Message Scheduling for Performant, Many-Core Belief Propagation

Mark Van der Merwe, Vinu Joseph, Ganesh Gopalakrishnan

School of Computing, University of Utah

THE UNIVERSITY OF UTAH®

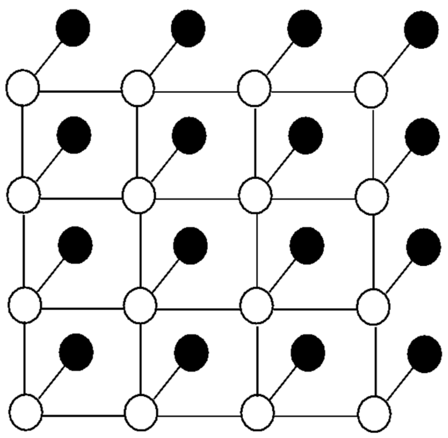
Overview

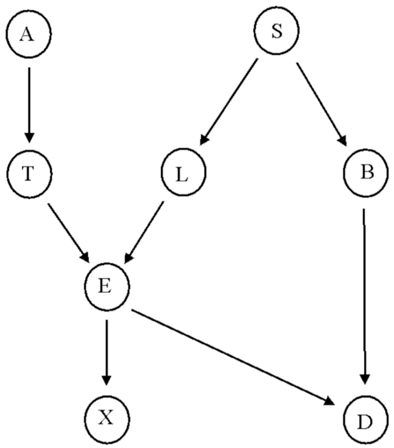
- What is Belief Propagation (BP)?
  - The Algorithm.
  - Performance: Convergence, Speed, Accuracy.
  - Message Scheduling and its Effect on Performance.
- Frontier-Based BP on the GPU.
  - Residual-Based Message Scheduling on the GPU.
  - Randomized Message Scheduling on the GPU.

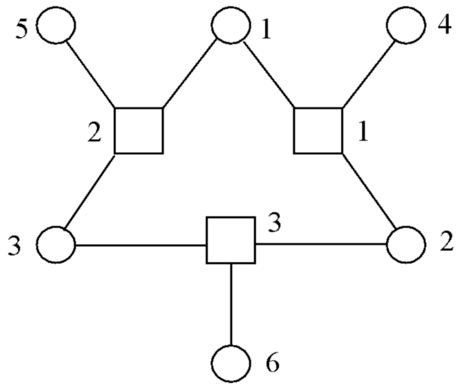
THE UNIVERSITY OF UTAH®

Probabilistic Graphical Models

- Encode a joint distribution over a set of variables.
- Vertices are random variables, edges are probabilistic relationships.

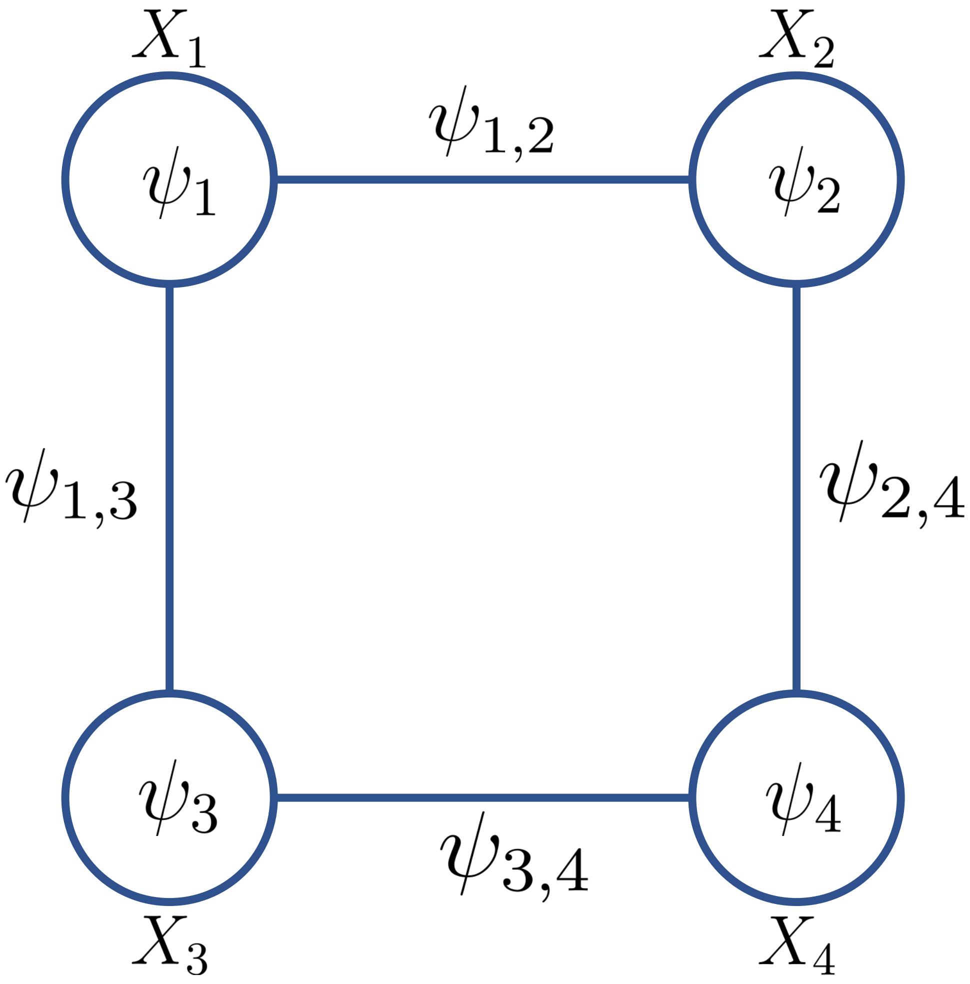






Source: *Understanding BP and its Generalizations, Jonathan Yedidia, William Freeman, Yair Weisss*

Markov Random Field (MRF)



- Undirected Graphical Model:
  - Random Variables:
    $X = \{X_1, \ldots, X_n\}; X_i \in A_i$
  - Undirected Graph:
    $G = (V, E)$
  - Parameterized by:
    $\{\psi_i : A_i \to \mathbb{R}^+ | i \in V\}$
    $\{\psi_{i,j} : A_i \times A_j \to \mathbb{R}^+ | (i, j) \in E\}$
  - Joint Distribution:
    $P(x_1, \ldots, x_N) \propto \prod_{i \in V} \psi_i(x_i) \prod_{\{i,j\} \in E} \psi_{i,j}(x_i, x_j)$

THE UNIVERSITY OF UTAH®

Inference

- Derive vertices's marginal distributions.

$$P(x_1, ..., x_N) \propto \prod_{i \in V} \psi_i(x_i) \prod_{\{i,j\} \in E} \psi_{i,j}(x_i, x_j)$$

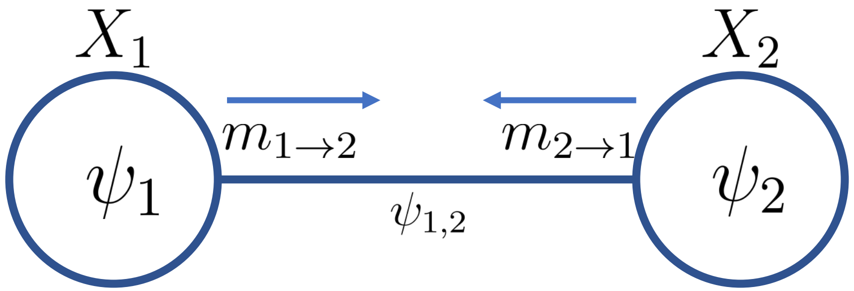
$$\Big\downarrow$$

$$P(x_i) \; \forall i \in V$$

- In general, exact inference is intractable. Enter approximate inference.

Belief Propagation

- Message Passing Algorithm for *Approximate* Inference:



- Messages indicate a vertex's belief about another vertex:

$$m_{i\to j}(x_j) \propto \sum_{x_i \in A_i} \psi_{i,j}(x_i, x_j)\psi_i(x_i) \prod_{k\in\Gamma_i \backslash j} m_{k\to i}(x_i)$$

- Local beliefs updated according to messages:

$$P(X_i = x_i) \approx b_i(x_i) \propto \psi_i(x_i) \prod_{k\in\Gamma_i} m_{k\to i}(x_i)$$

- Exact on tree graphs; approximate on "loopy" graphs.
- Iterative Convergent Algorithm.

THE UNIVERSITY OF UTAH®

Belief Propagation

- BP enables highly *general* inference over *complex* systems.
- Parallelism is key:
  - Faster is better!! Enable real-time applications (e.g. robotics).
  - Larger scale is better!! Enable internet-age modeling (e.g. hyperlink network).

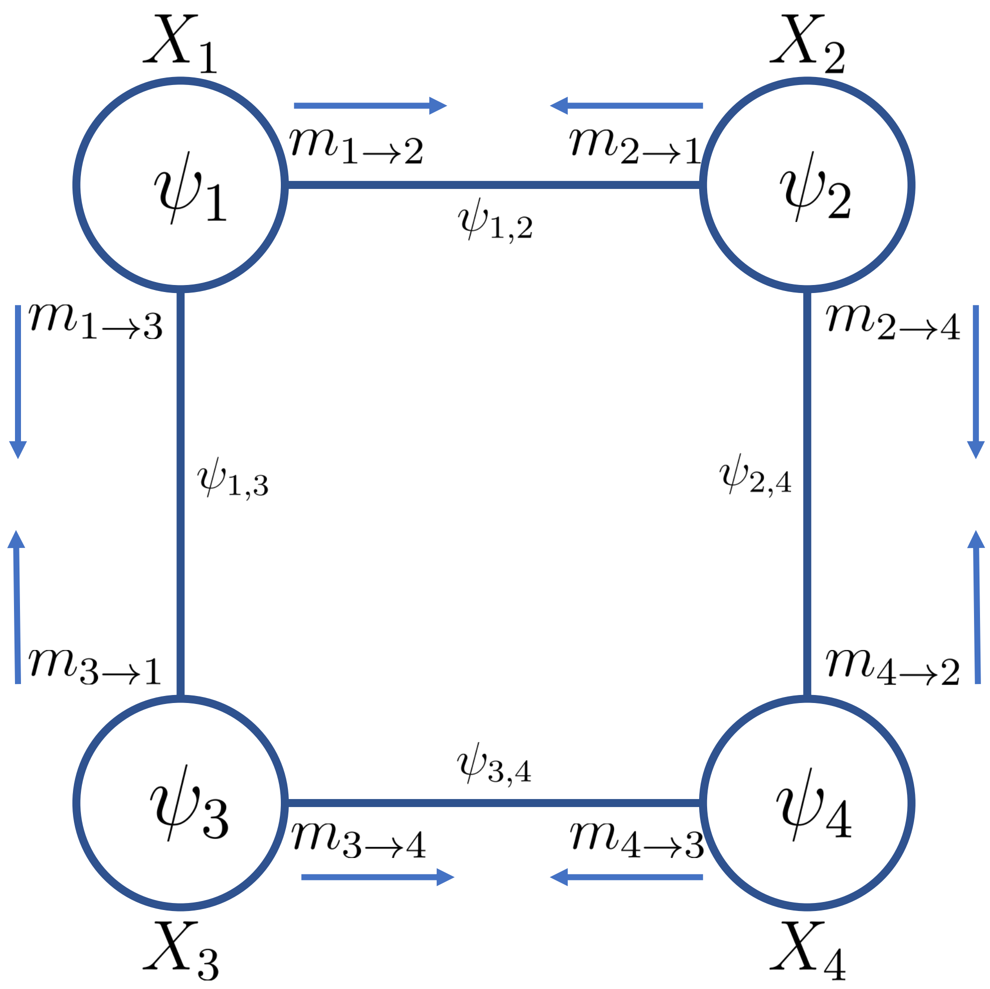
GPU BP!!

