Overcoming LDPC Trapping Sets with Informed Scheduling

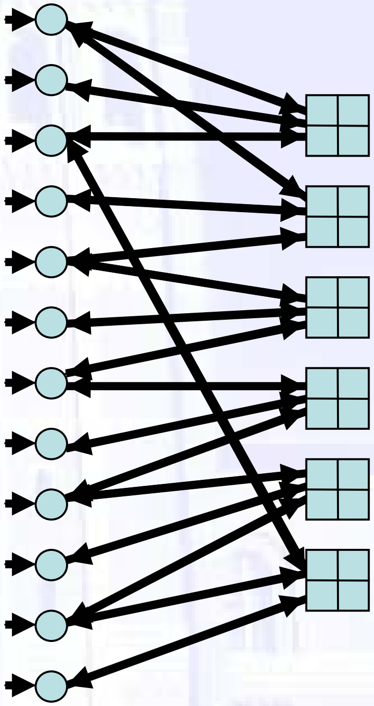
Andres I. Vila Casado
Miguel Griot
Richard Wesel

Outline

- Message-passing scheduling
  - Simultaneous vs sequential loopy BP
  - Sequential scheduling problem
- Informed scheduling strategies:
  - Residual Belief Propagation (RBP)
  - Node-wise RBP
- Reduced complexity strategies
- Informed scheduling and trapping sets
- Conclusions and open problems

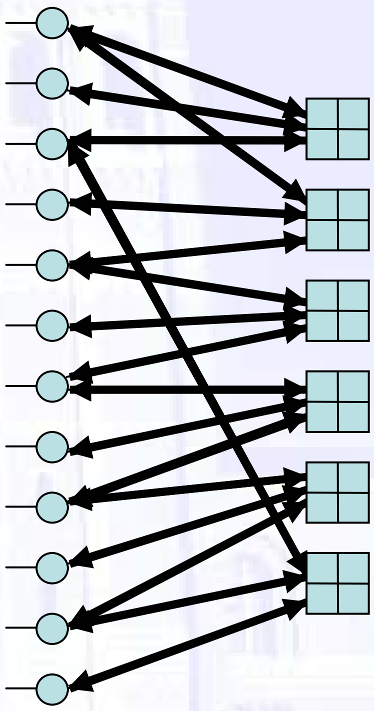
Simultaneous (Flooding) Schedule

- On every iteration
  - All variable nodes are simultaneously updated
  - All check nodes are simultaneously updated



Standard Sequential Schedule (SSS)

- Update messages sequentially:
  - [Mansour 03] (CN sequence)
  - [Kfir 03] (VN sequence)
  - [Hocevar 04] (LBP)
  - [Sharon 04] (Serial Schedule)
  - [Zhang 05] (Shuffled BP)
  - [Radosavljevic 05]
- Has been shown to converge at twice the speed than simultaneous scheduling
- What is the best sequence?



Residual Belief Propagation (RBP)

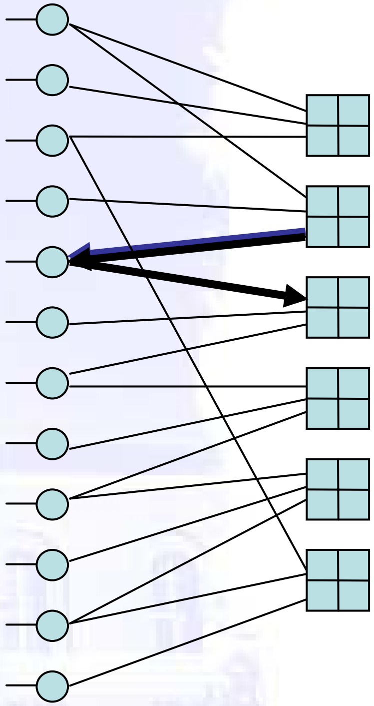
- Algorithm proposed by Elidan et al. for general loopy BP solutions
- Define residual as:

$$r = \left\| m_{new} - m_{old} \right\|$$

- As BP converges, the residuals go to 0
- RBP is a greedy algorithm that propagates the message with the largest residual

RBP for LDPC decoding

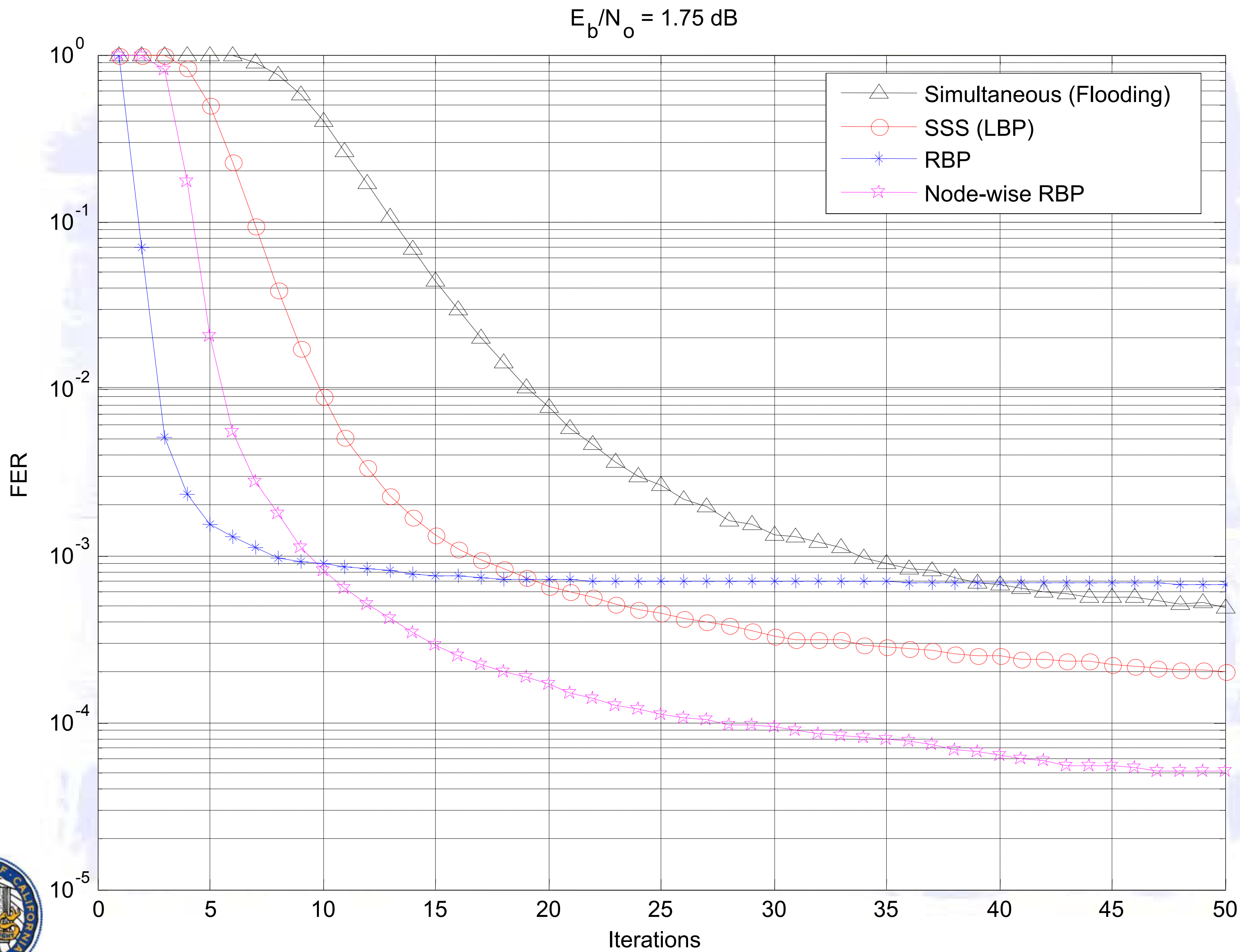
- Propagate message with biggest residual
- The variable-to-check messages that change will have the same residual so they are the biggest
- Therefore, RBP can be simplified
  - Propagate check-to-variable message with biggest residual
  - Propagate variable-to-check messages that change



