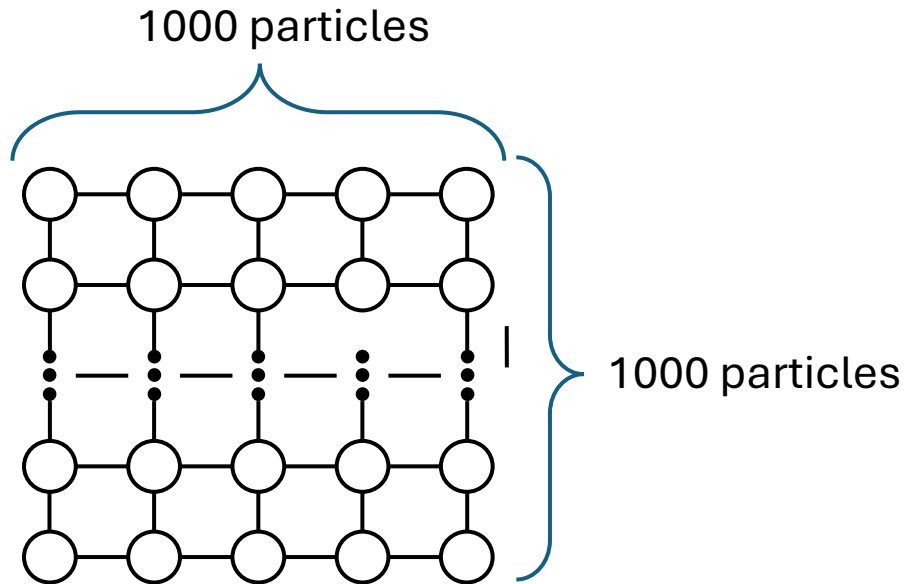
Ising

Models spin of +1 or -1

Potts

>2 possible states

What input MRFs do we evaluate on?