Performant Belief Propagation

- What defines *performant* Belief Propagation?

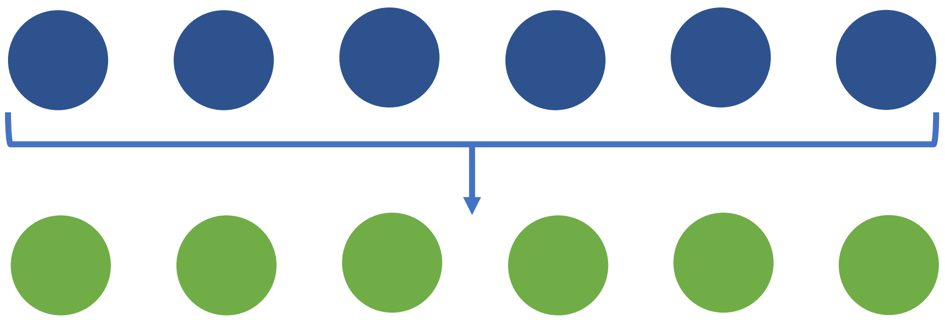
1. Accurate – are the results close to the true marginals?
2. Convergent – does the algorithm complete?
3. Fast – how fast does the algorithm complete?

Message Scheduling



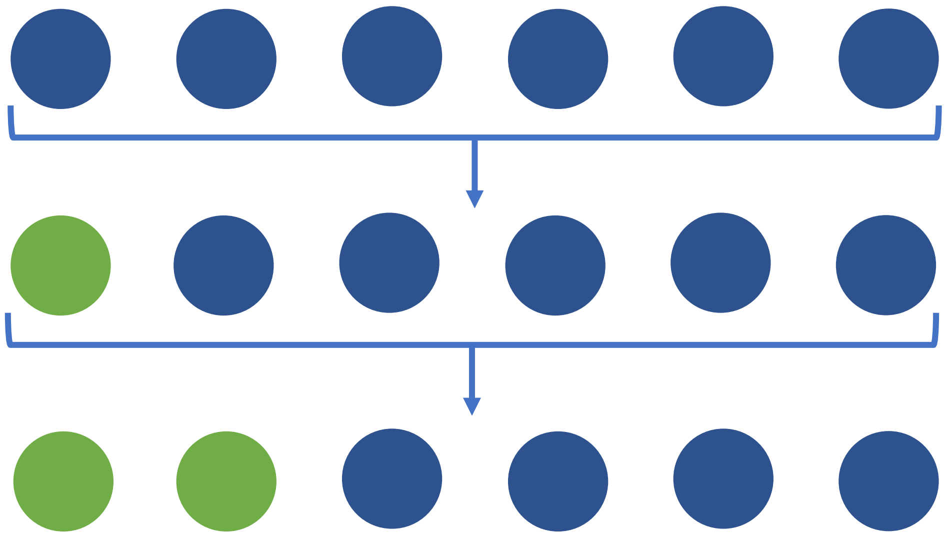
- In what order should the messages be passed to enable best performance?

Message Scheduling



Loopy Belief Propagation (LBP)
(AKA, *Synchronous* BP)

Can converge at times, but fails to on many graphs [1,2].



Asynchronous BP
Better convergence than LBP [3]!

THE UNIVERSITY OF UTAH®

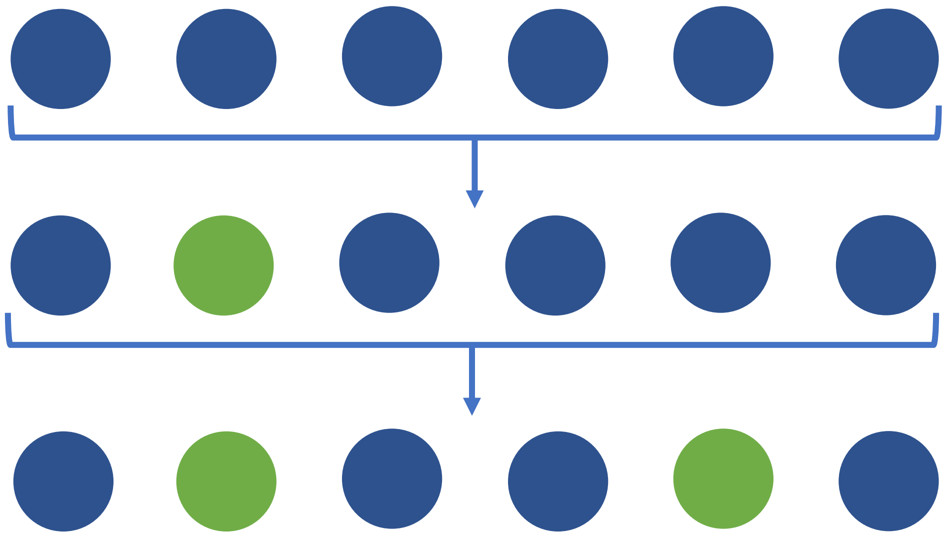
[1] *Loopy BP for Approximate Inference, Kevin Murphy, Yair Weiss, Michael Jordan*
[2] *Sufficient Conditions for Convergence of the Sum-Product Algorithm, Joris Mooij, Hilbert Kappen*
[3] *Residual Belief Propagation, Gal Elidan, Ian McGraw, Daphne Koller*

Message Scheduling

- Intuition: messages that don't update much are less important.
- Simple selection metric [3]:

$$r(m_i^t) = ||f_{BP}(m^t)_i - m_i^t||$$

- Select next message with Priority Queue: *greedy* scheduling.

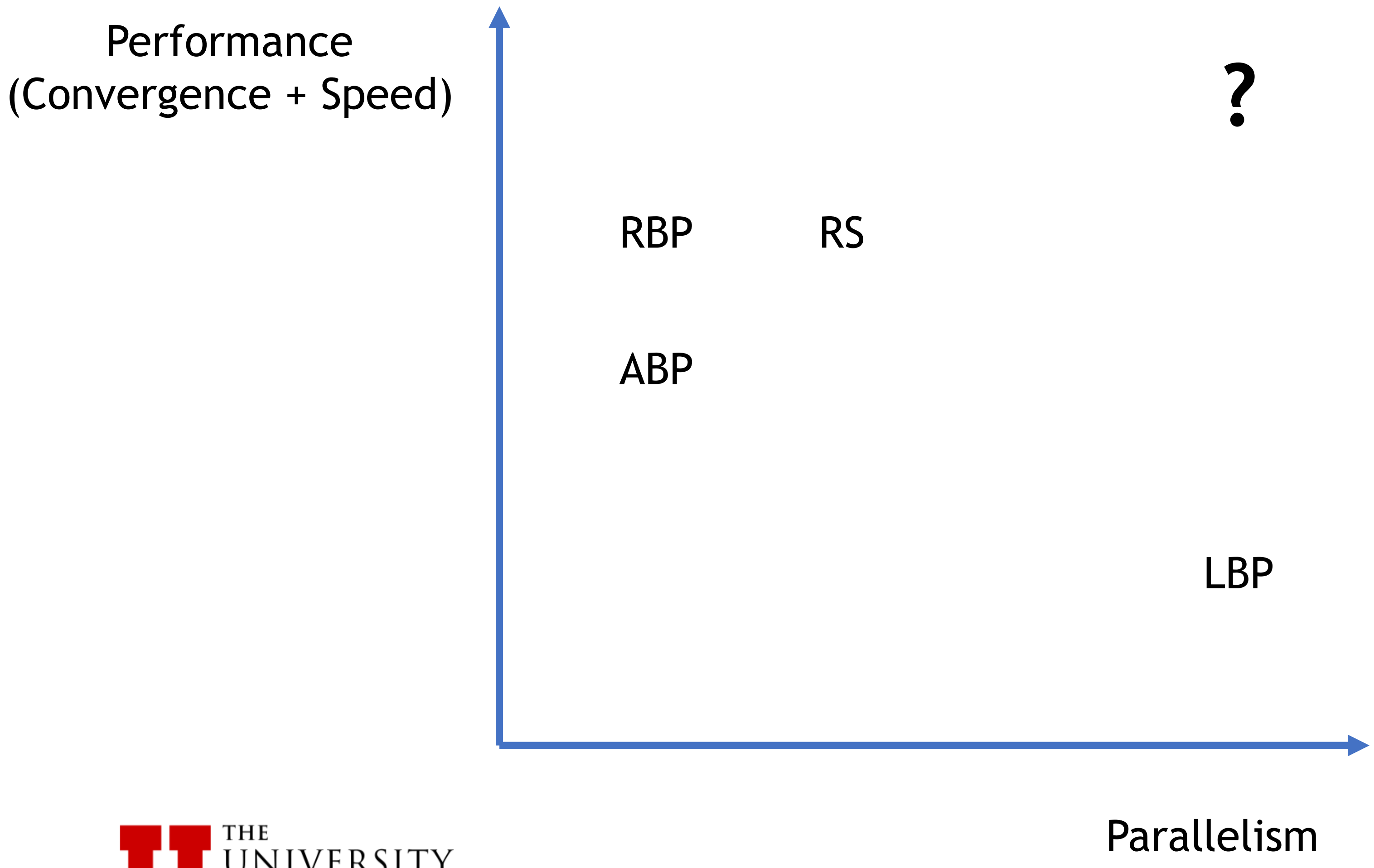


Residual BP
Better convergence than LBP/ABP [3]!

Select vertices; Residual Splash [4].

[3] *Residual Belief Propagation, Gal Elidan, Ian McGraw, Daphne Koller*
[4] *Residual Splash for Optimally Parallelizing Belief Propagation, Joseph Gonzalez, Yucheng Low, Carlos Guestrin*

Message Scheduling



Hypothesis:

There exists a tradeoff between parallelism and sequentialism in BP and GPUs can effectively harness it for performant BP.