Node-wise RBP

- RBP greediness makes it converge faster but less often
- Node-wise RBP simultaneously propagates all the outgoing messages from a check node
- The check-node updated is the one that has the message with the biggest residual
- This means that correcting messages are simultaneously send to all variable nodes that could be in error
- Node-wise RBP converges less fast than RBP but more often

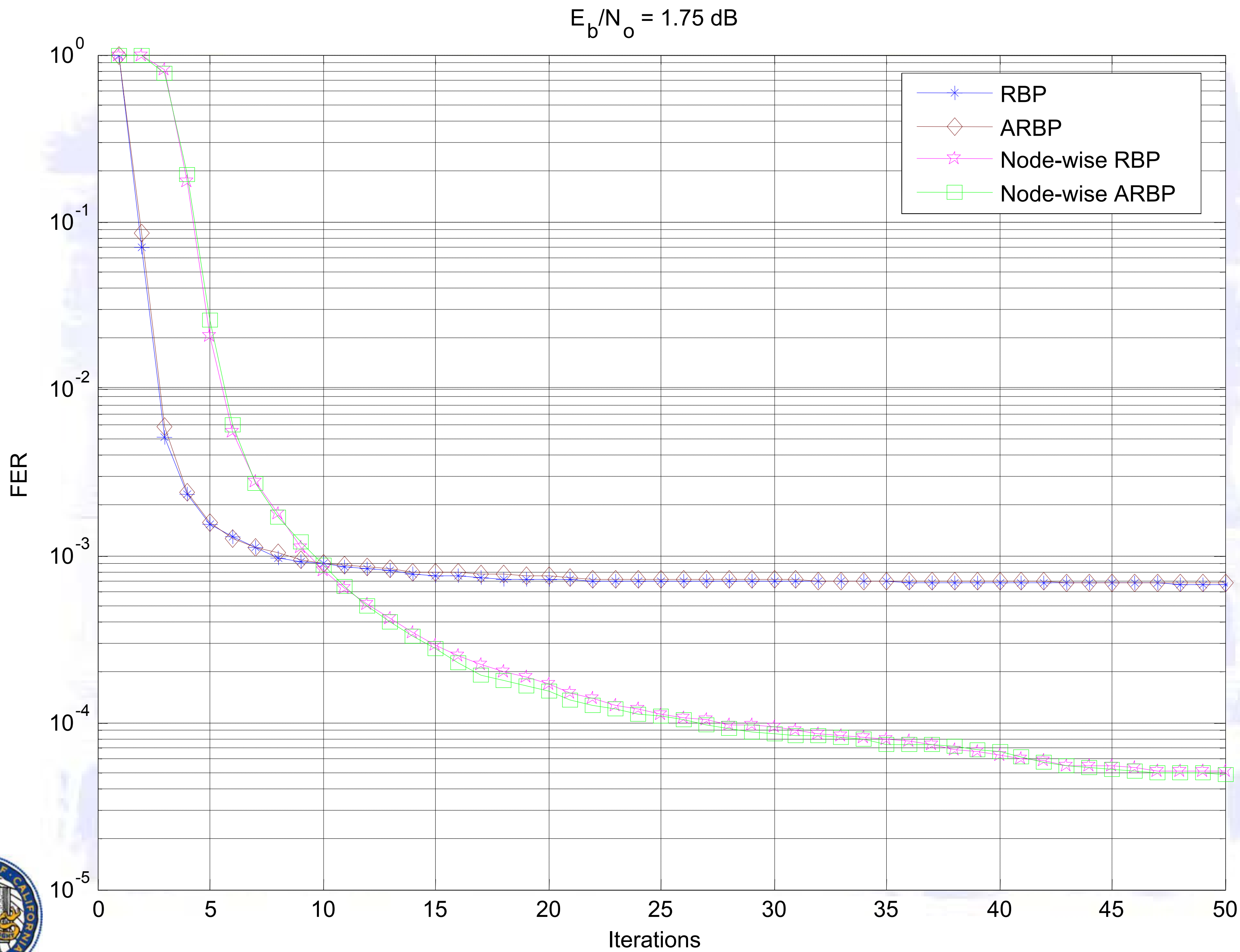
Results


E_b/N_o = 1.75 dB
Simultaneous (Flooding)
SSS (LBP)
RBP
Node-wise RBP
FER
Iterations
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
5
10
15
20
25
30
35
40
45
50