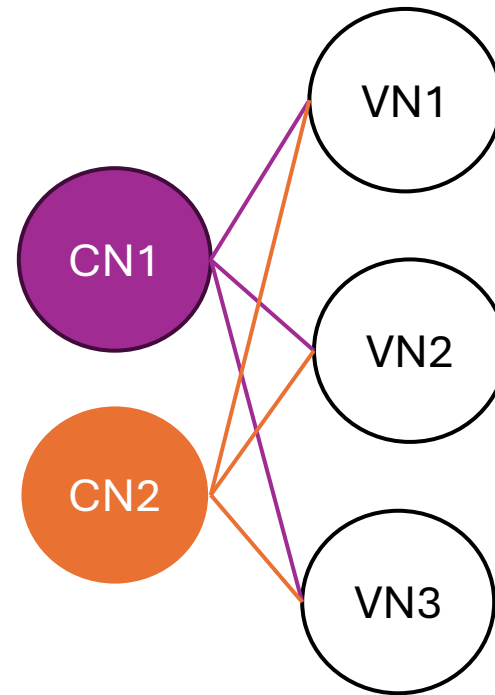


Ising

Models spin of +1 or -1

Potts

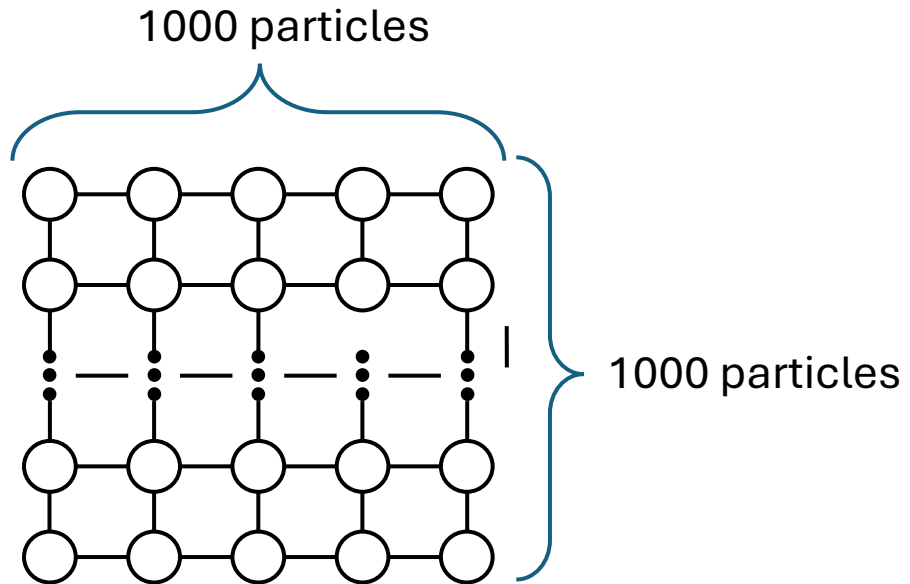
>2 possible states



Low-Density Parity Check (LDPC) codes

error correction using
check & variable nodes

What input MRFs do we evaluate on?