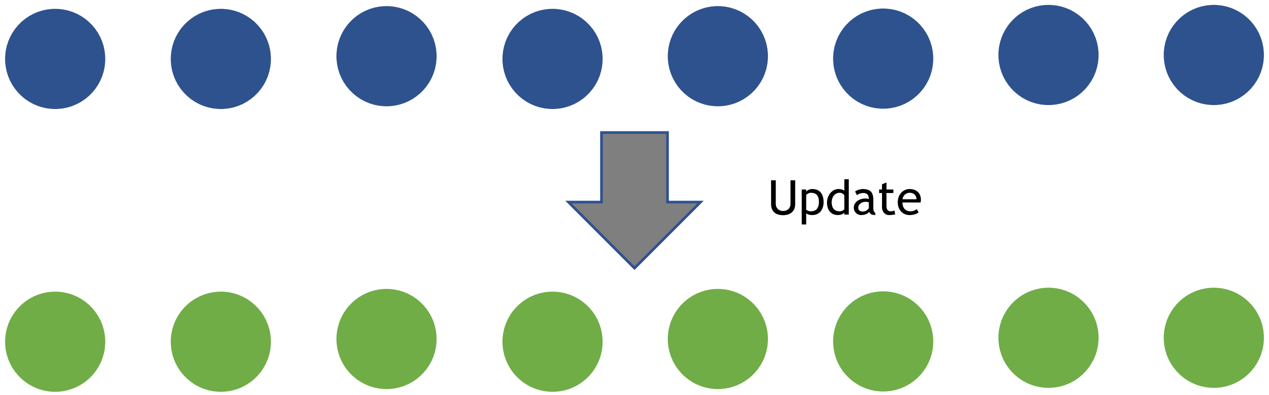
THE UNIVERSITY OF UTAH®

Frontier-Based BP

- GPUs are powerful – lots of parallelism.
- Thrive with bulk parallel, synchronous operations.
- Make a frontier of messages to update in bulk synchronous step.
- Changing the size/selection of this frontier lets us explore message scheduling on GPU.

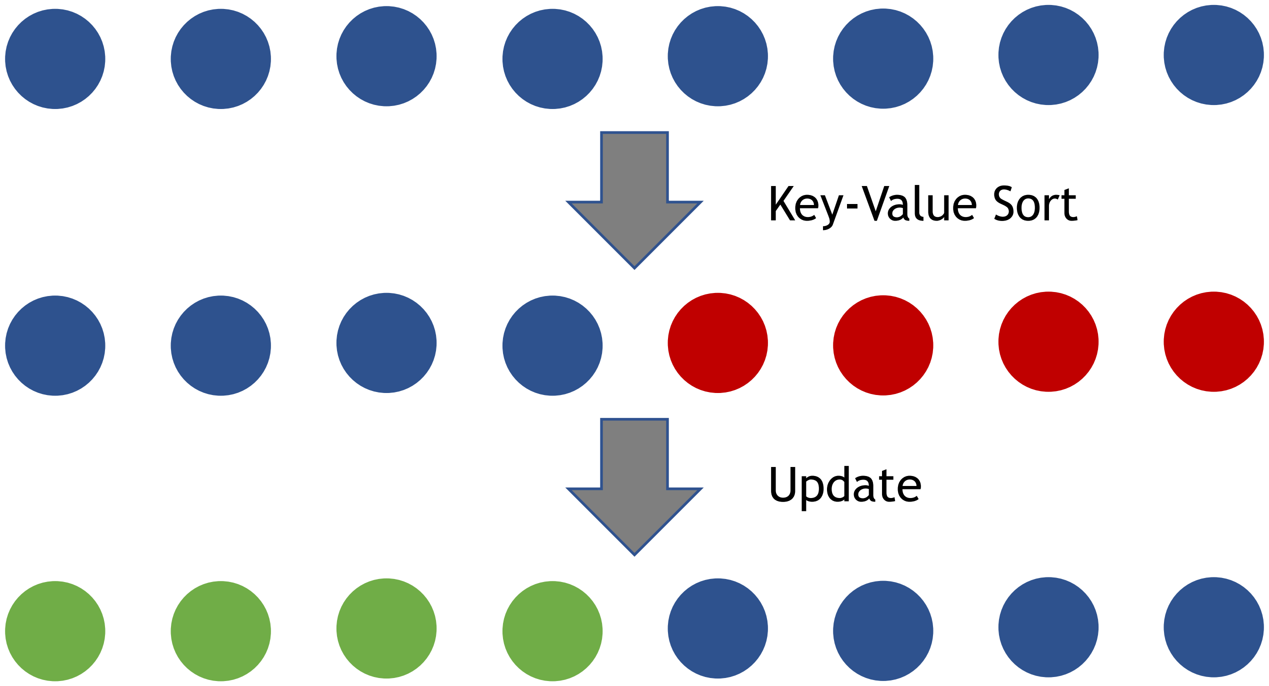
GPU LBP

Frontier = All Messages



GPU Residual BP

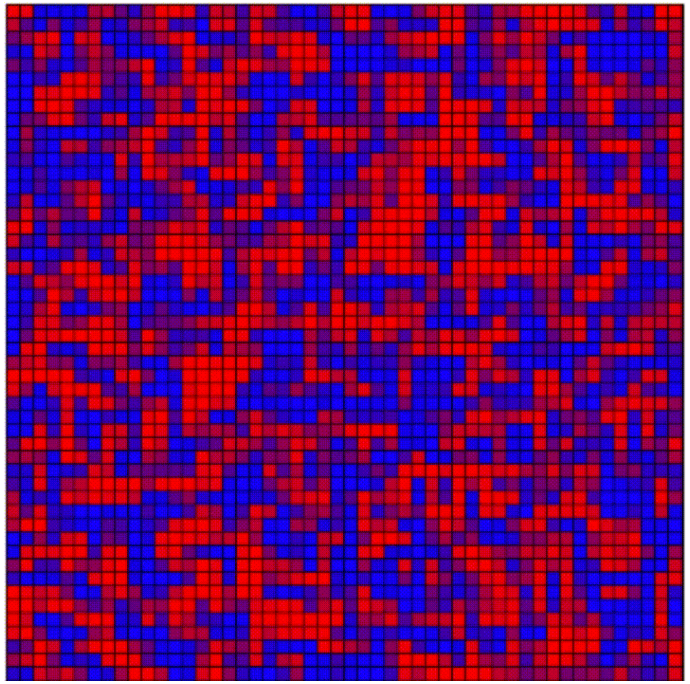
Frontier = Top-*k*



Implementation

- Implement GPU LBP, RBP, RS using Nvidia CUDA.
  - Simple Adjacency List format.
  - Edge/Vertex assigned IDs – threads assigned subset to update.
- RBP/RS: Select *k* via a multiplier *p*.
- Implement Serial RBP (SRBP) for baseline.
- Hardware:
  - GPU: Nvidia Tesla V100
  - CPU: Intel Xeon

Experiments



- Ising Grids (NxN binary grids):

$$\psi_i(x_i) = \begin{cases} p & x_i = 0 \\ 1-p & x_i = 1 \end{cases} ; p \in [0,1]$$

$$\psi_{i,j}(x_i, x_j) = \begin{cases} e^{\lambda C} & x_i = x_j \\ e^{-\lambda C} & x_i \neq x_j \end{cases} ; \lambda \in [-0.5, 0.5], C \in \mathbb{R}$$

- N={100,200}, C=2.5, 20 graphs at each setting.

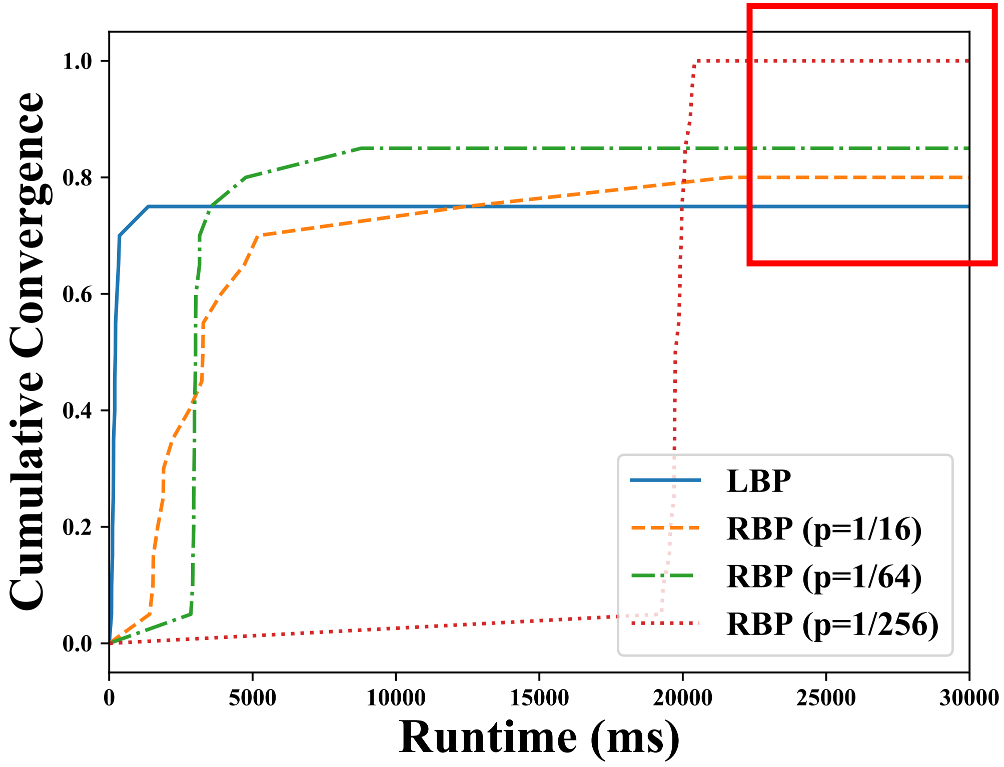
THE UNIVERSITY OF UTAH®

Source: https://personal.utdallas.edu/~nrr150130/ising_demo.html

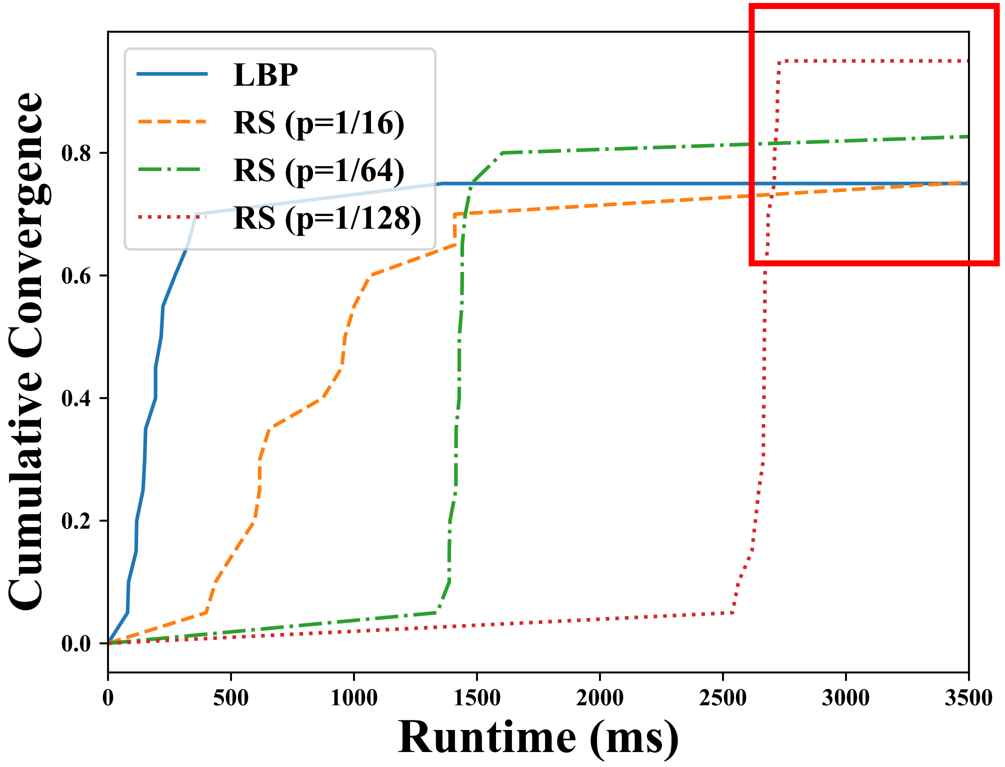
Ising 100x100, C=2.5

As parallelism decreases, convergence *increases*.