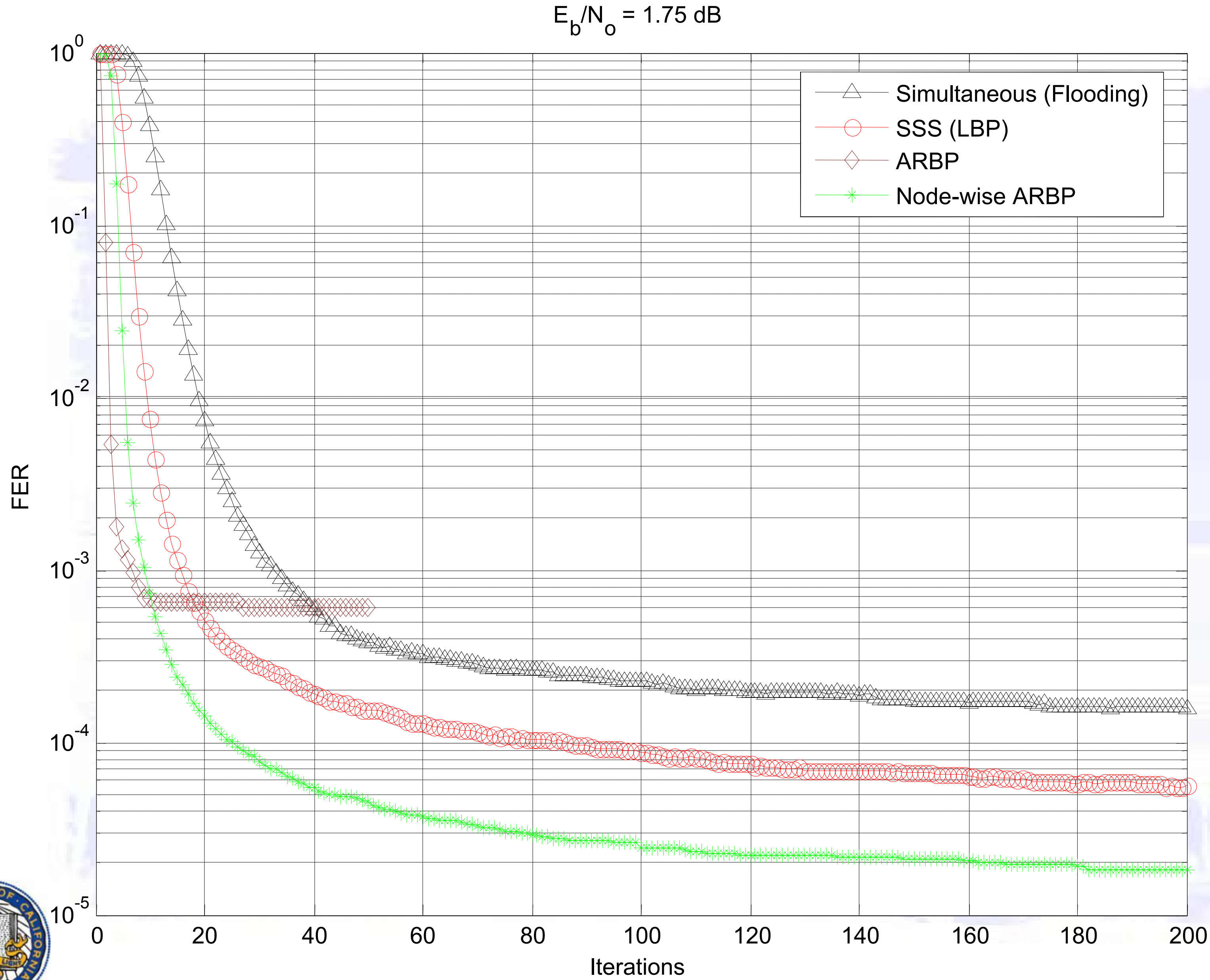
Complexity

- Residual computation requires the values of the messages to be propagated
- Many of those computations are then "wasted" since many of those messages aren't propagated
- Using the min-BP check-node update to compute residuals significantly reduces the complexity
- Approximate RBP (ARBP) and node-wise ARBP are the min-BP versions of RBP and node-wise RBP respectively

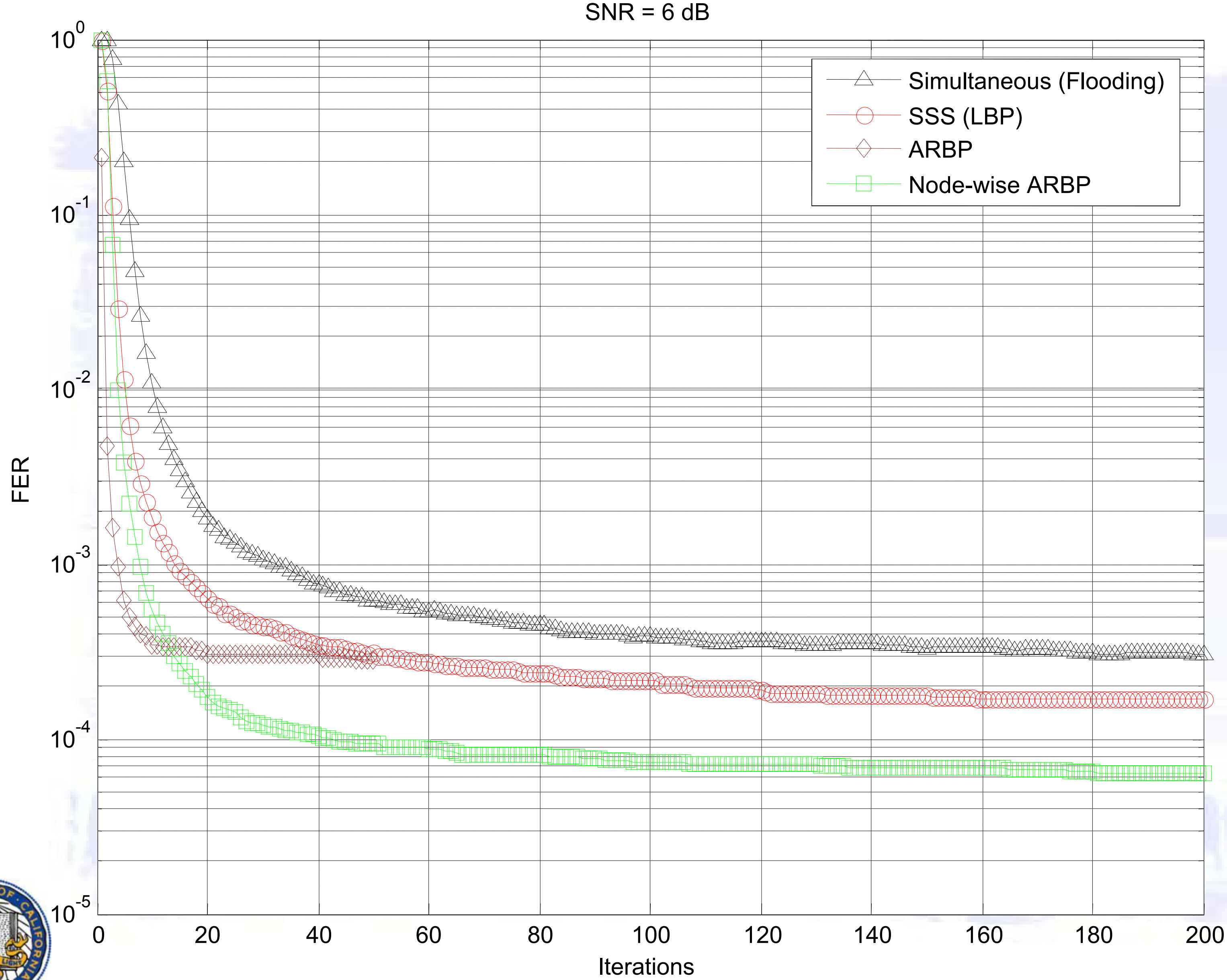
Results


E_b/N_o = 1.75 dB
FER
Iterations
RBP
ARBP
Node-wise RBP
Node-wise ARBP
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
5
10
15
20
25
30
35
40
45
50

Results 802.11n code rate 1/2



Results 802.11n code rate 5/6



Why does informed scheduling work faster?