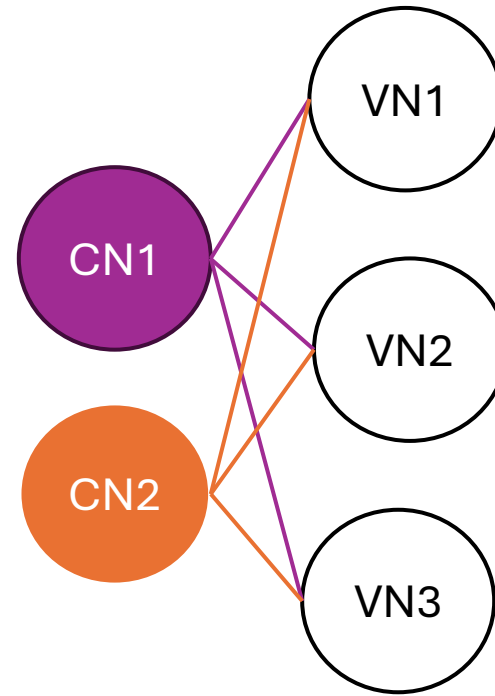


Ising

Models spin of +1 or -1

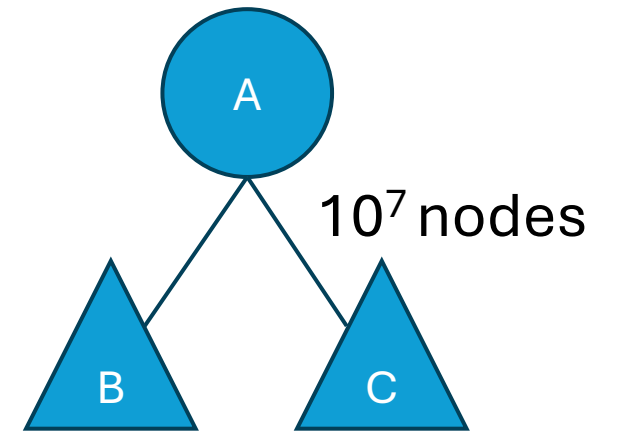
Potts

>2 possible states



Low-Density Parity Check (LDPC) codes

error correction using
check & variable nodes