GPU RBP



GPU RS

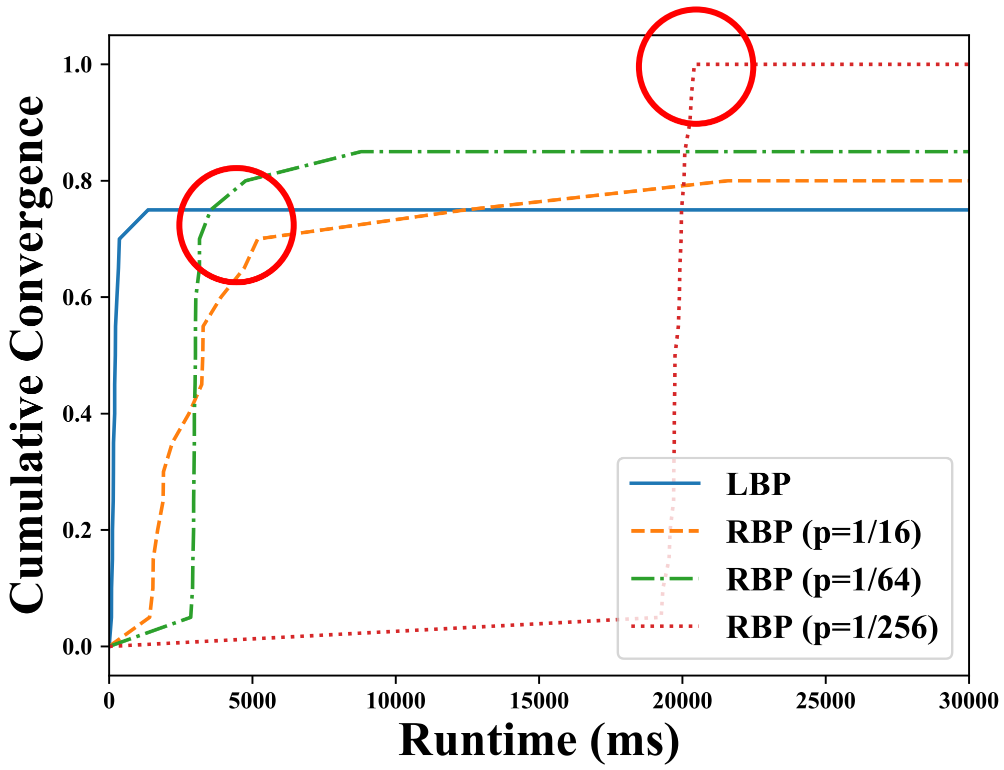


THE UNIVERSITY OF UTAH®

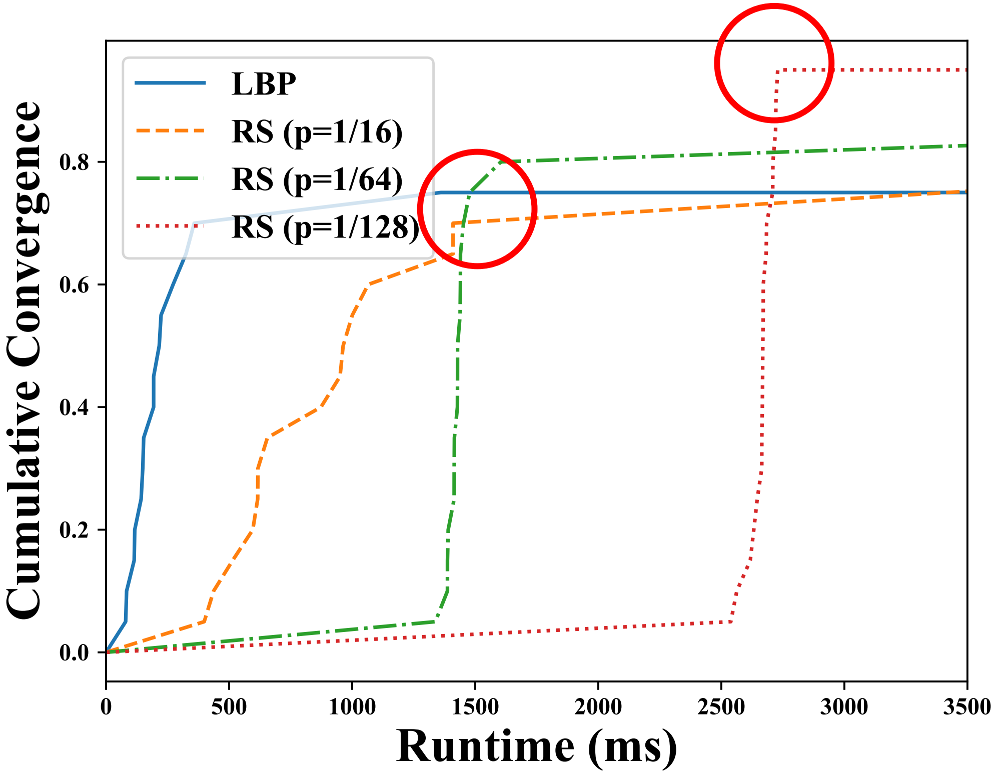
Ising 100x100, C=2.5

As parallelism decreases,
speed *decreases*.

GPU RBP

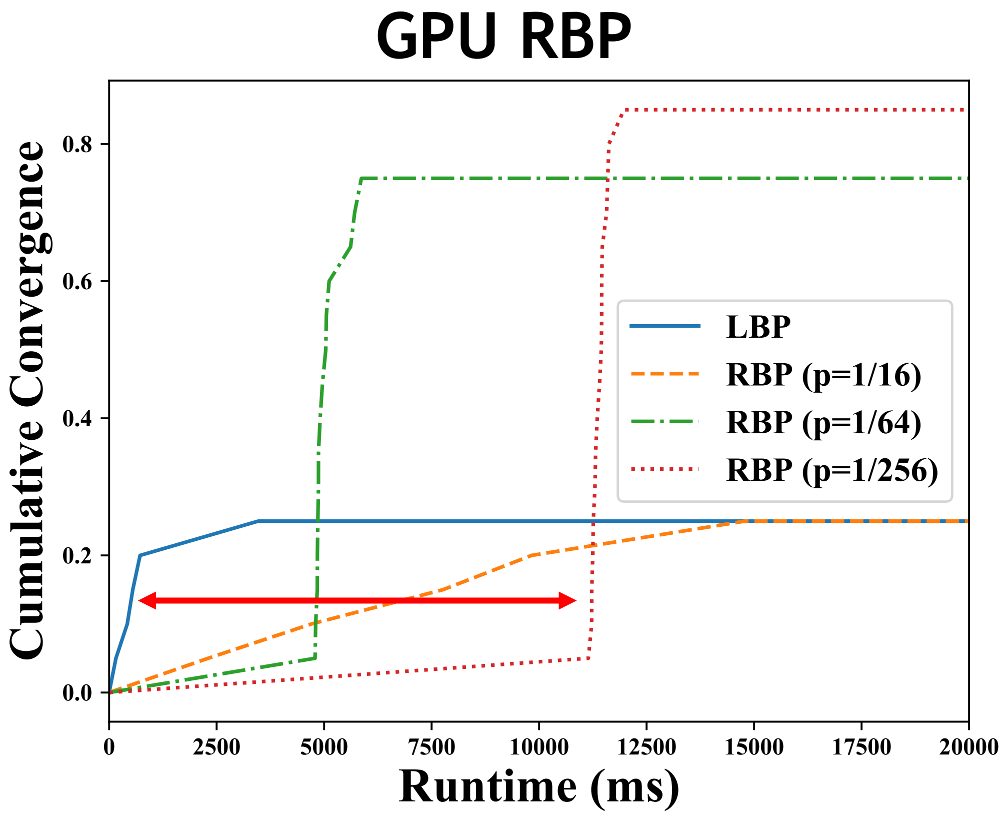


GPU RS

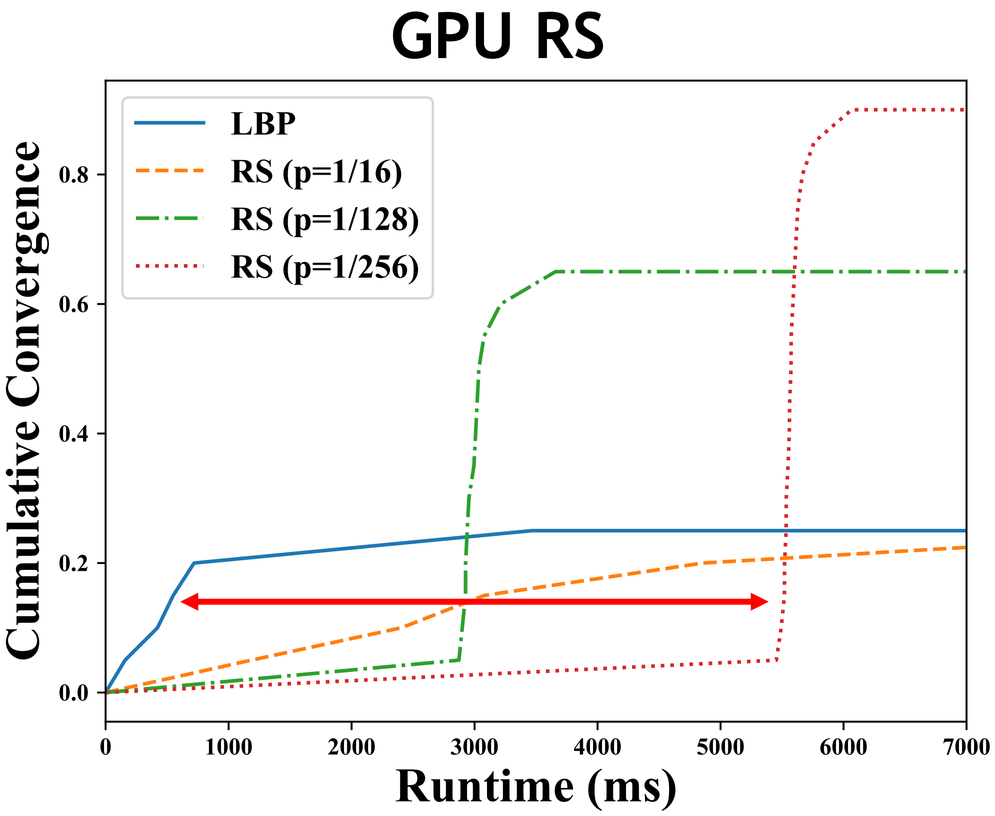


THE UNIVERSITY OF UTAH®

Ising 200x200, C=2.5



Dataset	Settings	SRBP Speedup
Ising 100×100, $C = 2.5$	$p = 1/256$	3.47x
Ising 200×200, $C = 2.5$	$p = 1/256$	$>$ 4.54x