- As mentioned before RBP focuses on the part of the graph that has not converged thus there are less "wasted" updates
- Furthermore, our informed scheduling strategies tend to chose to update nodes that have recently received new information

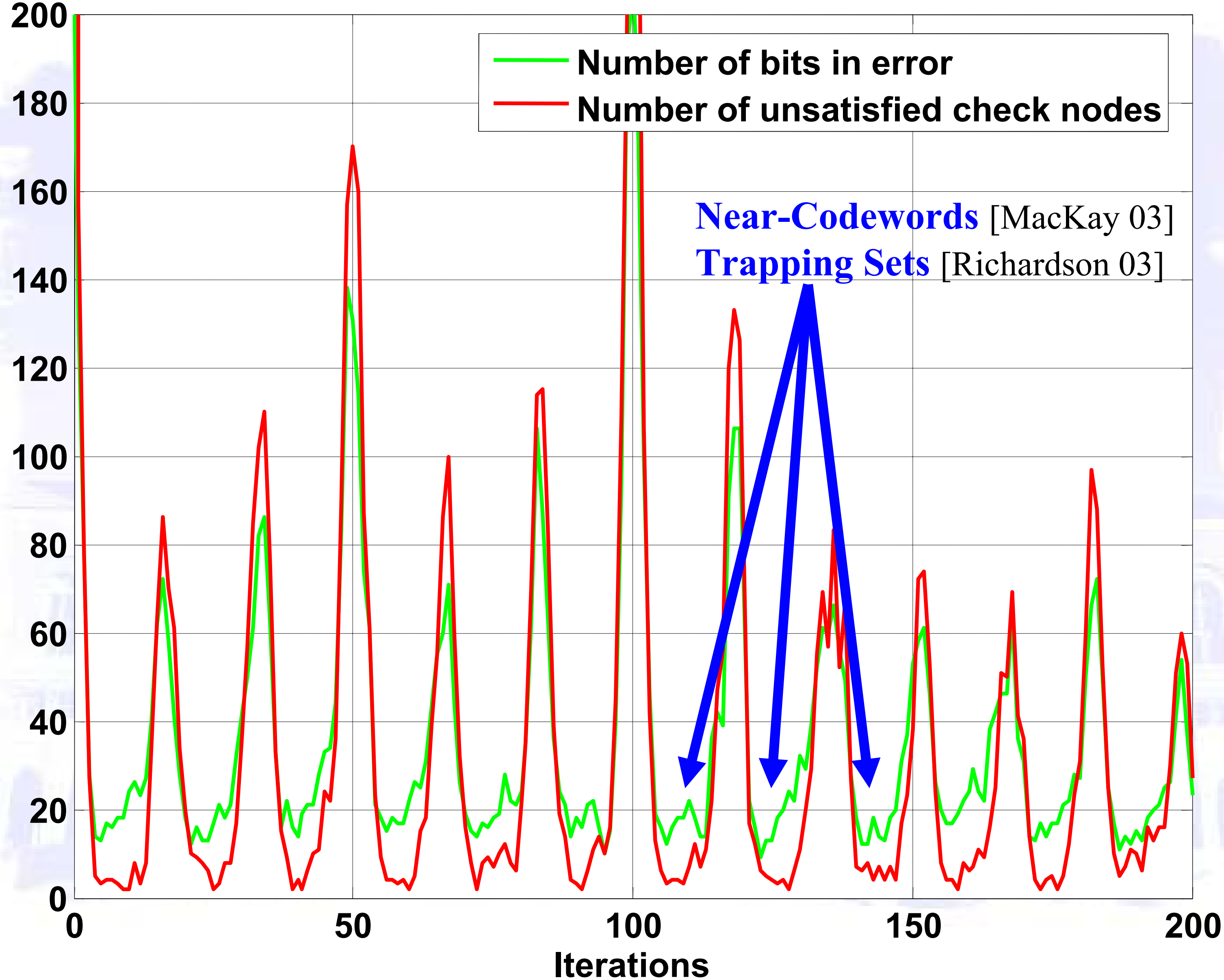
We understand faster, but why better?

- Performance plots show that our informed scheduling strategies perform not only faster but better than SSS
- There are many noise realizations that aren't corrected after 200 SSS iterations but are corrected after few NW-ARBP iterations
- This difference can't be explained with the argument of the "wasted" updates