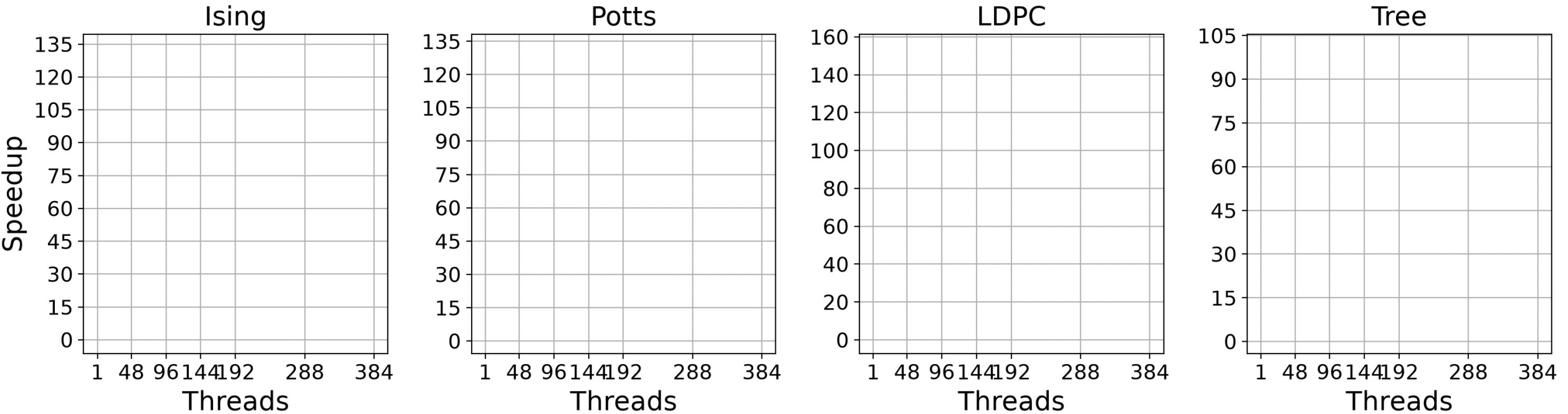


Tree

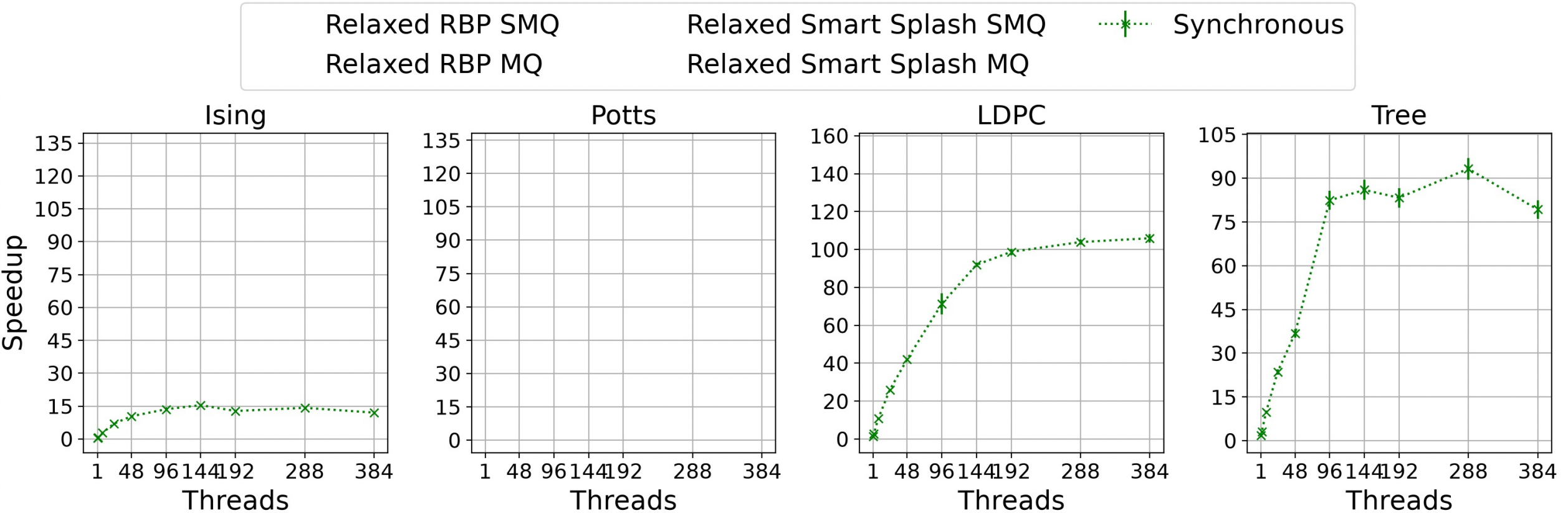
Randomized & binary

Experimental Results

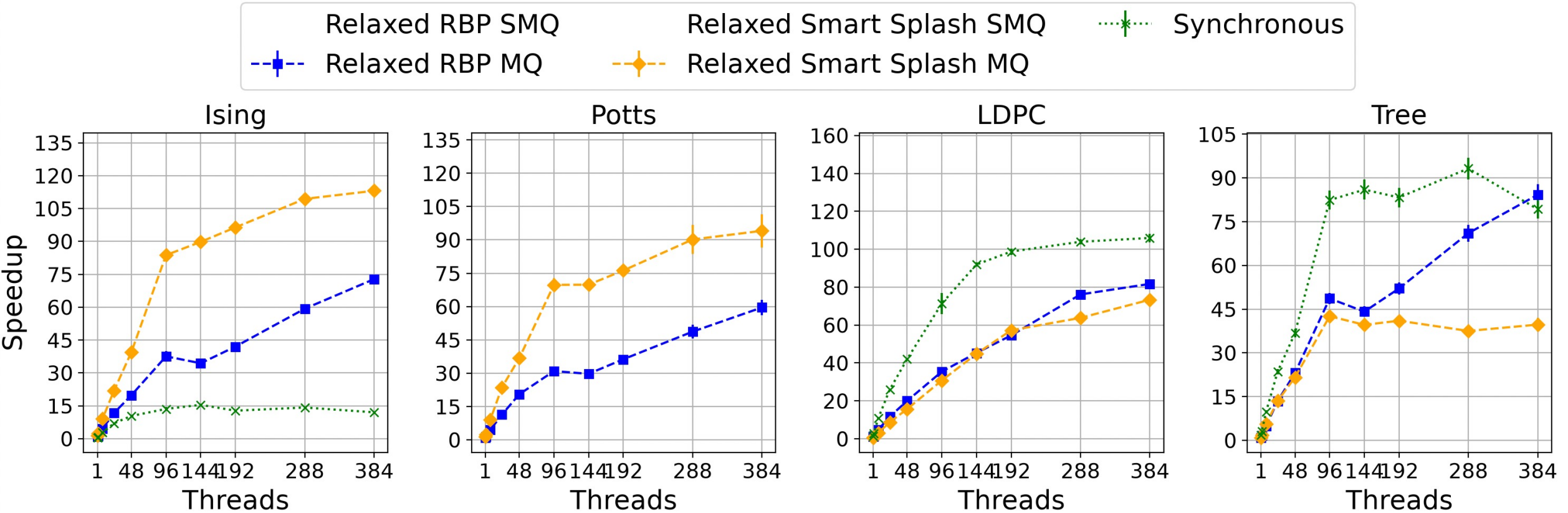
We compare performance up to 384 threads