Dataset	Settings	SRBP Speedup
Ising 100×100, $C = 2.5$	$p = 1/128$	25.85x
Ising 200×200, $C = 2.5$	$p = 1/256$	$>$ 16.08x

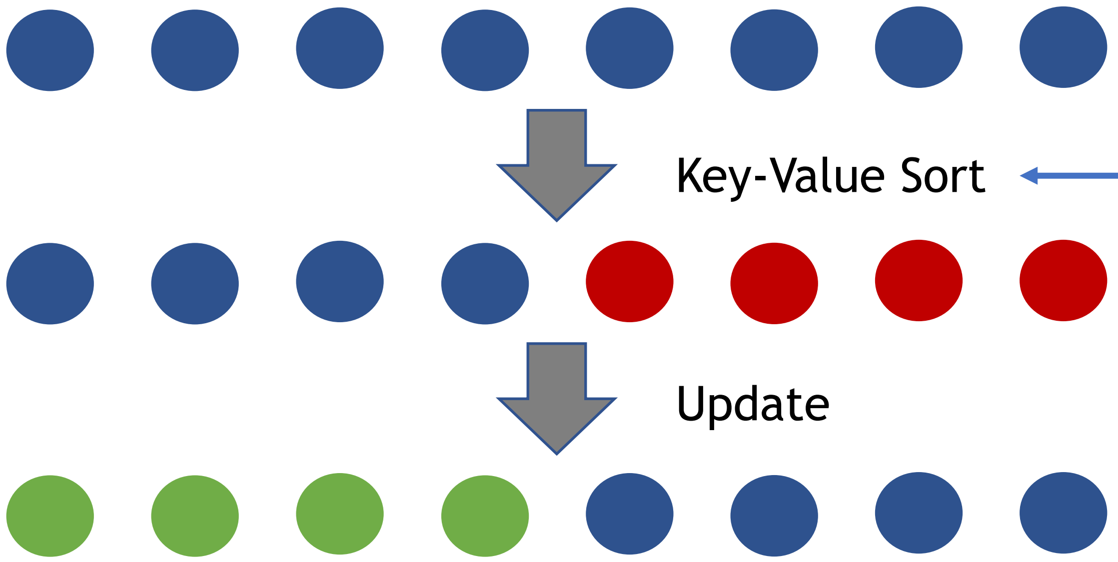
THE UNIVERSITY OF UTAH®

Where are we now?

The hypothesized tradeoff exists! But have we effectively exploited it?

1. Accurate – are the results close to the true marginals? ✓
2. Convergent – does the algorithm complete? ✓
3. Fast – how fast does the algorithm complete? ?

Revisiting GPU Residual BP



Large majority of time spent here.

How important is message selection?

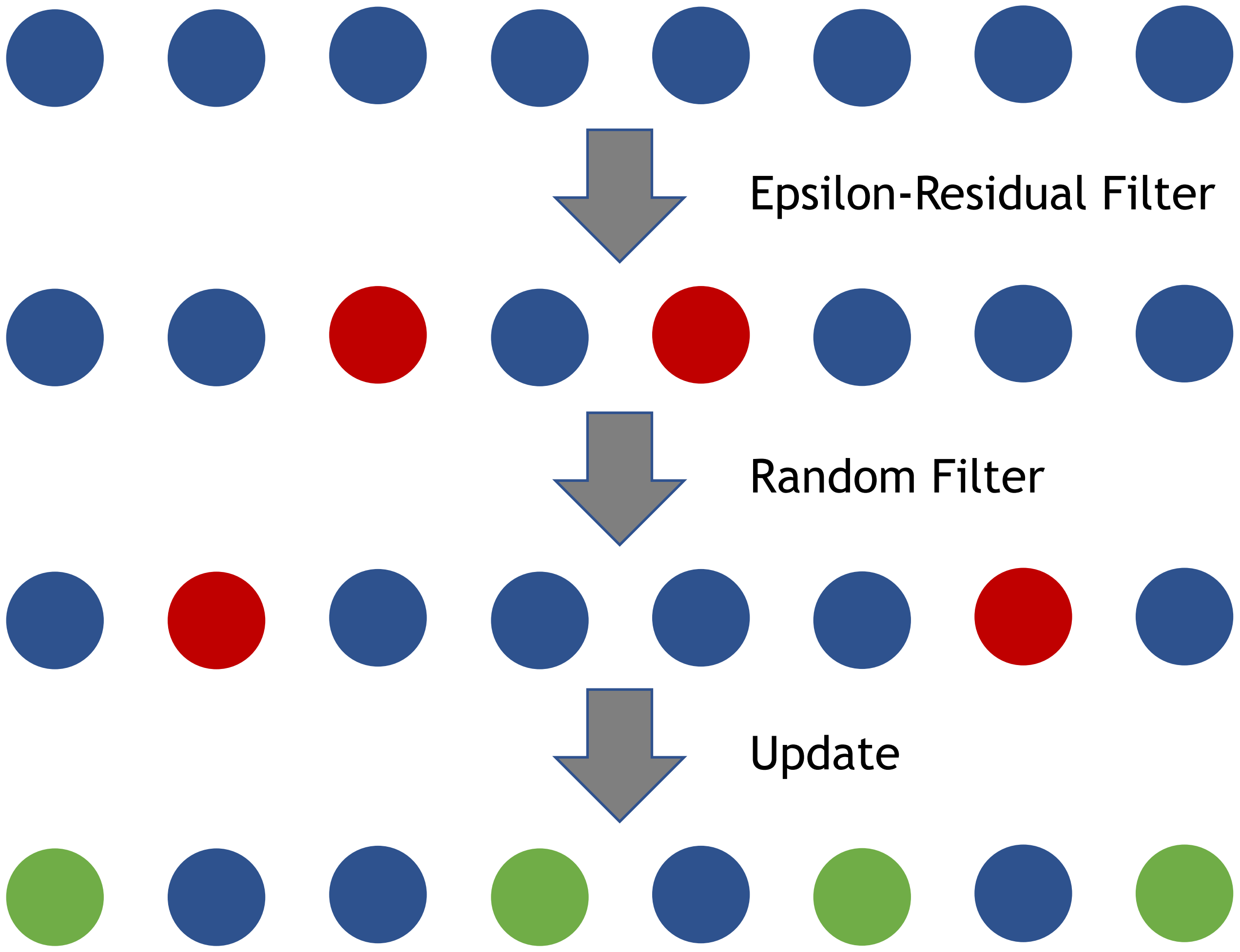
Hypothesis:

Varying the parallelism affects performance *more* than specific selection of messages in a many-core environment.

THE UNIVERSITY OF UTAH®

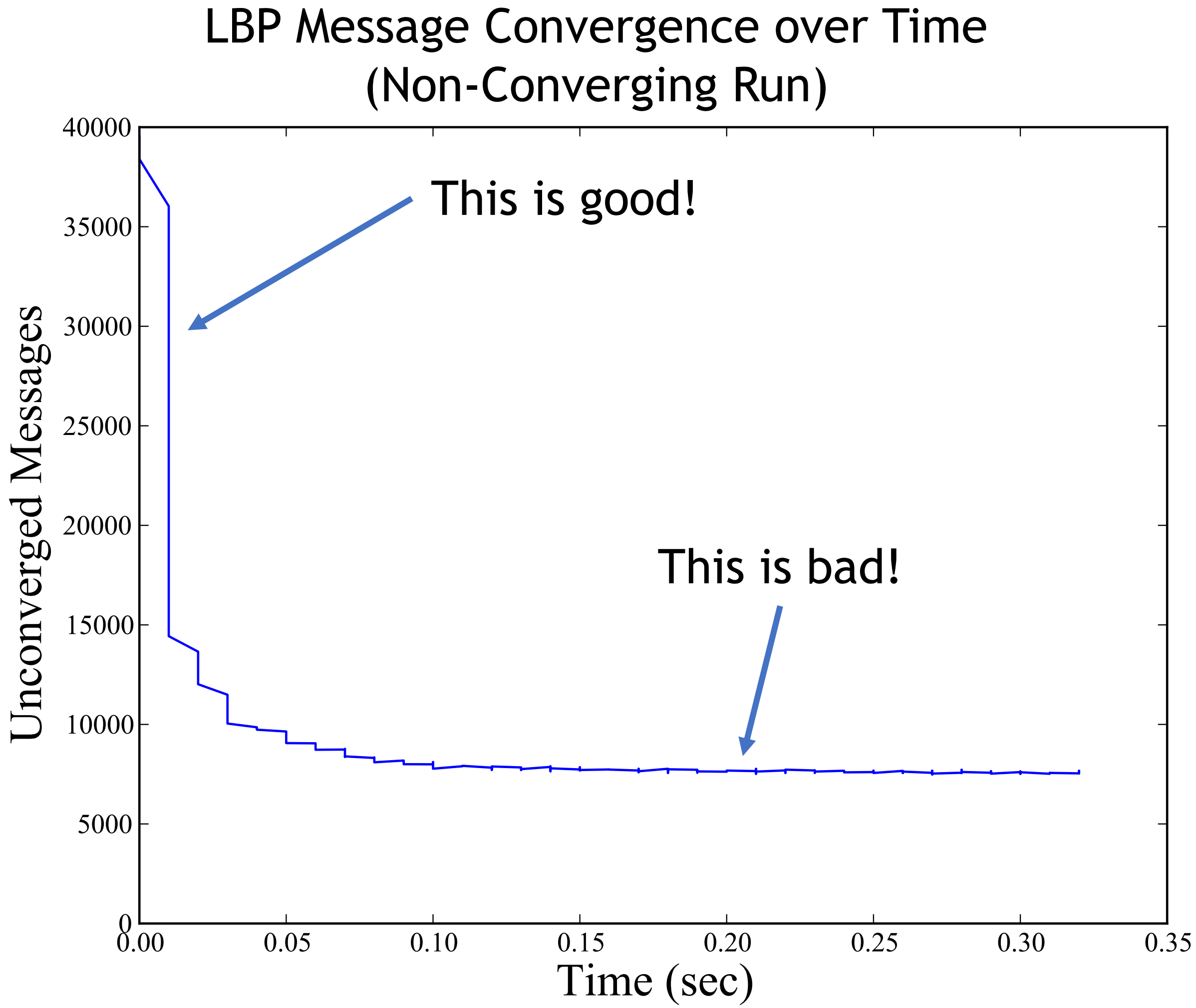
Randomized BP

Frontier = Random-*k*



THE UNIVERSITY OF UTAH®

Randomized BP



Idea: keep parallelism high when messages are converging quickly, make it low when they stop converging.