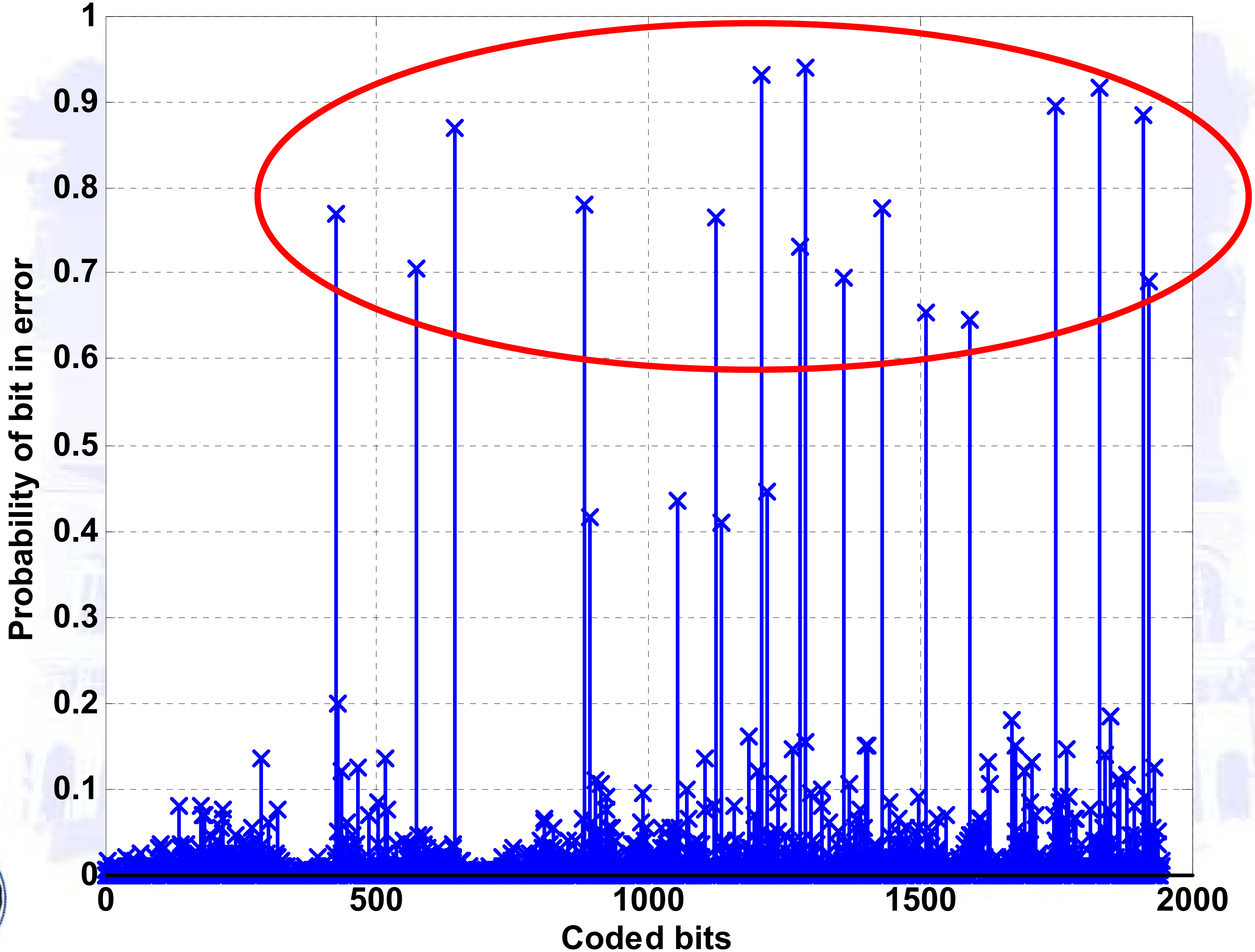
SSS Errors

- Let us characterize the following type of noise realizations
  - AWGN Channel
  - SSS (LBP) decoding doesn't converge after 200 iterations
  - NW-ARBP iterations converges after few iterations
  - Rate ½ 802.11n code
  - n=1944, k=972
  - SNR = 1.75 dB, waterfall region
- Most of these errors look similar let us see an example