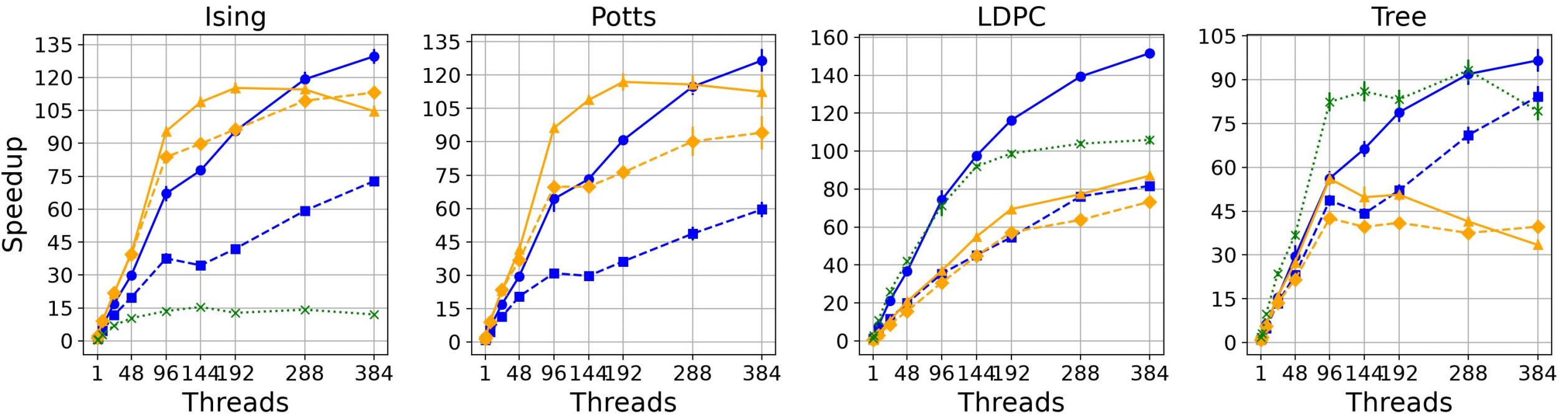
We compare performance up to 384 threads



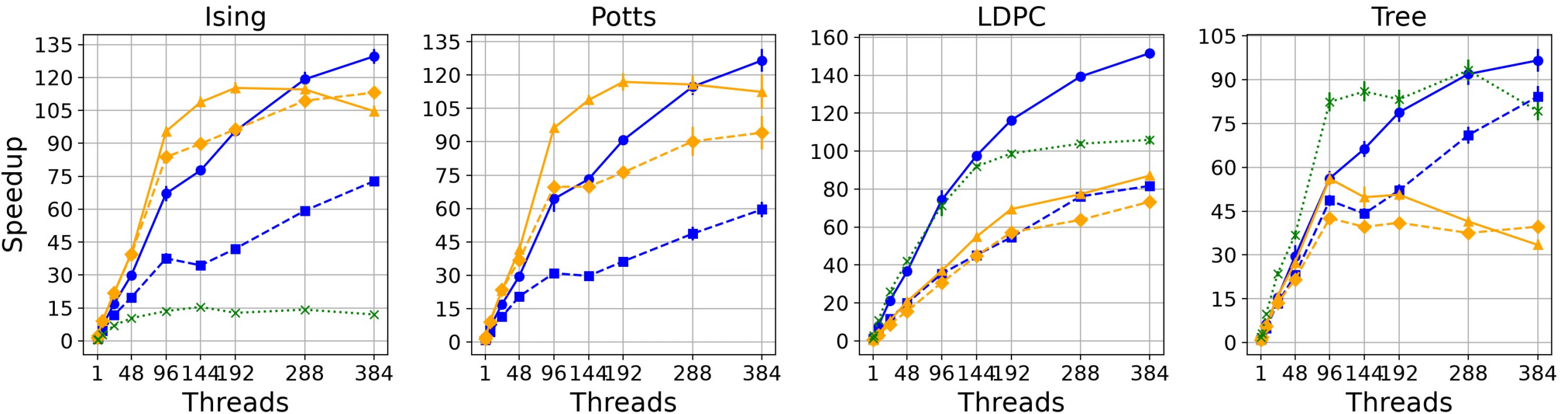
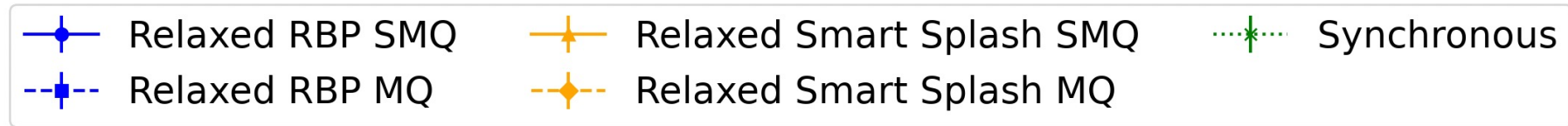
We compare performance up to 384 threads