$$EdgeRatio = \frac{NewEdgeCount}{OldEdgeCount}$$

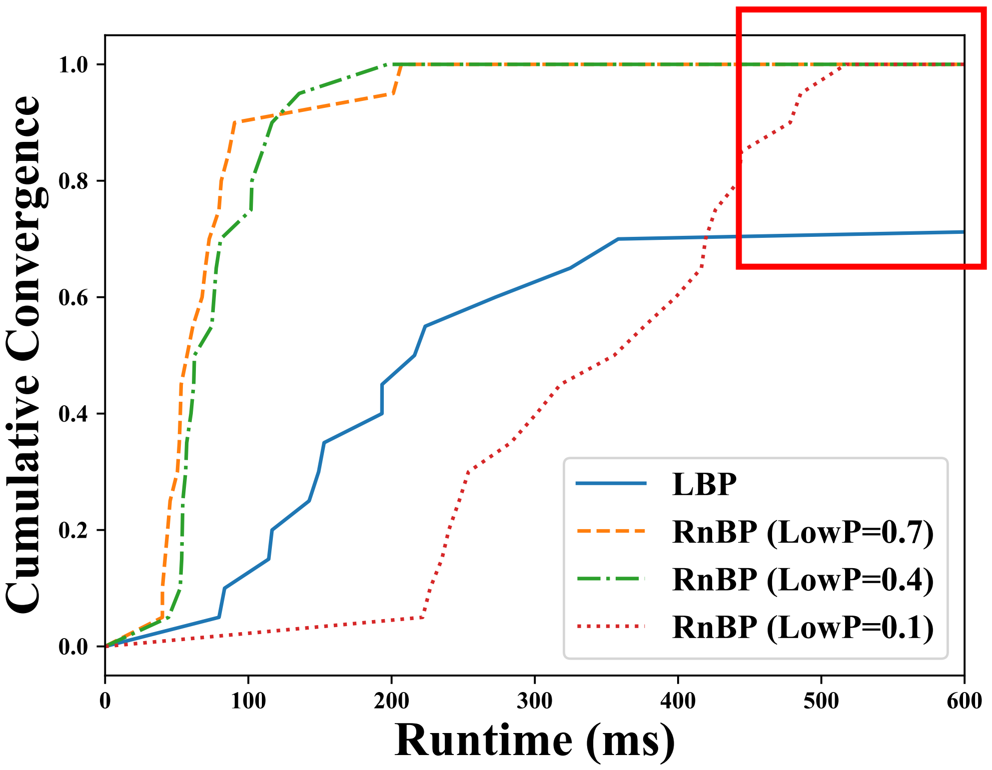
$$p = \begin{cases} p_{low} & EdgeRatio > 0.9 \\ p_{high} & o.w. \end{cases}$$

Like for RBP/RS adjusting *p* adjusts parallelism. We lock high to 1.0.

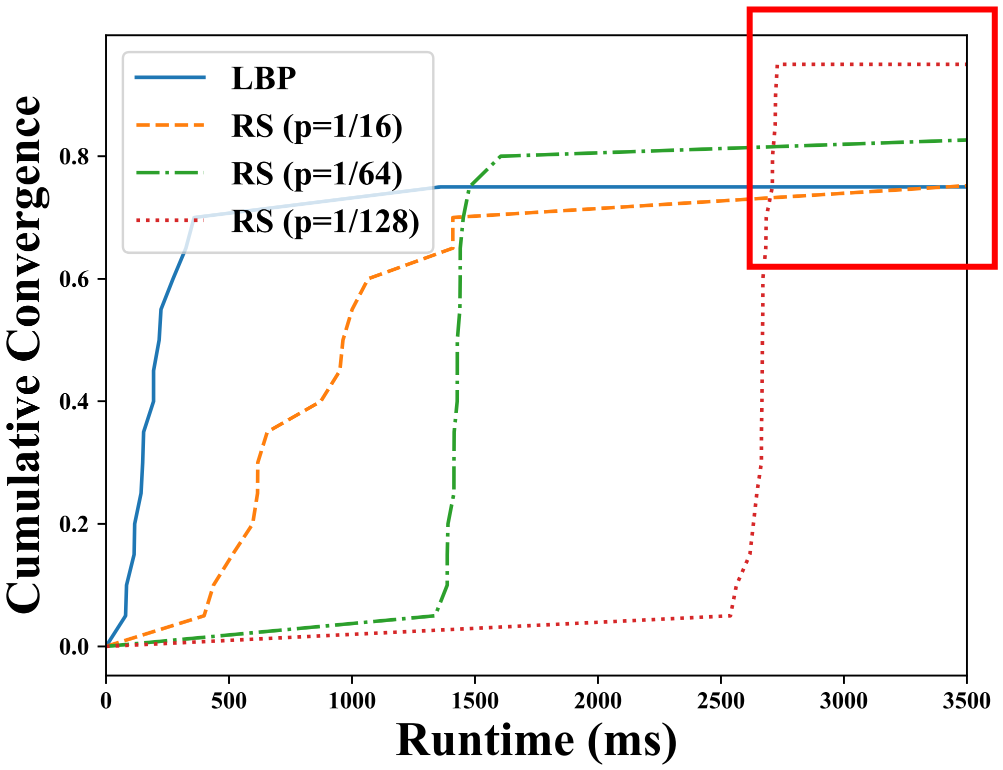
Ising 100x100, C=2.5

RnBP converges with higher parallelism.

GPU RnBP



GPU RS

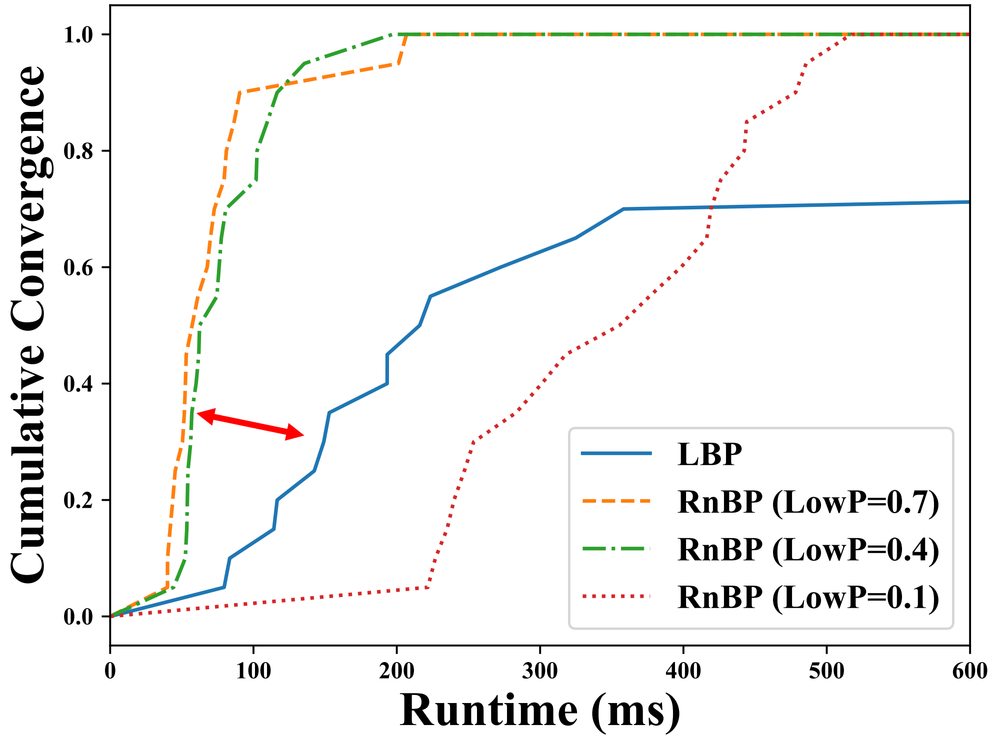


THE UNIVERSITY OF UTAH®

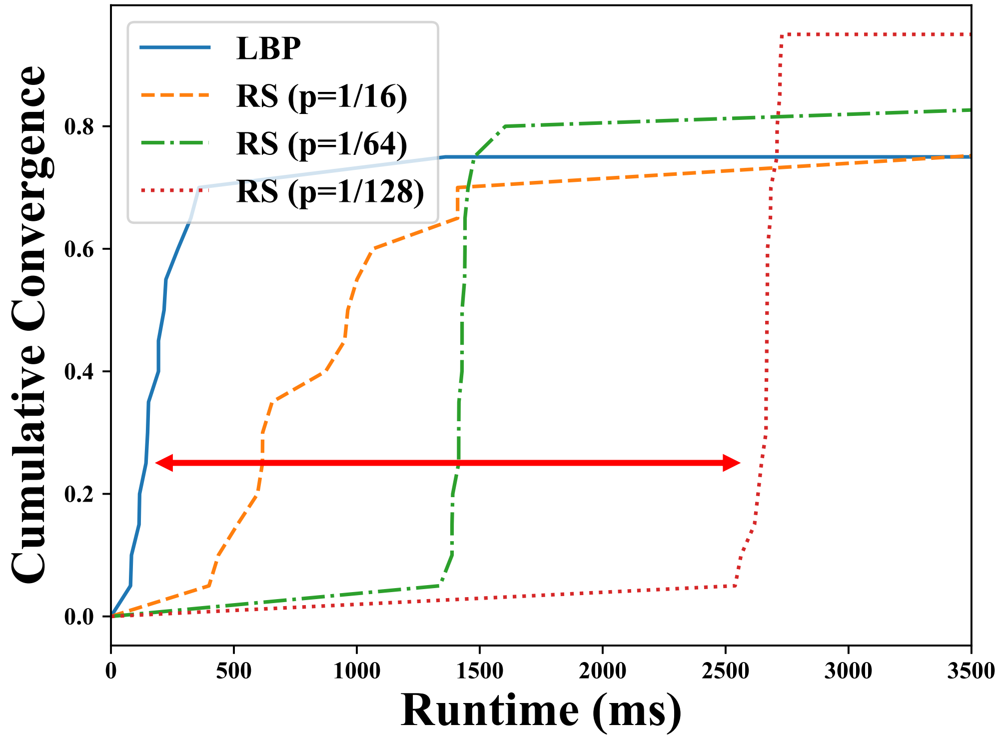
Ising 100x100, C=2.5

Does so while running *faster* than LBP.