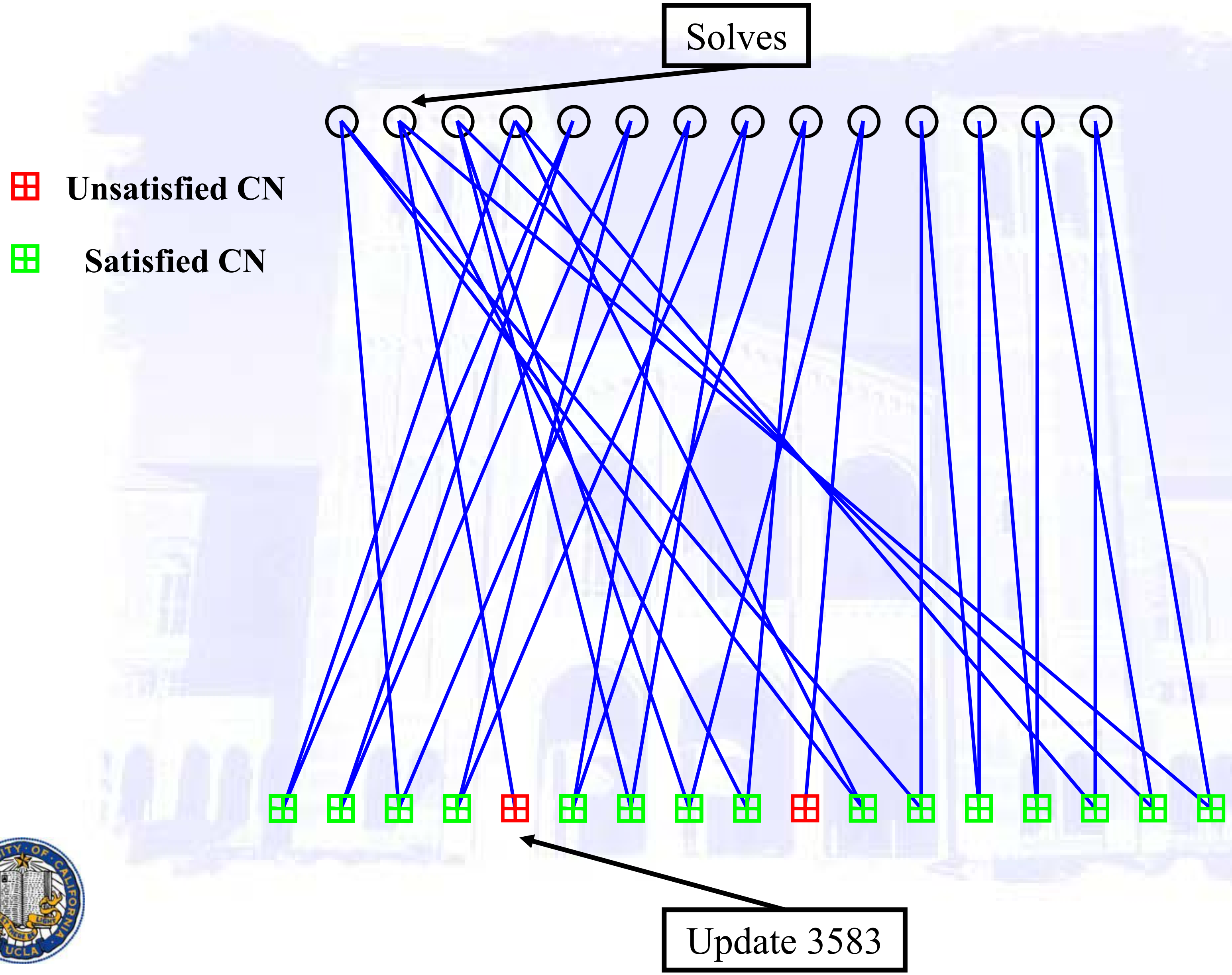
SSS (LBP) decoding behavior



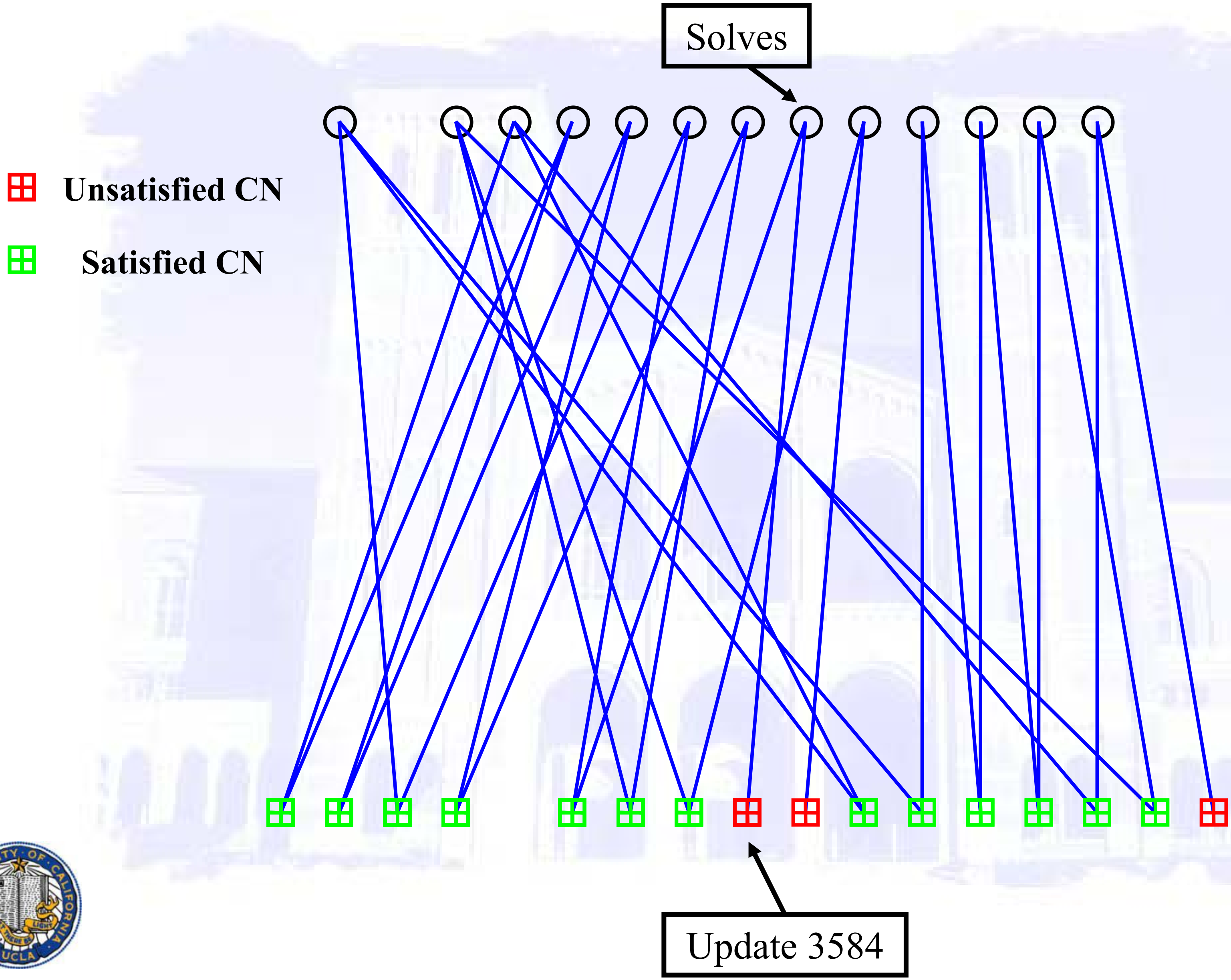
SSS (LBP) decoding behavior



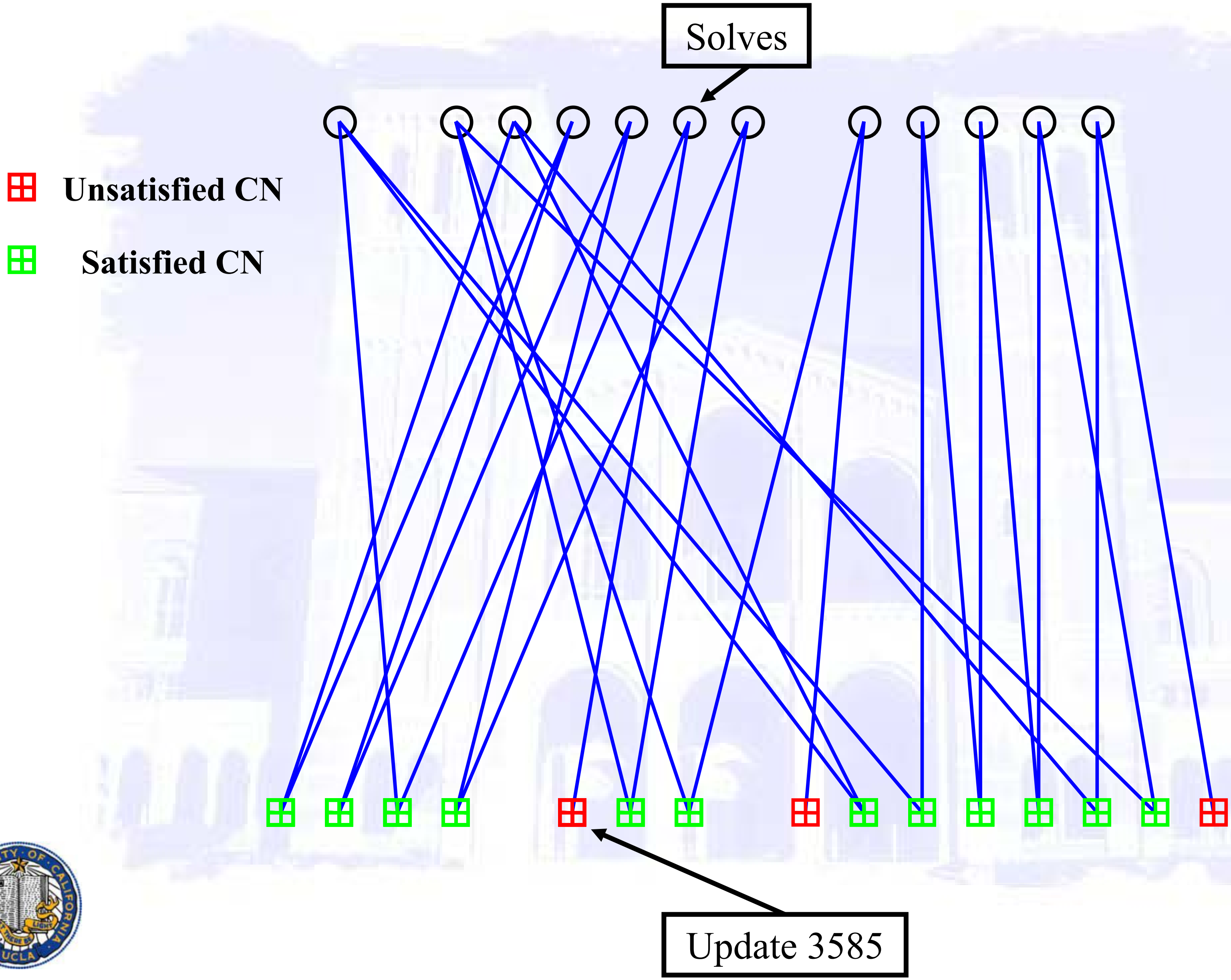
Informed Scheduling and Trapping Sets



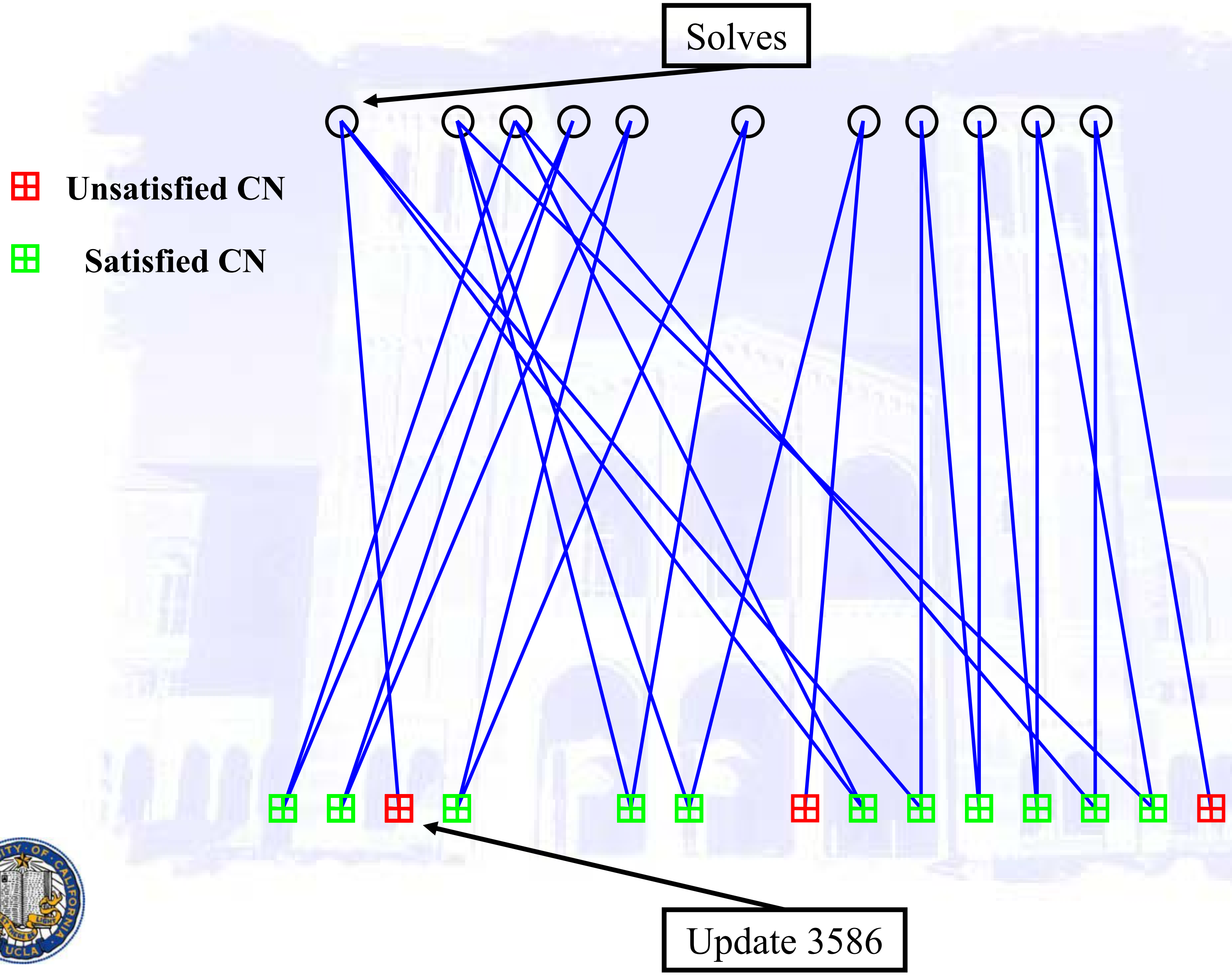
Informed Scheduling and Trapping Sets



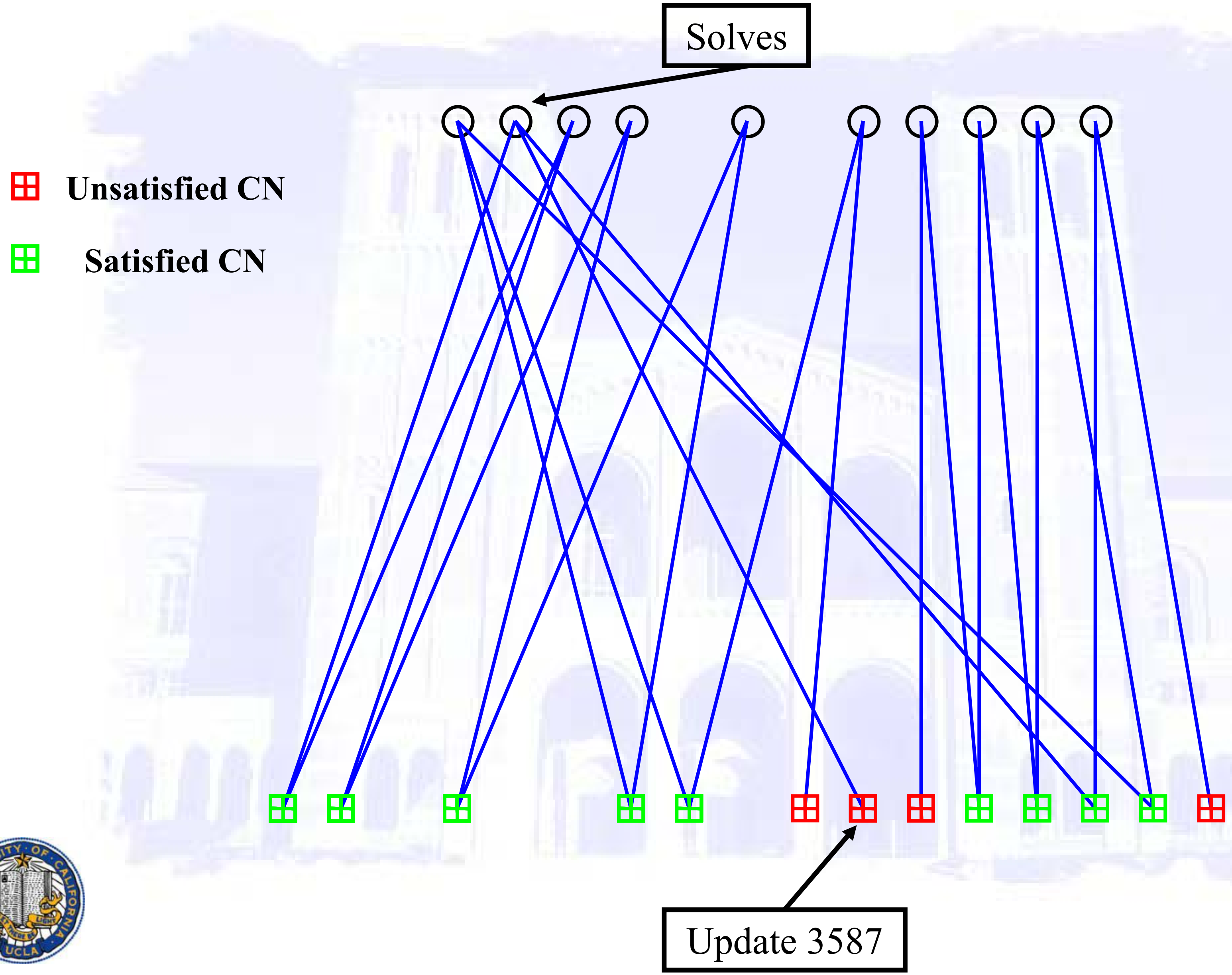
Informed Scheduling and Trapping Sets