We compare performance up to 384 threads

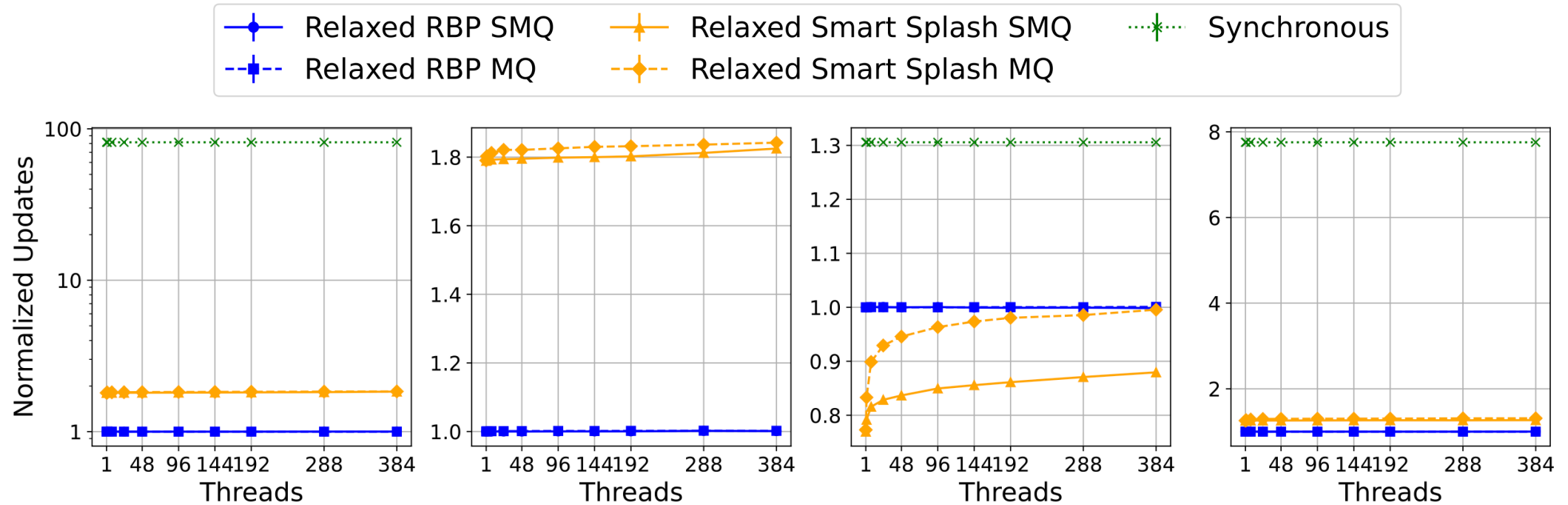


We compare performance up to 384 threads

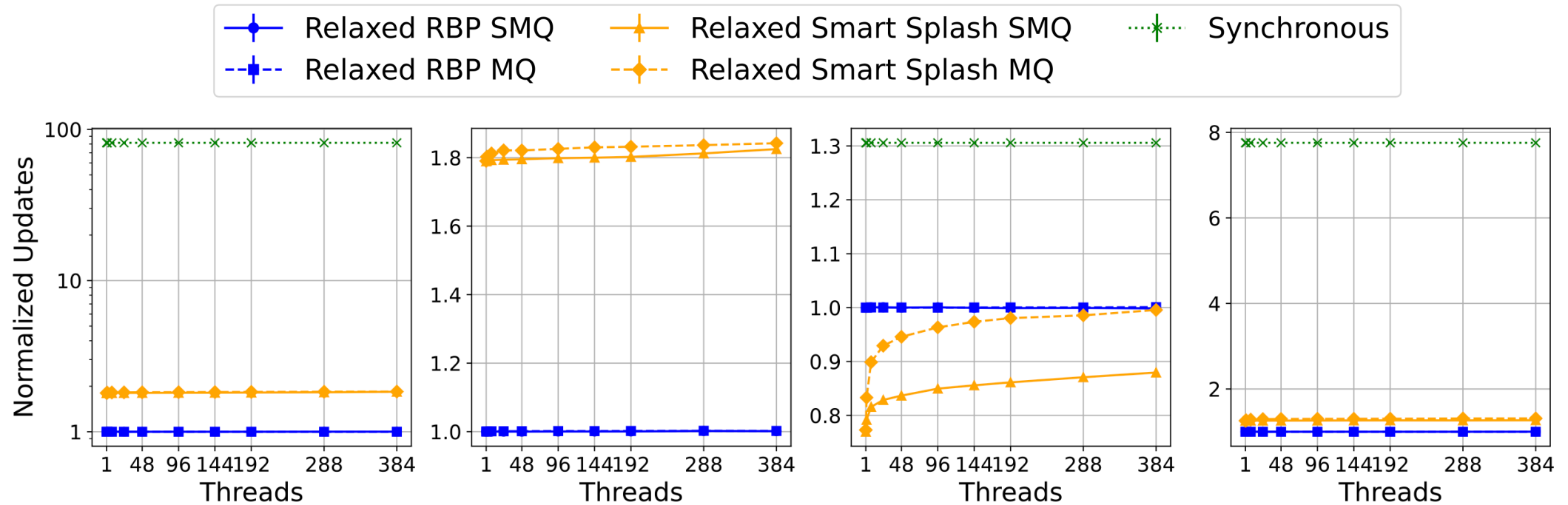


Our approach outperforms prior work (geomean 2.66× speedup)