GPU RnBP

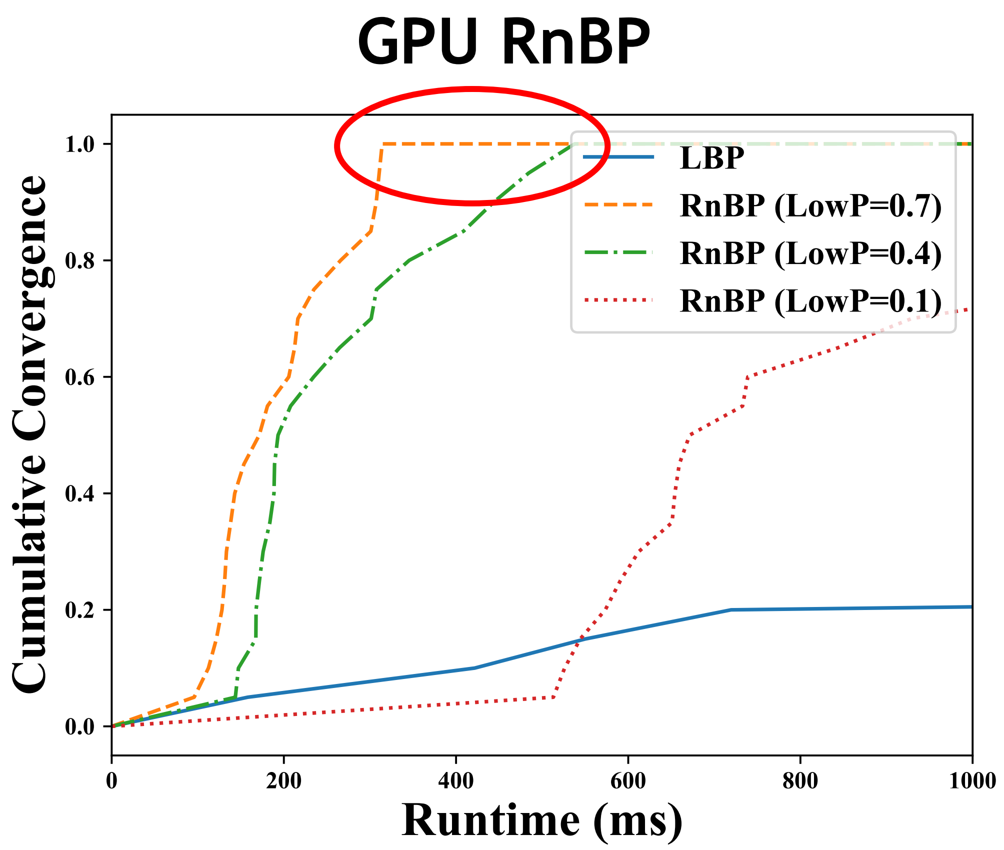


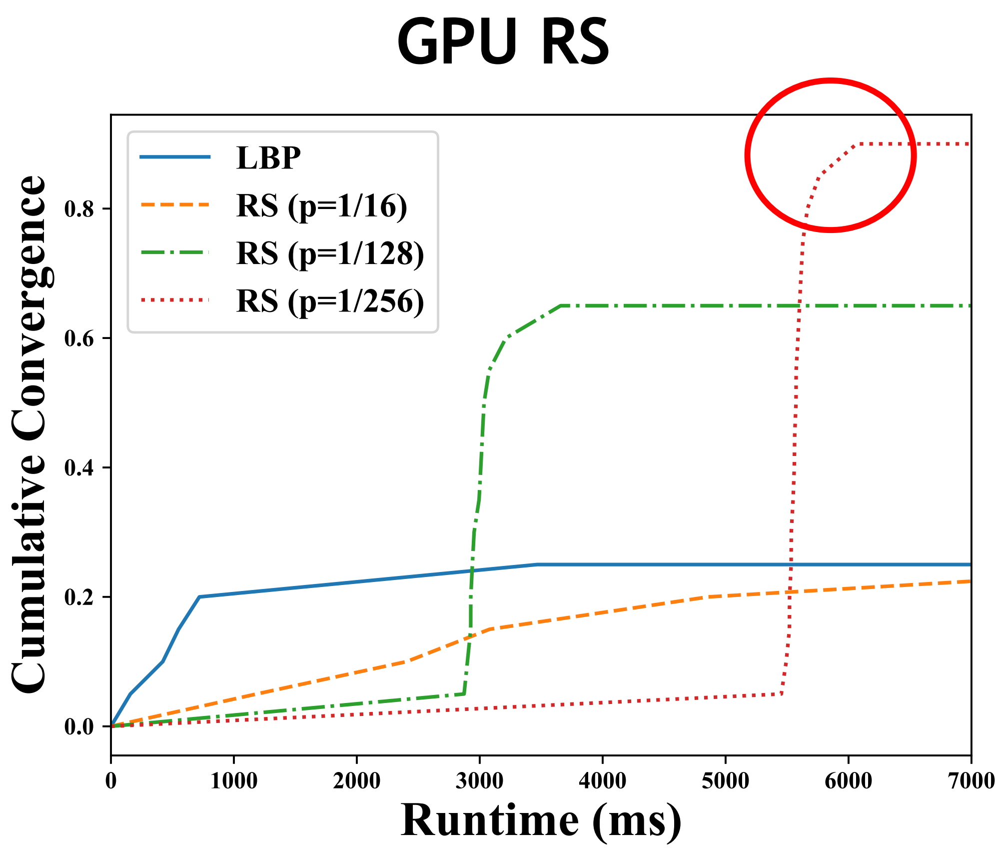
GPU RS



THE UNIVERSITY OF UTAH®

Ising 200x200, C=2.5

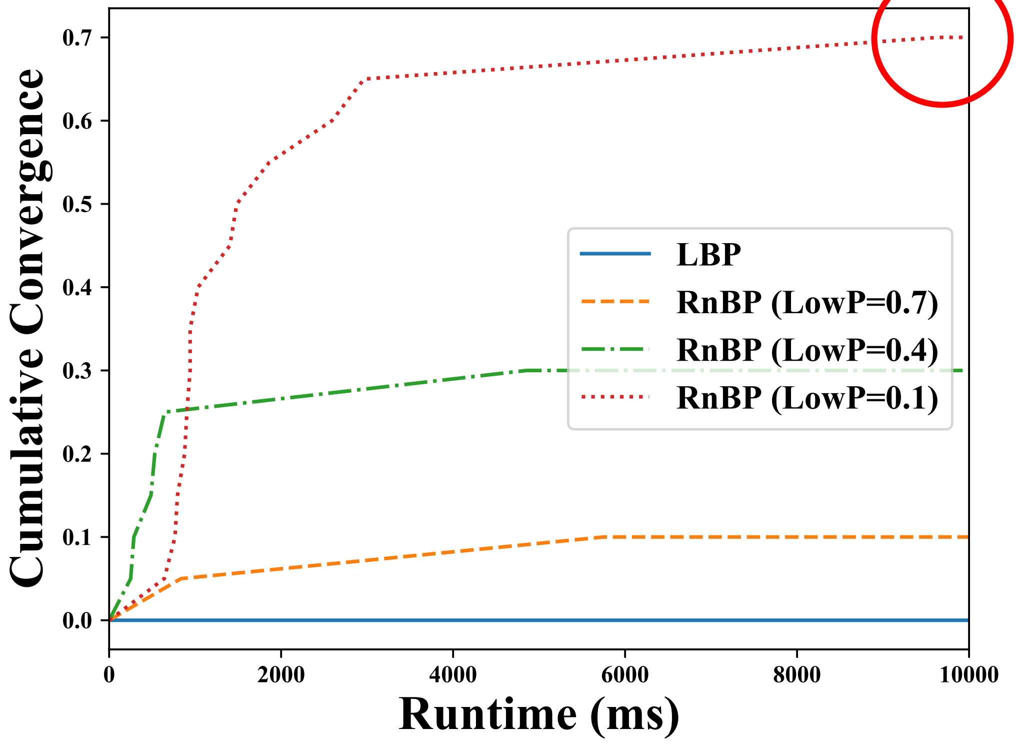




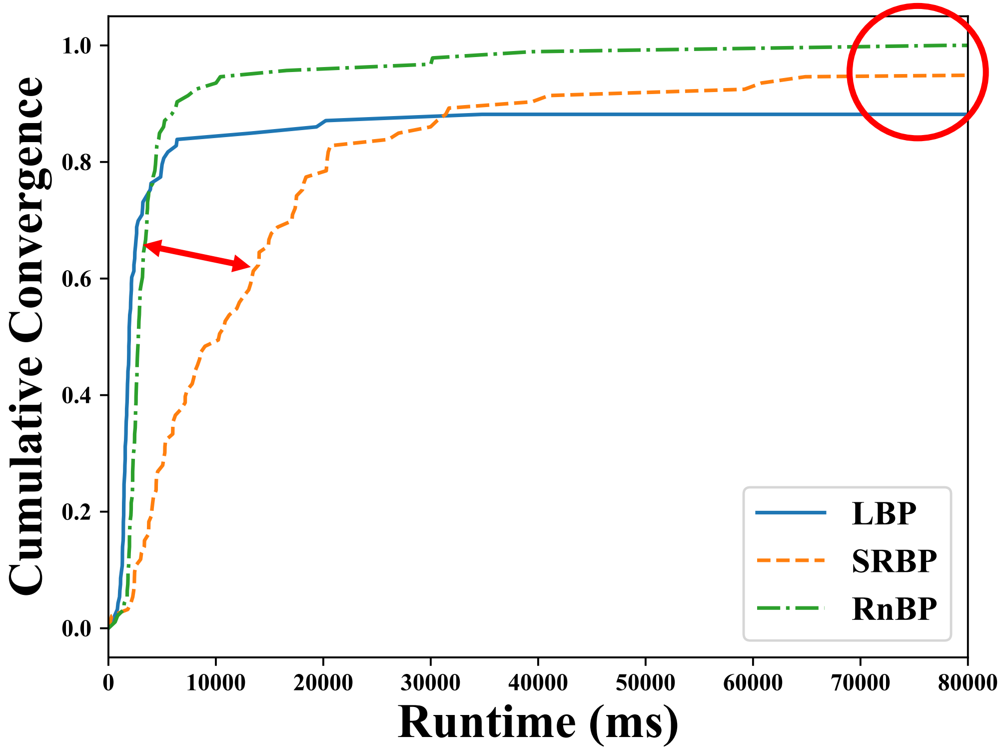
THE UNIVERSITY OF UTAH®

RnBP Additional Results

Ising 100x100, C=3



Protein Folding Dataset [1]



THE UNIVERSITY OF UTAH®

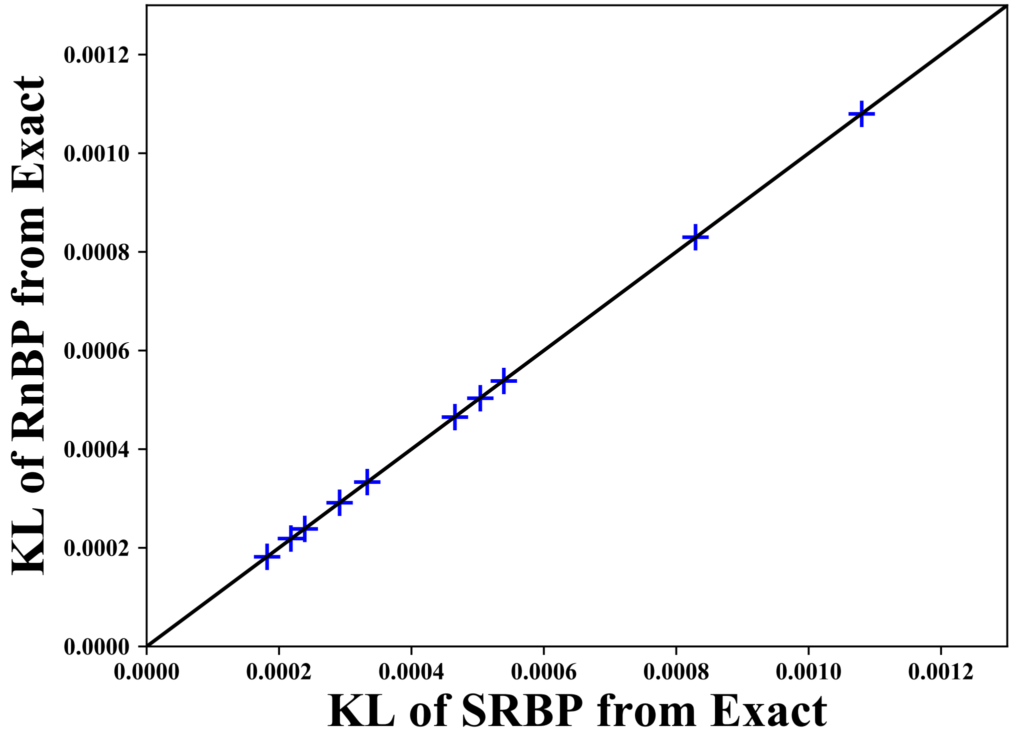
[1] *Approximate Inference and Protein-Folding, Chen Yanover, Yair Weiss*

RnBP Additional Results

RnBP Speedups

Dataset	Settings	SRBP Speedup
Ising 100×100, $C = 2$	$LowP = 0.7$	2203.58x
Ising 100×100, $C = 2.5$	$LowP = 0.7$	1135.05x
Ising 100×100, $C = 3$	$LowP = 0.1$	61.28x
Ising 200×200, $C = 2.5$	$LowP = 0.7$	> 529.997x
Chain 100000, $C = 10$	$LowP = 0.7$	> 1676.92x

Accuracy Comparison to SRBP/Variable Elimination



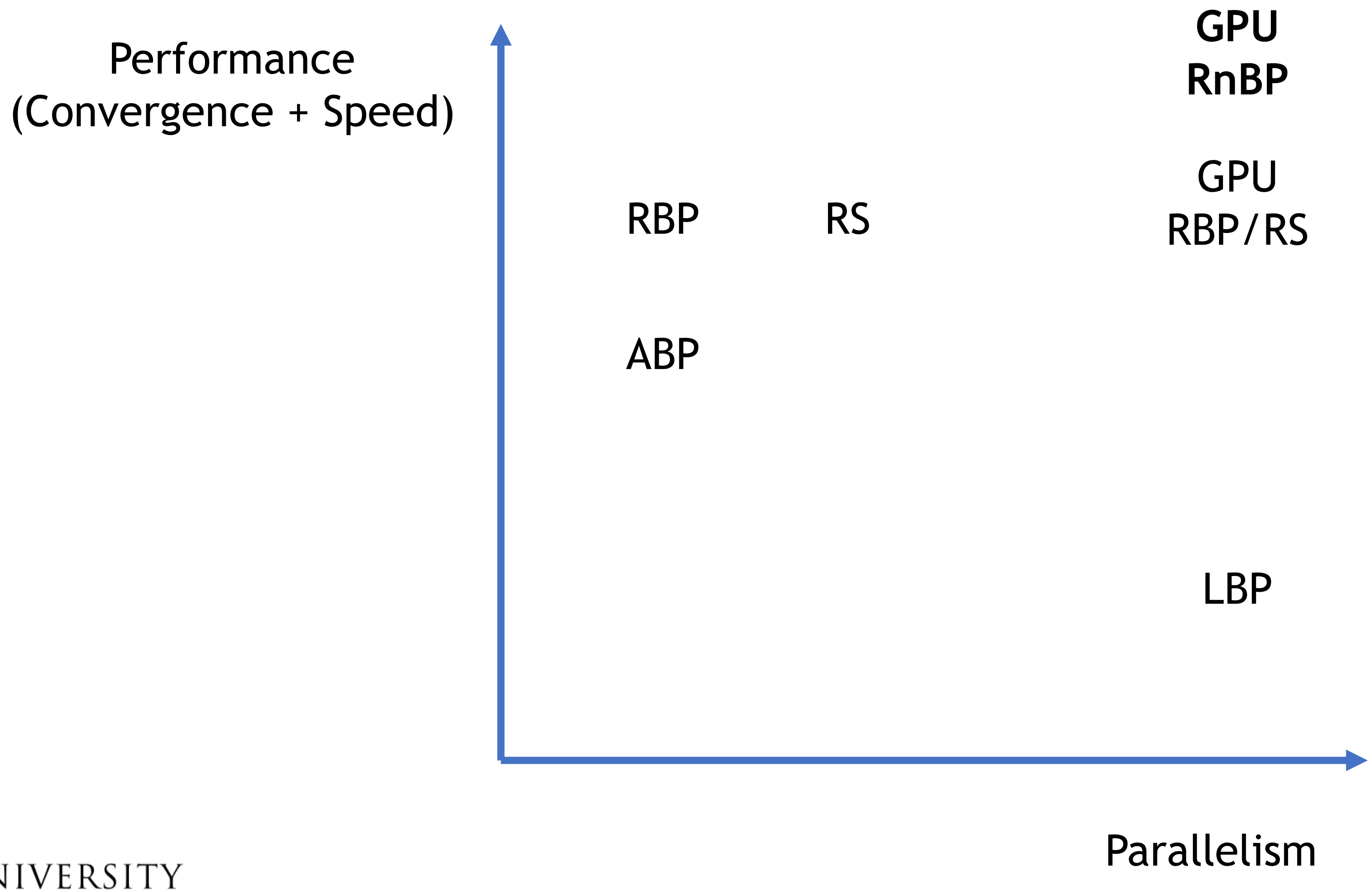
THE UNIVERSITY OF UTAH

Where are we now?

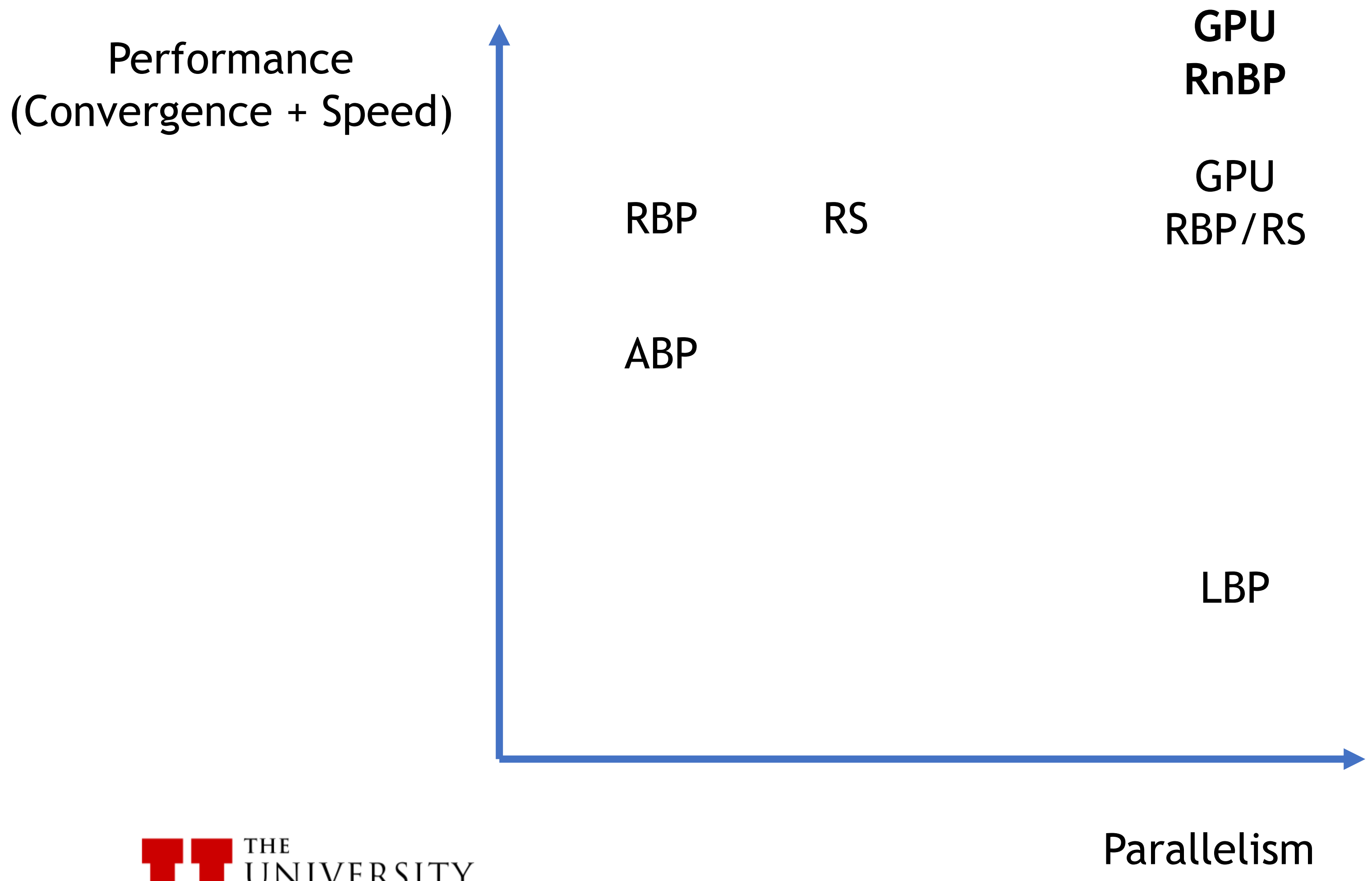
1. Accurate – are the results close to the true marginals? ✓
2. Convergent – does the algorithm complete? ✓
3. Fast – how fast does the algorithm complete? ✓

THE UNIVERSITY OF UTAH®

Where are we now?



Conclusion



- GPU LBP/RBP/RS.
- Tradeoff parallelism/ sequentialism.
- Shortcomings in existing approaches.
- Presented GPU RnBP!

THE UNIVERSITY OF UTAH®

Thank You!

or http://tiny.cc/bp-gpu

Slides + Code

THE UNIVERSITY OF UTAH®