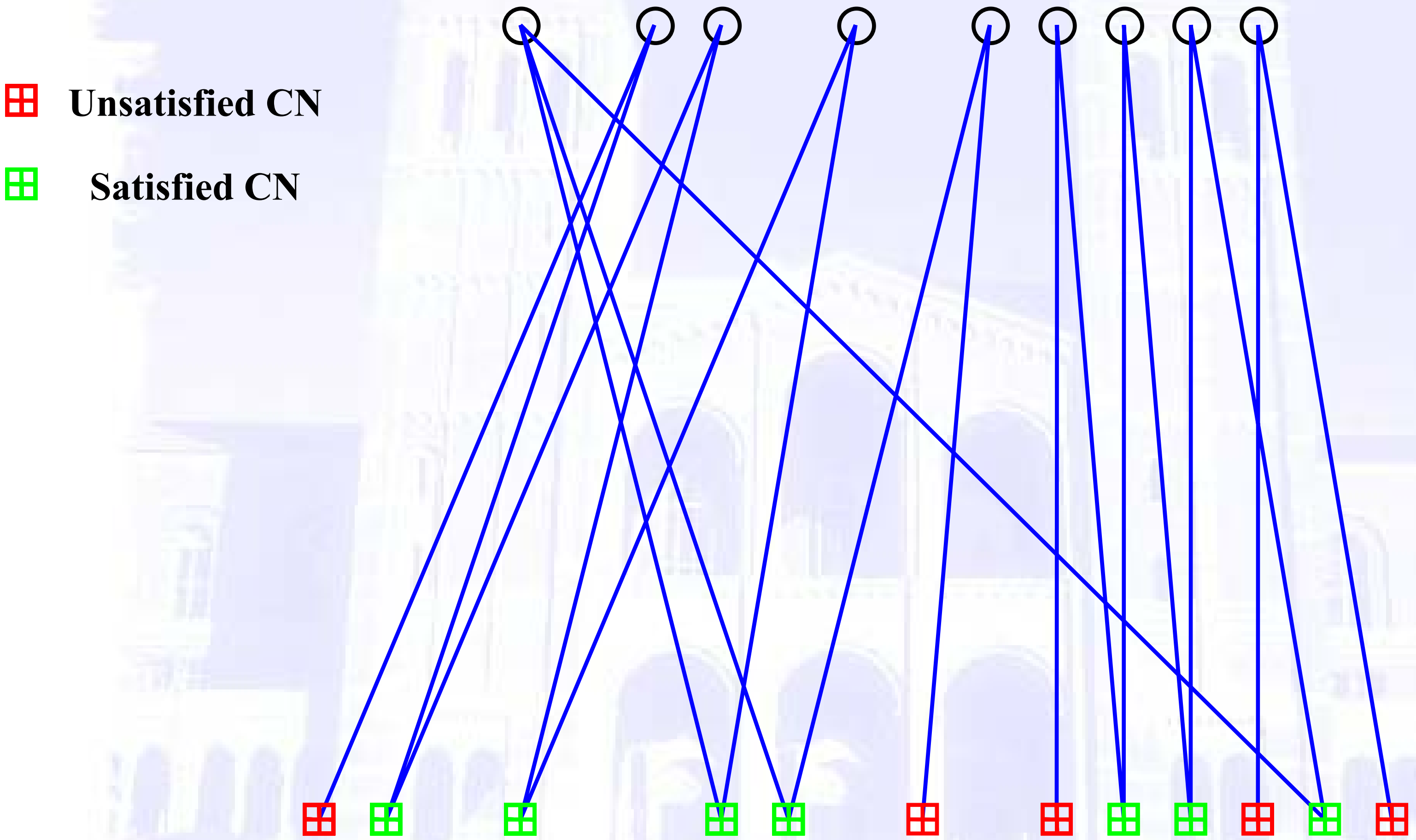
Informed Scheduling and Trapping Sets



Informed Scheduling and Trapping Sets



Informed Scheduling and Trapping Sets



Conclusions

- Scheduling significantly affects the performance of loopy BP LDPC decoding
- RBP and ARBP are good strategies for a small number of iterations or high target error-rates
- Node-wise RBP and node-wise ARBP perform better than SSS for all iterations
- Furthermore, NW-ARBP helps solve trapping set errors

Open problems

- Informed scheduling on the error-floor region
- Joint SSS (LBP) and informed scheduling
- Extensions to other applications that use loopy BP:
  - Turbo Codes
  - Turbo equalization
  - Joint demodulation and decoding
- Extensions to applications outside the communications area