We avoid excessive updates

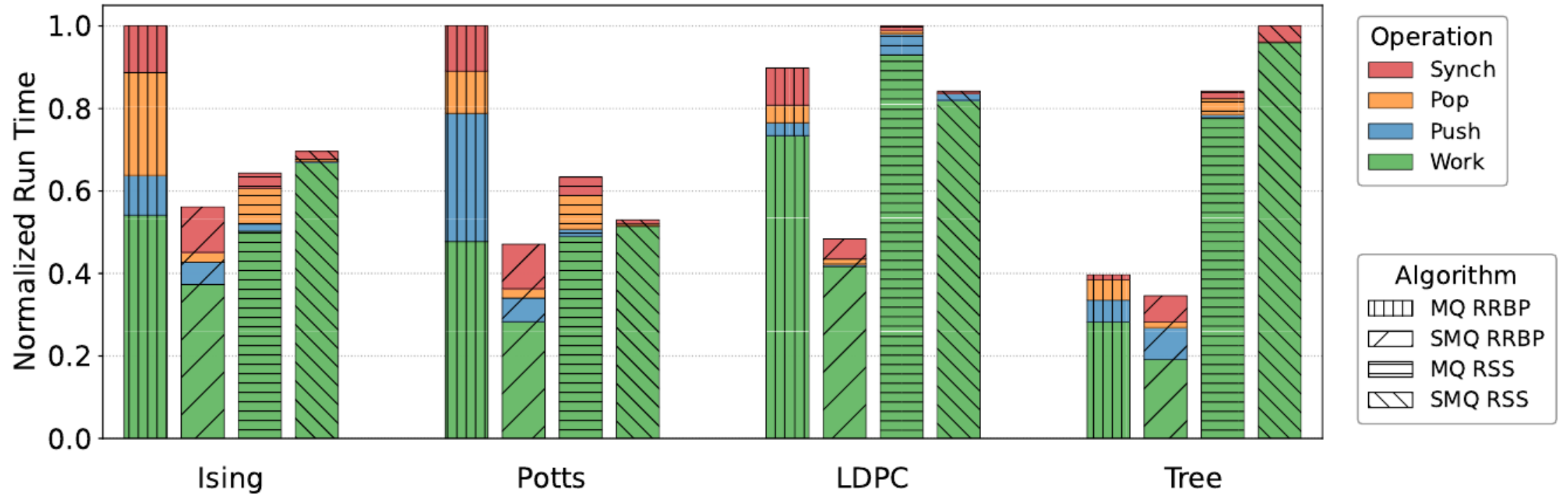


We avoid excessive updates

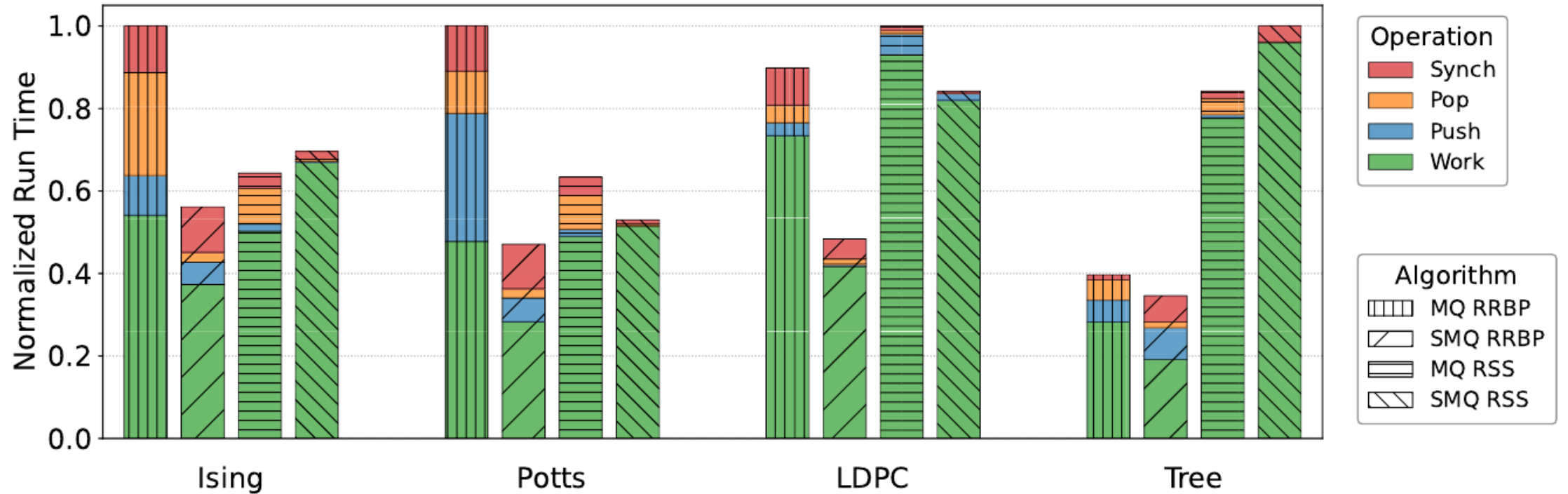


Our approach has a high convergence rate

Where does time go now?



Where does time go now?



Queue overheads decrease across MRFs

Optimized Priority Scheduling for Faster Scalable Belief Propagation

*Carnegie
Mellon
University

Abnash Bassi*, Gilead Posluns[†], Mark C. Jeffrey[†]
Probabilistic Graphical Models (PGM) 2026



Future Work: ablation studies & experiments on non-CPU hardware

Our Contributions:

1. Address BP's parallelism bottleneck by adopting a state-of-the-art priority scheduler and a lazy-update optimization
2. Identify one winning algorithm across inputs
3. First known work to scale parallel BP up to 384 threads $\rightarrow$ $>1.6\times$ per input type and geomean $2.66\times$ speedup

GitHub Repo:

