Message Passing Algorithms:
Communication, Inference and Optimization

Jinwoo Shin

KAIST EE
Message Passing Algorithms in My Research

- Communication Networks (e.g. Internet)
Message Passing Algorithms in My Research

- Communication Networks (e.g. Internet)
- Statistical Networks (e.g. Bayesian networks)
- Social Networks (e.g. Facebook)
Message Passing Algorithms in My Research

• Communication Networks (e.g. Internet)
  - Scheduling (e.g. medium access, packet switching, optical-core networks)

• Statistical Networks (e.g. Bayesian networks)

• Social Networks (e.g. Facebook)
Message Passing Algorithms in My Research

- Communication Networks (e.g. Internet)
  - Scheduling (e.g. medium access, packet switching, optical-core networks)
  - Distributed optimization and consensus (e.g. gossip algorithms)

- Statistical Networks (e.g. Bayesian networks)

- Social Networks (e.g. Facebook)
Message Passing Algorithms in My Research

• Communication Networks (e.g. Internet)
  - Scheduling (e.g. medium access, packet switching, optical-core networks)
  - Distributed optimization and consensus (e.g. gossip algorithms)

• Statistical Networks (e.g. Bayesian networks)
  - Inference (e.g. Markov chain Monte Carlo, belief propagation)

• Social Networks (e.g. Facebook)
Message Passing Algorithms in My Research

- **Communication Networks (e.g. Internet)**
  - Scheduling (e.g. medium access, packet switching, optical-core networks)
  - Distributed optimization and consensus (e.g. gossip algorithms)

- **Statistical Networks (e.g. Bayesian networks)**
  - Inference (e.g. Markov chain monte carlo, belief propagation)

- **Social Networks (e.g. Facebook)**
  - Game theoretical modeling and analysis (e.g. best reply, logit-response)

- N. Abramson (1970s)
- J. Tsitsiklis (1980s)
- N. Metropolis (1950s)
- J. Nash (1950s)
Message Passing Algorithms in My Research

- **Communication Networks (e.g. Internet)**
  - Scheduling (e.g. medium access, packet switching, optical-core networks)
  - Distributed optimization and consensus (e.g. gossip algorithms)

- **Statistical Networks (e.g. Bayesian networks)**
  - Inference (e.g. Markov chain monte carlo, belief propagation)

- **Social Networks (e.g. Facebook)**
  - Game theoretical modeling and analysis (e.g. best reply, logit-response)
Message Passing Algorithms in My Research

- Communication Networks (e.g. Internet)
  - Scheduling (e.g. medium access, packet switching, optical-core networks)
  - Distributed optimization and consensus (e.g. gossip algorithms)

- Statistical Networks (e.g. Bayesian networks)
  - Inference (e.g. Markov chain monte carlo, belief propagation)

- Social Networks (e.g. Facebook)
  - Game theoretical modeling and analysis (e.g. best reply, logit-response)
- Part I -
Message Passing in Communication Networks

- Focus on medium access for wireless networks
  Joint work with Devavrat Shah (MIT)

- Later describe how the result is extended
  Joint work with Yung Yi (KAIST), Seyoung Yun, (MSR), Tonghoon Suk (Gatech)
Motivation: Wireless Network

- A and C cannot send packets to B simultaneously
  - ‘A->B’ and ‘C->B’ interfere with each other & packets collide
Motivation: Wireless Network

- However, ‘A->B’ and ‘D->C’ do not interfere with each other
Motivation: Wireless Network

- Question: How to avoid interference?
  - While transmitting as many packets as possible
• **Motivation:** Wireless Network

- **Question:** How to avoid interference?
  - While transmitting as many packets as possible

- **Need a contention resolution protocol**
  - Also called *medium access algorithm*
  - e.g. CSMA/CA, ALOHA, TDMA, CDMA, etc.
Motivation: Wireless Network

- Question: How to avoid interference?
  - While transmitting as many packets as possible

- Need a contention resolution protocol
  - Also called medium access algorithm
  - e.g. CSMA/CA, ALOHA, TDMA, CDMA, etc.

However, no protocol is known to be optimal in a certain sense.
Our Goal

- Design a `provably` optimal medium access algorithm
Our Goal

- Design a `provably' optimal medium access algorithm

Next: Describe a mathematical model
Our Goal

- Design a `provably' optimal medium access algorithm

Next: Describe a mathematical model
Next: Definition of `optimality'
Our Goal

- Design a `provably' optimal medium access algorithm

Next: Describe a mathematical model

Next: Definition of `optimality'

- For simplicity, I will consider a simple model (i.e. discrete-time, single-hop, single-channel)

- However, the same story goes through for other models (i.e. continuous-time, multi-hop, multi-channel, collisions, time-varying)
Model: Wireless Network

- Wireless network = Collection of queues
  - Each queue represents a communication link (e.g. A→B)
Wireless network = Collection of queues
- Each queue represents a communication link (e.g. A→B)
- Interference graph G

“Queues form vertices & two queues share an edge if they cannot transmit simultaneously”
Model: Wireless Network

- We assume
  - Packets arrive at queue i with rate $\lambda(i)$ e.g. Bernoulli stochastic process
  - Time is discrete & at most one packet can depart from each queue at each time instance
We assume - Packets arrive at queue i with rate $\lambda(i)$ e.g. Bernoulli stochastic process

- Time is discrete & at most one packet can depart from each queue at each time instance

At each time instance, each queue attempts to transmit or keeps silent

- The decision is made by a medium access algorithm
- A packet departs if queue attempts and no neighbor attempts to transmit simultaneously
We assume:

- Packets arrive at queue i with rate $\lambda(i)$ e.g. Bernoulli stochastic process
- Time is discrete & at most one packet can depart from each queue at each time instance

At each time instance, each queue attempts to transmit or keeps silent:

- The decision is made by a medium access algorithm
- A packet departs if queue attempts and no neighbor attempts to transmit simultaneously

Next: Example of simple medium access algorithm
Example of Medium Access Algorithm

- Each individual queue attempts to transmit at time $t$
  - If success, attempt to transmit at time $t+1$
  - Else, keep silent for a ‘random’ time interval
Example of Medium Access Algorithm

- Each individual queue attempts to transmit at time $t$
  - If success, attempt to transmit at time $t+1$
  - Else, keep silent for a ‘random’ time interval

How to decide the length of the time interval?
Example of Medium Access Algorithm

- Each individual queue attempts to transmit at time $t$
  - If success, attempt to transmit at time $t+1$
  - Else, keep silent for a ‘random’ time interval

- Back-off protocol
  - The (expected) length of time interval is a function of consecutive failures

How to decide the length of the time interval?
Example of Medium Access Algorithm

- Each individual queue attempts to transmit at time $t$
  - If success, attempt to transmit at time $t+1$
  - Else, keep silent for a ‘random’ time interval

- Back-off protocol
  - The (expected) length of time interval is a function of consecutive failures
  - [Hastad, Leighton, Rogoff 1988] The back-off protocol is throughput-optimal if
    “the function is a polynomial and the interference graph is complete”
  - Throughput-optimal = Keep queues finite under the largest possible arrival rates $\lambda(i)$

How to decide the length of the time interval?
Example of Medium Access Algorithm

- Each individual queue attempts to transmit at time $t$
  - If success, attempt to transmit at time $t+1$
  - Else, keep silent for a ‘random’ time interval

- Back-off protocol
  - The (expected) length of time interval is a function of consecutive failures
  - [Hastad, Leighton, Rogoff 1988] The back-off protocol is throughput-optimal if “the function is a polynomial and the interference graph is complete”
  - Throughput-optimal = Keep queues finite under the largest possible arrival rates $\lambda(i)$
Example of Medium Access Algorithm

- Each individual queue attempts to transmit at time \( t \)
  - If success, attempt to transmit at time \( t+1 \)
  - Else, keep silent for a ‘random’ time interval

- Back-off protocol
  - The (expected) length of time interval is a function of consecutive failures
  - [Hastad, Leighton, Rogoff 1988] The back-off protocol is throughput-optimal if
    “the function is a polynomial and the interference graph is complete”
  - Throughput-optimal = Keep queues finite under the largest possible arrival rates \([\lambda(i)]\)
Open Question for General Interference Graph

- Much Research starting from 1970s in various setups

  [Abramson and Kuo 73] [Metcalfe and Bogg 76] [Mosely and Humble 85] [Kelly and MacPhee 87] [Aldous 87] [Tsybakov and Likhanov 87] [Hastad, Leighton, Rogoff 96] [Tassiulas 98] [Goldberg and MacKenzie 99] [Goldberg, Jerrum, Kannan and Paterson 00] [Gupta and Stolyar 06] [Dimakis and Walrand 06] [Modiano, Shah and Zussman 06] [Marbach 07] [Eryilmaz, Marbach and Ozdaglar 07] [Leconte, Ni and Srikant 09] ...

- Till 1990s: complete interference graph (e.g. Ethernet)
- From 1990s: general interference graph (e.g. wireless networks)
Open Question for General Interference Graph

- Much Research starting from 1970s in various setups

  [Abramson and Kuo 73] [Metcalfe and Bogg 76] [Mosely and Humble 85] [Kelly and MacPhee 87] [Aldous 87] [Tsybakov and Likhanov 87] [Hastad, Leighton, Rogoff 96] [Tassiulas 98] [Goldberg and MacKenzie 99] [Goldberg, Jerrum, Kannan and Paterson 00] [Gupta and Stolyar 06] [Dimakis and Walrand 06] [Modiano, Shah and Zussman 06] [Marbach 07] [Eryilmaz, Marbach and Ozdaglar 07] [Leconte, Ni and Srikant 09] ...

- Till 1990s: complete interference graph (e.g. Ethernet)

- From 1990s: general interference graph (e.g. wireless networks)

- No simple distributed throughput-optimal protocol is known for general interference graph

- Next: Recent progress for this open question
Medium Access using Carrier Sensing

- Assume Carrier Sensing information
  - Knowledge whether neighbors attempted to transmit (at the previous time instance)
  - Medium access algorithm using this information is called CSMA
Medium Access using Carrier Sensing

• Assume Carrier Sensing information
  - Knowledge whether neighbors attempted to transmit (at the previous time instance)
  - Medium access algorithm using this information is called CSMA

• CSMA protocol: At each time t, each queue i
  - Check whether some (interfering) neighbors attempted to transmit at time t-1
  - If no, attempts to transmit with probability $p_i(t)$
  - Else, keep silent
Medium Access using Carrier Sensing

- Assume Carrier Sensing information
  - Knowledge whether neighbors attempted to transmit (at the previous time instance)
  - Medium access algorithm using this information is called CSMA

- CSMA protocol: At each time \( t \), each queue \( i \)
  - Check whether some (interfering) neighbors attempted to transmit at time \( t-1 \)
  - If no, attempts to transmit with probability \( p_i(t) \)
  - Else, keep silent

How to design this `access probability` \( p_i(t) \) for each queue \( i \)?
How to Choose Access Probability in CSMA:
Rate-based
How to Choose Access Probability in CSMA: Rate-based

- [Jiang and Walrand 2007] prove that there exists a `fixed, optimal' access probability $p_i(t) = p_i(G, \lambda)$ for throughput-optimality.
How to Choose Access Probability in CSMA: Rate-based

- [Jiang and Walrand 2007] prove that
  - There exists a `fixed, optimal' access probability $p_i(t) = p_i(G, \lambda)$ for throughput-optimality
  - Also propose updating rules of $p_i(t)$ at time $T, 2T, 3T, ...$
How to Choose Access Probability in CSMA: Rate-based

- [Jiang and Walrand 2007] prove that
  - There exists a `fixed, optimal' access probability $p_i(t)=p_i(G, \lambda)$ for throughput-optimality
  - Also propose updating rules of $p_i(t)$ at time $T, 2T, 3T, ...$
  - And `conjecture' that $p_i(t)$ converges to $p_i(G, \lambda)$, and hence throughput optimal
How to Choose Access Probability in CSMA: Rate-based

- [Jiang and Walrand 2007] prove that
  - There exists a 'fixed, optimal' access probability $p_i(t) = p_i(G, \lambda)$ for throughput-optimality
  - Also propose updating rules of $p_i(t)$ at time $T, 2T, 3T, ...$
    
    \[ p_i(2T) = p_i(T) \pm \varepsilon \]
    
    depending on whether queue $i$ increases in time $[T, 2T]$'

- And 'conjecture' that $p_i(t)$ converges to $p_i(G, \lambda)$, and hence throughput optimal
How to Choose Access Probability in CSMA: Rate-based

- [Jiang and Walrand 2007] prove that
  - There exists a `fixed, optimal’ access probability \( p_i(t) = p_i(G, \lambda) \) for throughput-optimality
  - Also propose updating rules of \( p_i(t) \) at time \( T, 2T, 3T, \ldots \)
    
    \[
    p_i(2T) = p_i(T) \pm \varepsilon \text{ depending on whether queue } i \text{ increases in time } [T, 2T]
    \]
  - And `conjecture’ that \( p_i(t) \) converges to \( p_i(G, \lambda) \), and hence throughput optimal
How to Choose Access Probability in CSMA: Rate-based

- [Jiang and Walrand 2007] prove that
  - There exists a ‘fixed, optimal’ access probability $p_i(t)=p_i(G, \lambda)$ for throughput-optimality
  - Also propose updating rules of $p_i(t)$ at time $T, 2T, 3T, ...$
    
    “$p_i(2T)=p_i(T)\pm \varepsilon$ depending on whether queue i increases in time $[T,2T]$”
  
  - And ‘conjecture’ that $p_i(t)$ converges to $p_i(G, \lambda)$, and hence throughput optimal

- [Jiang, Shah, S. and Walrand 2008]* provide provable $T$ and $\varepsilon$

How to Choose Access Probability in CSMA: Rate-based

- [Jiang and Walrand 2007] prove that
  - There exists a `fixed, optimal' access probability $p_i(t)=p_i(G, \lambda)$ for throughput-optimality
  - Also propose updating rules of $p_i(t)$ at time $T, 2T, 3T, ...$

  \[ p_i(2T)=p_i(T)\pm\varepsilon \]
  depending on whether queue $i$ increases in time $[T,2T]$"

- And `conjecture' that $p_i(t)$ converges to $p_i(G, \lambda)$, and hence throughput optimal

- [Jiang, Shah, S. and Walrand 2008]* provide provable $T$ and $\varepsilon$

- Further improvements have been made, for example
  - [Liu, Yi, Proutiere, Chiang and Poor 2009] proved that even $T=1$ works.
  - [Chaporkar and Proutiere 2013] developed an algorithm even in SNR model
  - [Lee, Lee, Yi, Chong, Nardelli, Knightly and Chiang 2013] implemented them in 802.11

How to Choose Access Probability in CSMA: Queue-based

\[
\frac{1}{f(Q_i(t))}
\]
How to Choose Access Probability in CSMA:

Queue-based

- Another approach is choosing access probability $p_i(t) = 1 - \frac{1}{f(Q_i(t))}$
  - $Q_i(t) =$ Queue-size at time $t$ and $f$ is some increasing function
  - It is called Queue-based CSMA
How to Choose Access Probability in CSMA: Queue-based

- Another approach is choosing access probability $p_i(t) = 1 - \frac{1}{f(Q_i(t))}$
  - $Q_i(t) =$ Queue-size at time $t$ and $f$ is some increasing function
  - It is called Queue-based CSMA
  - However, it is known to be harder to analyze
How to Choose Access Probability in CSMA:

Queue-based

• Another approach is choosing access probability $p_i(t) = 1 - \frac{1}{f(Q_i(t))}$
  
  - $Q_i(t) =$ Queue-size at time $t$ and $f$ is some increasing function
  
  - It is called Queue-based CSMA
  
  - However, it is known to be harder to analyze
  
  - Simulations on 10 by 10 grid graph
How to Choose Access Probability in CSMA: Queue-based

- Another approach is choosing access probability $p_i(t) = 1 - \frac{1}{f(Q_i(t))}$
  - $Q_i(t)$ = Queue-size at time $t$ and $f$ is some increasing function
  - It is called Queue-based CSMA
  - However, it is known to be harder to analyze
  - Simulations on 10 by 10 grid graph

Which function $f$ is best?

![Graph showing performance of different functions](image)
How to Choose Access Probability in CSMA:
Summary and My Contribution
How to Choose Access Probability in CSMA: Summary and My Contribution

- Two recent approaches

<table>
<thead>
<tr>
<th></th>
<th>Rate-based CSMA</th>
<th>Queue-based CSMA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td>Easier to analyze</td>
<td>Easier to implement</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>Less robust</td>
<td>Harder to analyze</td>
</tr>
<tr>
<td><strong>Main Question</strong></td>
<td>How to design updating rules?</td>
<td>How to choose a function f?</td>
</tr>
</tbody>
</table>
How to Choose Access Probability in CSMA: Summary and My Contribution

- Two recent approaches

<table>
<thead>
<tr>
<th>Pros</th>
<th>Rate-based CSMA</th>
<th>Queue-based CSMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cons</td>
<td>Less robust</td>
<td>Harder to analyze</td>
</tr>
<tr>
<td>Main Question</td>
<td>How to design updating rules?</td>
<td>How to choose a function f?</td>
</tr>
<tr>
<td>My Contribution: First provable answers for both questions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rate-based CSMA: Easier to analyze, Easier to implement

Queue-based CSMA: Less robust, Harder to analyze
How to Choose Access Probability in CSMA: Summary and My Contribution

- Two recent approaches

<table>
<thead>
<tr>
<th></th>
<th>Rate-based CSMA</th>
<th>Queue-based CSMA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td>Easier to analyze</td>
<td>Easier to implement</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>Less robust</td>
<td>Harder to analyze</td>
</tr>
<tr>
<td><strong>Main Question</strong></td>
<td>How to design updating rules?</td>
<td>How to choose a function $f$?</td>
</tr>
<tr>
<td><strong>My Contribution:</strong> First provable answers for both questions</td>
<td>Previous slide: First provable throughput-optimal updating rule*</td>
<td></td>
</tr>
</tbody>
</table>

# How to Choose Access Probability in CSMA: Summary and My Contribution

- **Two recent approaches**

<table>
<thead>
<tr>
<th></th>
<th>Rate-based CSMA</th>
<th>Queue-based CSMA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td>Easier to analyze</td>
<td>Easier to implement</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>Less robust</td>
<td>Harder to analyze</td>
</tr>
<tr>
<td><strong>Main Question</strong></td>
<td>How to design updating rules ?</td>
<td>How to choose a function f ?</td>
</tr>
<tr>
<td><strong>My Contribution:</strong> First provable answers for both questions</td>
<td>Previous slide: First provable throughput-optimal updating rule*</td>
<td>Next slide: First provable throughput-optimal function†</td>
</tr>
</tbody>
</table>

† Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009
Network Adiabatic Theorem


Queue-based CSMA with $f(x) \approx \log x$ is throughput-optimal for general interference graph

- If any (even centralized) other algorithm can stabilize the network, this algorithm can also do it

* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009
Network Adiabatic Theorem


Queue-based CSMA with $f(x) \approx \log x$ is throughput-optimal for general interference graph

- If any (even centralized) other algorithm can stabilize the network, this algorithm can also do it

- We show positive recurrence (i.e. stability) of underlying network Markov chain

* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009
Network Adiabatic Theorem


Queue-based CSMA with $f(x) \approx \log x$ is throughput-optimal for general interference graph

- If any (even centralized) other algorithm can stabilize the network, this algorithm can also do it
- We show positive recurrence (i.e. stability) of underlying network Markov chain

**Proof requires**

- Queueing theory (e.g. Lyapunov-Foster theorem)
- Information theory (e.g. Gibbs maximal principle)
- Spectral theory for matrices (e.g. Cheeger’s inequality)
- Combinatorics (e.g. Markov chain tree theorems)
- Martingales (e.g. Doob’s optional sampling theorem)

* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009
Network Adiabatic Theorem


Queue-based CSMA with \( f(x) \approx \log x \) is throughput-optimal for general interference graph

- If any (even centralized) other algorithm can stabilize the network, this algorithm can also do it
- We show positive recurrence (i.e. stability) of underlying network Markov chain

- Proof requires
  - Queueing theory (e.g. Lyapunov-Foster theorem)
  - Information theory (e.g. Gibbs maximal principle)
  - Spectral theory for matrices (e.g. Cheeger’s inequality)
  - Combinatorics (e.g. Markov chain tree theorems)
  - Martingales (e.g. Doob’s optional sampling theorem)

- Next: Why we choose \( f(x) \approx \log x \) for the positive recurrence?

* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009
Network Adiabatic Theorem


Queue-based CSMA with $f(x) \approx \log x$ is

throughput-optimal for **general** interference graph

- If any (even centralized) other algorithm can stabilize the network, this algorithm can also do it

- We show positive recurrence (i.e. stability) of underlying network Markov chain


*Network Adiabatic Theorem* [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009

**Proof requires**

- Queueing theory (e.g. Lyapunov-Foster theorem)
- Information theory (e.g. Gibbs maximal principle)
- Spectral theory for matrices (e.g. Cheeger’s inequality)
- Combinatorics (e.g. Markov chain tree theorems)
- Martingales (e.g. Doob’s optional sampling theorem)

- Next : Why we choose $f(x) \approx \log x$ for the positive recurrence ?

why not $f(x) = x, x^{1/2}$ or $\log \log x$ ?
Proof Strategy for Positive Recurrence

- Network MC (Markov Chain) $X(t) = \{Q(t), A(t)\}$
  - $Q(t) = [Q_i(t)]$ : Vector of queue-sizes at time $t$
  - $A(t) = [A_i(t)] \in \{0, 1\}^n$ : Vector of attempting status at time $t$
Proof Strategy for Positive Recurrence

- Network MC (Markov Chain) $X(t)=\{Q(t), A(t)\}$
  - $Q(t)=[Q_i(t)]$: Vector of queue-sizes at time $t$
  - $A(t)=[A_i(t)]\in\{0, 1\}^n$: Vector of attempting status at time $t$

Want $Q(t)$ is stable
Proof Strategy for Positive Recurrence

- Network MC (Markov Chain) $X(t) = \{Q(t), A(t)\}$
  - $Q(t) = [Q_i(t)]$ : Vector of queue-sizes at time $t$
  - $A(t) = [A_i(t)] \in \{0, 1\}^n$ : Vector of attempting status at time $t$

- Proof Strategy
  - Step I : $A(t)$ is ‘stable’
  - Step II : Stability of $A(t)$ implies Stability of $Q(t)$
Proof Strategy for Positive Recurrence

- Network MC (Markov Chain) \( X(t) = \{Q(t), A(t)\} \)
  - \( Q(t) = [Q_i(t)] \): Vector of queue-sizes at time \( t \)
  - \( A(t) = [A_i(t)] \in \{0, 1\}^n \): Vector of attempting status at time \( t \)

- Proof Strategy
  - Step I : \( A(t) \) is ‘stable’
  - Step II : Stability of \( A(t) \) implies Stability of \( Q(t) \)

\[ Q(t) \leftrightarrow A(t) \]

Want \( Q(t) \) is stable
Proof Strategy for Positive Recurrence

- Network MC (Markov Chain) $X(t) = \{Q(t), A(t)\}$
  - $Q(t) = [Q_i(t)]$: Vector of queue-sizes at time $t$
  - $A(t) = [A_i(t)] \in \{0, 1\}^n$: Vector of attempting status at time $t$

Proof Strategy

- Step I: $A(t)$ is ‘stable’
- Step II: Stability of $A(t)$ implies Stability of $Q(t)$

Step I

- Observe that $A(t)$ is a ‘time-varying’ MC with transition matrix $P(t) = P(Q(t))$
Proof Strategy for Positive Recurrence

- Network MC (Markov Chain) \( X(t) = \{Q(t), A(t)\} \)
  - \( Q(t) = [Q_i(t)] : \) Vector of queue-sizes at time \( t \)
  - \( A(t) = [A_i(t)] \in \{0, 1\}^n : \) Vector of attempting status at time \( t \)

- Proof Strategy
  - Step I : \( A(t) \) is ‘stable’
  - Step II : Stability of \( A(t) \) implies Stability of \( Q(t) \)

- Step I
  - Observe that \( A(t) \) is a ‘time-varying’ MC with transition matrix \( P(t) = P(Q(t)) \)
  - Prove that Distribution of \( A(t) \) converges to Stationary distribution \( \pi(t) \) of \( P(t) \)
Proof Strategy for Positive Recurrence

- Network MC (Markov Chain) $X(t) = \{Q(t), A(t)\}$
  - $Q(t) = [Q_i(t)]$ : Vector of queue-sizes at time $t$
  - $A(t) = [A_i(t)] \in \{0, 1\}^n$ : Vector of attempting status at time $t$

Proof Strategy

- Step I : $A(t)$ is ‘stable’
- Step II : Stability of $A(t)$ implies Stability of $Q(t)$

Step I

- Observe that $A(t)$ is a ‘time-varying’ MC with transition matrix $P(t) = P(Q(t))$
- Prove that Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$

Main issue : $P(t)$ is time-varying

This talk

Want $Q(t)$ is stable
Proof Strategy for Positive Recurrence

- Network MC (Markov Chain) $X(t)=\{Q(t), A(t)\}$
  - $Q(t)=[Q_i(t)]$ : Vector of queue-sizes at time $t$
  - $A(t)=[A_i(t)]\in\{0,1\}^n$ : Vector of attempting status at time $t$

- Proof Strategy
  - Step I : $A(t)$ is ‘stable’
  - Step II : Stability of $A(t)$ implies Stability of $Q(t)$

- Step I
  - Observe that $A(t)$ is a ‘time-varying’ MC with transition matrix $P(t)=P(Q(t))$
  - Prove that Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$

Main issue :
$P(t)$ is time-varying

`If $P(t)$ changes slower than it mixes`
**Proof Strategy for Positive Recurrence**

- **Network MC (Markov Chain) $X(t) = \{Q(t), A(t)\}$**
  - $Q(t) = [Q_i(t)]$ : Vector of queue-sizes at time $t$
  - $A(t) = [A_i(t)] \in \{0, 1\}^n$ : Vector of attempting status at time $t$

- **Proof Strategy**
  - Step I : $A(t)$ is ‘stable’
  - Step II : Stability of $A(t)$ implies Stability of $Q(t)$

- **Step I**
  - Observe that $A(t)$ is a ‘time-varying’ MC with transition matrix $P(t) = P(Q(t))$
  - Prove that Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$

- **Main issue** : $P(t)$ is time-varying

- **If $P(t)$ changes slower than it mixes’**

- **Want $Q(t)$ is stable**

- **Next : Holds if $f = \log$**

**This talk**
Proof Strategy for Positive Recurrence

`If P(t) changes slower than it mixes’

Next : Holds if f=log

`If P(t) changes slower than it mixes’
Proof Idea for Positive Recurrence : Why log ?

- Step 1: Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if mixing speed of $P(t) >$ changing speed of $P(t)$
Proof Idea for Positive Recurrence : Why log?

- Step I : Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if

  \[
  \text{mixing speed of } P(t) > \text{changing speed of } P(t)
  \]

  \[
  \| \text{spectral gap of } P(t) = P(Q(t)) \]
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if

\[
\text{mixing speed of } P(t) > \text{ changing speed of } P(t)
\]

Each non-zero entry of $P(t)$ is

\[
\frac{1}{f(Q(t))} \quad \text{or} \quad 1 - \frac{1}{f(Q(t))}
\]

due to our algorithm design.
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if

$$ \text{mixing speed of } P(t) > \text{changing speed of } P(t) $$

$$ \frac{\text{spectral gap of } P(t) = P(Q(t))}{\text{changing speed of its entry } \frac{1}{f(Q(t))}} $$

Each non-zero entry of $P(t)$ is

$$ \frac{1}{f(Q(t))} \text{ or } \frac{1 - \frac{1}{f(Q(t))}}{f(Q(t))} $$

due to our algorithm design
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if

mixing speed of $P(t)$ > changing speed of $P(t)$

$\begin{align*}
&|\| \text{spectral gap of } P(t)=P(Q(t)) | \\
&|\| \\
&|\| \\
&\frac{1}{f(Q(t))^n} \\
&\frac{1}{f(Q(t))} \quad \frac{d}{dt} \frac{1}{f(Q(t))}
\end{align*}$

Each non-zero entry of $P(t)$ is 

$\frac{1}{f(Q(t))}$ or $\frac{1}{f(Q(t))}$

due to our algorithm design.
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if

$$\text{mixing speed of } P(t) > \text{changing speed of } P(t)$$

Each non-zero entry of $P(t)$ is $\frac{1}{f(Q(t))}$ or $\frac{1}{f(Q(t))}$ due to our algorithm design.

$$\frac{||}{f(Q(t))^n}$$

$$\frac{d}{dt} \frac{1}{f(Q(t))} \quad \frac{f'(Q(t))}{f(Q(t))^2} \quad \frac{d Q(t)}{dt}$$

chain rule
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if

mixing speed of $P(t) >$ changing speed of $P(t)$

Each non-zero entry of $P(t)$ is

$$\frac{1}{f(Q(t))} \text{ or } 1 - \frac{1}{f(Q(t))}$$

due to our algorithm design
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if

  \[
  \text{mixing speed of } P(t) > \text{changing speed of } P(t)
  \]

  \[
  \text{spectral gap of } P(t) = P(Q(t)) \quad \text{and} \quad \frac{d}{dt} \frac{1}{f(Q(t))} \quad \frac{f'(Q(t))}{f(Q(t))^2}
  \]

  Each non-zero entry of $P(t)$ is

  \[
  \frac{1}{f(Q(t))} \quad \text{or} \quad 1 - \frac{1}{f(Q(t))}
  \]

  due to our algorithm design.
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of $A(t)$ converges to Stationary distribution $\pi(t)$ of $P(t)$ if

\[
\text{mixing speed of } P(t) > \text{changing speed of } P(t)
\]

Choose $f(x) = x$

Each non-zero entry of $P(t)$ is

\[
\frac{1}{f(Q(t))} \text{ or } \frac{1}{f(Q(t))}
\]
due to our algorithm design

Spectral gap of $P(t) = P(Q(t))$

Changing speed of its entry $\frac{1}{f(Q(t))}$

\[
\frac{d}{dt} \frac{1}{f(Q(t))} = \frac{f'(Q(t))}{f(Q(t))^2}
\]

Chain rule

\[
\frac{1}{f(Q(t))} \text{ or } \frac{1}{f(Q(t))}
\]

\[
\frac{1}{Q(t)} \text{ or } \frac{1}{Q(t)}
\]

\[
\frac{1}{f(Q(t))} \text{ or } \frac{1}{f(Q(t))}
\]
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of A(t) converges to Stationary distribution $\pi(t)$ of P(t) if

\[ \text{mixing speed of } P(t) > \text{changing speed of } P(t) \]

- Each non-zero entry of P(t) is

\[ \frac{1}{f(Q(t))} \text{ or } 1 - \frac{1}{f(Q(t))} \]

due to our algorithm design.
Proof Idea for Positive Recurrence: Why log?

- Step 1: Distribution of A(t) converges to Stationary distribution π(t) of P(t) if mixing speed of P(t) > changing speed of P(t)

Each non-zero entry of P(t) is 
\[ \frac{1}{f(Q(t))} \quad \text{or} \quad 1 - \frac{1}{f(Q(t))} \]

due to our algorithm design

\[ \frac{\text{spectral gap of } P(t) = P(Q(t))}{f(Q(t))^2} \]

\[ \frac{\text{changing speed of its entry}}{f(Q(t))} \]

Choose \( f(x) = \log x \)

\[ \frac{d}{dt} \frac{1}{f(Q(t))} \]

\[ \frac{f'(Q(t))}{f(Q(t))^2} \]

\[ \frac{1}{Q(t)(\log Q(t))^2} \]

\[ (\log Q(t))^n \]

\[ f(Q(t)) \]

\[ f(Q(t)) \]

Chain rule
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than $x^\epsilon$ e.g. $\log x$, $\log \log x$, $e^{\sqrt{\log x}}$ ...
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than $x^\varepsilon$ e.g. $\log x$, $\log \log x$, $e^{\sqrt{\log x}}$ ...

- Simulation on grid interference graph

![Graph showing throughput over time steps with functions $\log x$ and $x^{1/2}$]
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than $x^\epsilon$ e.g. $\log x$, $\log \log x$, $e^{\sqrt{\log x}}$ ...

- Simulation on grid interference graph
  - Tradeoff: Faster growing function is better for queue-size but worse for stability
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than $x^\varepsilon$ e.g. $\log x$, $\log \log x$, $e^{\sqrt{\log x}}$ ...

- Tradeoff: Faster growing function is better for queue-size but worse for stability

- Hence, the best function is the fastest growing one as long as it guarantees stability i.e.
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than $x^\epsilon$ e.g. $\log x$, $\log \log x$, $e^{\sqrt{\log x}}$ ...

- Simulation on grid interference graph
  - Tradeoff: Faster growing function is better for queue-size but worse for stability
  - Hence, the best function is the fastest growing one as long as it guarantees stability i.e.
    \[
    \text{mixing speed of } P(t) > \text{changing speed of } P(t)
    \]
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than $x^\varepsilon$ e.g. $\log x$, $\log \log x$, $e^{\sqrt{\log x}}$ ...

- Simulation on grid interference graph

- Tradeoff: Faster growing function is better for queue-size but worse for stability

- Hence, the best function is the fastest growing one as long as it guarantees stability i.e. mixing speed of $P(t) >$ changing speed of $P(t)$

Depends on spectral gap of a certain matrix called ‘Glauber dynamics’
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than $x^\epsilon$ e.g. $\log x$, $\log \log x$, $e^{\sqrt{\log x}}$ ...

Queue-size $\approx \frac{1}{\text{Spectral gap of Glauber dynamics}}$ if choose the best function $f$

- Tradeoff: Faster growing function is better for queue-size but worse for stability
- Hence, the best function is the fastest growing one as long as it guarantees stability i.e.

  $$\text{mixing speed of } P(t) > \text{changing speed of } P(t)$$

Depends on spectral gap of a certain matrix called ‘Glauber dynamics’
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than $x^\epsilon$ e.g. $\log x$, $\log \log x$, $e^{\sqrt{\log x}}$ ...

  \[
  \text{Queue-size} \approx \frac{1}{\text{Spectral gap of Glauber dynamics}} \text{ if choose the best function } f
  \]

- Hence, Queue-size = poly(n) or exp(n) depending on graph structure
  - Tradeoff: Faster growing function is better for queue-size but worse for stability

- Hence, the best function is the fastest growing one as long as it guarantees stability i.e.
  - Mixing speed of $P(t) >$ changing speed of $P(t)$
    
    Depends on spectral gap of a certain matrix called ‘Glauber dynamics’
Summary of Network Adiabatic Theorem

• In summary,

Designing a high performance medium access algorithm

\[ \pi(t) = \pi(Q(t)) \text{ satisfying MW property [Tassiulas and Ephremides 92]} \]

Medium access algorithm (Queue-based CSMA) = Distributed Iterative sampling mechanism
Summary of Network Adiabatic Theorem

• In summary,

Designing a high performance medium access algorithm

\[ \pi(t) = \pi(Q(t)) \] satisfying MW property [Tassiulas and Ephremides 92]

Medium access algorithm (Queue-based CSMA) = Distributed Iterative sampling mechanism
In summary,

Designing a high performance medium access algorithm

Sampling time-varying distribution $\pi(t)=\pi(Q(t))$ satisfying MW property [Tassiulas and Ephremides 92]

Medium access algorithm (Queue-based CSMA) = Distributed Iterative sampling mechanism

Next: We generalize this framework
Generalized Network Adiabatic Theorem

- [S. and Suk 2014]* establish the following generic framework

Designing a high performance combinatorial resource allocation algorithm

\[ \pi(t) = \pi(Q(t)) \text{ satisfying MW property} \]

Queue-based low-complexity and distributed mechanism:
Run only one iteration of the optimization method per each time

* Scheduling using Interactive Oracles [Shin and Suk] ACM SIGMETRICS 2014
Generalized Network Adiabatic Theorem

- [S. and Suk 2014] establish the following generic framework

Designing a high performance combinatorial resource allocation algorithm

\[ \pi(t) = \pi(Q(t)) \]

Satisfying MW property

- Queue-based low-complexity and distributed mechanism:
  - Run only one iteration of the optimization method per each time

Examples of iterative optimization methods: MCMC, Belief Propagation, Exhaustive Search

*Scheduling using Interactive Oracles [Shin and Suk] ACM SIGMETRICS 2014*
Generalized Network Adiabatic Theorem

- [S. and Suk 2014]* establish the following generic framework

Designing a high performance combinatorial resource allocation algorithm

\[ \text{Step II} \]

Iterative & distributed optimization methods computing time-varying distribution \( \pi(t) = \pi(Q(t)) \) satisfying MW property

\[ \text{Step I} \]

Queue-based low-complexity and distributed mechanism: Run only one iteration of the optimization method per each time

- Examples of iterative optimization methods: MCMC, Belief Propagation, Exhaustive Search
- We prove that throughput-optimality is guaranteed if \( f \) grow slower than the logarithm of the convergence time of an iterative method

* Scheduling using Interactive Oracles [Shin and Suk] ACM SIGMETRICS 2014
Generalized Network Adiabatic Theorem

- [S. and Suk 2014]* establish the following generic framework

Designing a high performance combinatorial resource allocation algorithm

\[ \text{Step II} \]

Iterative & distributed optimization methods computing time-varying distribution \( \pi(t) = \pi(Q(t)) \) satisfying MW property

\[ \text{Step I} \]

Queue-based low-complexity and distributed mechanism:
Run only one iteration of the optimization method per each time

- Examples of iterative optimization methods: MCMC, Belief Propagation, Exhaustive Search

- We prove that throughput-optimality is guaranteed if \( f \) grow slower than the logarithm of the convergence time of an iterative method

- Network Adiabatic Theorem is a special case for medium access using MCMC
Pick-and-Compare [Tassiulas 1998] is a special case using Exhaustive Search

* Scheduling using Interactive Oracles [Shin and Suk] ACM SIGMETRICS 2014
Time-varying Network Adiabatic Theorem

• [Yun, S. and Yi 2013]* Suppose channel states are time-varying

   Designing a high performance medium access algorithm

   Sampling time-varying distribution $\pi(t)=\pi(Q(t))$ satisfying MW property [Tassiulas and Ephremides 92]

   Medium access algorithm (Queue-based CSMA) = Distributed Iterative sampling mechanism

* Distributed Medium Access over Time-varying Channels [Yun, Shin and Yi] ACM MOBIHOC 2013
Time-varying Network Adiabatic Theorem

- [Yun, S. and Yi 2013]* Suppose channel states are time-varying

Designing a high performance medium access algorithm

\[ \pi(t) = \pi(Q(t)) \text{ satisfying MW property [Tassiulas and Ephremides 92]} \]

Medium access algorithm (Queue-based CSMA) = Distributed Iterative sampling mechanism

- We prove that throughput-optimality is guaranteed if \( f(x) = (\log x)^c \) where \( c \) is the current channel state

* Distributed Medium Access over Time-varying Channels [Yun, Shin and Yi] ACM MOBIHOC 2013
Time-varying Network Adiabatic Theorem

- [Yun, S. and Yi 2013]* Suppose channel states are time-varying

Designing a high performance medium access algorithm

\[
\text{Sampling time-varying distribution } \pi(t) = \pi(Q(t)) \text{ satisfying MW property [Tassiulas and Ephremides 92]}
\]

Medium access algorithm (Queue-based CSMA) = Distributed Iterative sampling mechanism

- We prove that throughput-optimality is guaranteed if \( f(x) = (\log x)^c \) where \( c \) is the current channel state
- We also prove that \( \log f(x) \) should be linear with respect to \( c \) for throughput-optimality
My Contribution for Distributed Scheduling

Queue-based Algorithms

- Network adiabatic theorem for medium access
- Generalized network adiabatic theorem
  [S. and Suk] SIGMETRICS 2014
- Network adiabatic theorem without message passing
  [Shah, S. and Prasad] FOCS 2011
- O(1) delay for medium access
  [Shah and S.] SIGMETRICS 2010
  [Lee, Yun, Yun, S. and Yi] INFOCOM 2014
- Time-varying network adiabatic theorem
  [Yun, S. and Yi] MOBIHOC 2013
My Contribution for Distributed Scheduling

Queue-based Algorithms

Network adiabatic theorem for medium access

Generalized network adiabatic theorem
[S. and Suk] SIGMETRICS 2014

Local stability is enough for global stability for multi-hop queueing networks

O(1) delay for medium access
[Shah and S.] SIGMETRICS 2010

Time-varying network adiabatic theorem
[Yun, S. and Yi] MOBIHOC 2013

[Lee, Yun, Yun, S. and Yi] INFOCOM 2014

Network adiabatic theorem without message passing
[Shah, S. and Prasad] FOCS 2011
My Contribution for Distributed Scheduling

Queue-based Algorithms

Network adiabatic theorem for medium access

Generalized network adiabatic theorem
[S. and Suk] SIGMETRICS 2014

Local stability is enough for global stability for multi-hop queueing networks

O(1) delay for medium access
[Shah. and S.] SIGMETRICS 2010
[Lee, Yun, Yun, S. and Yi] INFOCOM 2014

Time-varying network adiabatic theorem
[Yun, S. and Yi] MOBIHOC 2013

Rate-based Algorithms

Throughput optimality for rate-based CSMA

Belief Propagation for medium access

Network adiabatic theorem without message passing
[Shah, S. and Prasad] FOCS 2011

O(1) delay for medium access
[Shah. and S.] SIGMETRICS 2010

Belief Propagation for medium access
- Part II -
Message Passing in Statistical Networks

- Convergence and Correctness of Belief Propagation
  Joint work with Sungsoo Ahn (KAIST), Michael Cherkov (LANL) and Sejun Park (KAIST)
Graphical Model

• A graphical model (GM) is a way to represent probabilistic relationships between random variables through a graph

• Factor Graph for GM
Graphical Model

A graphical model (GM) is a way to represent probabilistic relationships between random variables through a graph.

A joint distribution of \( n \) (binary) random variables \( Z = [Z_i] \in \{0, 1\}^n \) is called a Graphical Model (GM) if it factorizes as follows: for \( z = [z_i] \in \Omega^n \),

\[
\Pr[Z = z] \propto \prod_{i \in \{1, \ldots, n\}} \psi_i(z_i) \prod_{\alpha \in F} \psi_\alpha(z_\alpha),
\]

where \( \{\psi_i, \psi_\alpha\} \) are (given) non-negative functions, the so-called factors; \( F \) is a collection of subsets

\[
F = \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subseteq 2^{\{1, 2, \ldots, n\}}
\]

Factor Graph for GM

Figure 1: Factor graph for the graphical model \( \Pr[z] \propto \psi_{\alpha_1}(z_1, z_3)\psi_{\alpha_2}(z_1, z_2, z_4)\psi_{\alpha_3}(z_2, z_3, z_4) \), i.e., \( F = \{\alpha_1, \alpha_2, \alpha_3\} \) and \( n = 4 \). Each \( \alpha_j \) selects a subset of \( z \). For example, \( \alpha_1 \) selects \( \{z_1, z_3\} \).
Computational Challenges in Graphical Model

- A graphical model (GM) is a way to represent probabilistic relationships between random variables through a graph.

A joint distribution of $n$ (binary) random variables $Z = [Z_i] \in \{0, 1\}^n$ is called a Graphical Model (GM) if it factorizes as follows: for $z = [z_i] \in \Omega^n$,

$$\Pr[Z = z] \propto \prod_{i \in \{1, \ldots, n\}} \psi_i(z_i) \prod_{\alpha \in F} \psi_{\alpha}(z_{\alpha}),$$

where $\{\psi_i, \psi_{\alpha}\}$ are (given) non-negative functions, the so-called factors; $F$ is a collection of subsets $F = \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subseteq 2^{\{1, 2, \ldots, n\}}$.

- Two fundamental questions: Compute
  - (Marginal probability) $P[Z_i = 1]$
  - (MAP) arg max $p(Z)$
Computational Challenges in Graphical Model

- A graphical model (GM) is a way to represent probabilistic relationships between random variables through a graph.

A joint distribution of $n$ (binary) random variables $Z = [Z_i] \in \{0, 1\}^n$ is called a Graphical Model (GM) if it factorizes as follows: for $z = [z_i] \in \Omega^n$,

$$\Pr[Z = z] \propto \prod_{i \in \{1, \ldots, n\}} \psi_i(z_i) \prod_{\alpha \in F} \psi_{\alpha}(z_{\alpha}),$$

where $\{\psi_i, \psi_{\alpha}\}$ are (given) non-negative functions, the so-called factors; $F$ is a collection of subsets $F = \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset 2\{1,2,\ldots,n\}$.

- Two fundamental questions: Compute
  - (Marginal probability) $P[Z_i = 1]$
  - (MAP) $\arg \max p(Z)$

- They are $\#P$ and NP hard
Computational Challenges in Graphical Model

- A graphical model (GM) is a way to represent probabilistic relationships between random variables through a graph.

A joint distribution of \( n \) (binary) random variables \( Z = [Z_i] \in \{0, 1\}^n \) is called a Graphical Model (GM) if it factorizes as follows: for \( z = [z_i] \in \Omega^n \),

\[
\Pr[Z = z] \propto \prod_{i \in \{1, \ldots, n\}} \psi_i(z_i) \prod_{\alpha \in F} \psi_\alpha(z_\alpha),
\]

where \( \{\psi_i, \psi_\alpha\} \) are (given) non-negative functions, the so-called factors; \( F \) is a collection of subsets

\[
F = \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset 2^{\{1,2,\ldots,n\}}
\]

- Two fundamental questions: Compute
  
  - (Marginal probability) \( P[Z_i = 1] \)
  - (MAP) \( \arg \max p(Z) \)

- They are \#P and NP hard

Need some heuristics or approximation algorithms!
Belief Propagation for Marginal Probability
Belief Propagation for Marginal Probability

- Consider a random variable $X = [X_v] \in \{0, 1\}^n$ and a tree graph $G$

\[
p[X] = \frac{1}{Z} \prod_{(u,v) \in E} (1 - X_u X_v)
\]

- Goal: Compute marginal probabilities $p[X_v=1]$ for all $v$
Belief Propagation for Marginal Probability

- Consider a random variable \( X = [X_v] \in \{0, 1\}^n \) and a tree graph \( G \).

\[
p[X] = \frac{1}{Z} \prod_{(u,v) \in E} (1 - X_u X_v)
\]

- Goal: Compute marginal probabilities \( p[X_v = 1] \) for all \( v \).

- (Divide & Conquer) Solve similar problems in sub-trees i.e.
Belief Propagation for Marginal Probability

- Consider a random variable $X = [X_v] \in \{0, 1\}^n$ and a tree graph $G$

$$p[X] = \frac{1}{\mathcal{Z}} \prod_{(u,v) \in E} (1 - X_u X_v)$$

- Goal: Compute marginal probabilities $p[X_v = 1]$ for all $v$

- (Divide & Conquer) Solve similar problems in sub-trees i.e.

$$\frac{p[X_v = 1]}{p[X_v = 0]} = \prod_{u \in N(v)} M_{u \rightarrow v}$$
Belief Propagation for Marginal Probability

- Consider a random variable $X = [X_v] \in \{0, 1\}^n$ and a tree graph $G$
  
  \[
p[X] = \frac{1}{Z} \prod_{(u,v) \in E} (1 - X_u X_v) \]

  - Goal: Compute marginal probabilities $p[X_v = 1]$ for all $v$
  
  - (Divide & Conquer) Solve similar problems in sub-trees i.e.

  \[
  \frac{p[X_v = 1]}{p[X_v = 0]} = \prod_{u \in N(v)} M_{u \rightarrow v} \]

  Marginal ratio $M_{u \rightarrow v}$ in sub-tree

  \[
  \frac{p[X_v = 1]}{p[X_v = 0]} \]
Belief Propagation for Marginal Probability

- Consider a random variable $X = [X_v] \in \{0, 1\}^n$ and a tree graph $G$

$$p[X] = \frac{1}{Z} \prod_{(u,v) \in E} (1 - X_u X_v)$$

- Goal: Compute marginal probabilities $p[X_v=1]$ for all $v$

- (Divide & Conquer) Solve similar problems in sub-trees i.e.

$$\frac{p[X_v=1]}{p[X_v=0]} = \prod_{u \in N(v)} M_{u \rightarrow v}$$

\[ M_{u \rightarrow v} \quad \text{Marginal ratio} \quad \frac{p[X_v=1]}{p[X_v=0]} \text{ in sub-tree} \]

\[ \frac{1 + \prod_{w \in N(u)/v} M_{w \rightarrow u}} {1 + \prod_{w \in N(u)/v} M_{w \rightarrow u}} \quad \text{Marginal ratio in sub-sub-tree} \]
Belief Propagation for Marginal Probability

- Consider a random variable $X = [X_v] \in \{0, 1\}^n$ and a tree graph $G$

$$p[X] = \frac{1}{Z} \prod_{(u,v) \in E} (1 - X_u X_v)$$

- Goal: Compute marginal probabilities $p[X_v=1]$ for all $v$

- (Divide & Conquer) Solve similar problems in sub-trees i.e.

$$\frac{p[X_v=1]}{p[X_v=0]} = \prod_{u \in N(v)} M_{u \rightarrow v}$$

$$\frac{p[X_v=1]}{p[X_v=0]} = \frac{1 + \prod_{w \in N(u)/v} M_{w \rightarrow u}}{1}$$

Marginal ratio in sub-sub-tree

Marginal ratio in sub-tree

Marginal ratio $\frac{p[X_v=1]}{p[X_v=0]}$ in sub-tree
Belief Propagation for Marginal Probability

- Consider a random variable $X = [X_v] \in \{0, 1\}^n$ and a tree graph $G$

\[
p[X] = \frac{1}{Z} \prod_{(u,v) \in E} (1 - X_u X_v)
\]

- Goal: Compute marginal probabilities $p[X_v = 1]$ for all $v$

- (Divide & Conquer) Solve similar problems in sub-trees i.e.

\[
\frac{p[X_v = 1]}{p[X_v = 0]} = \prod_{u \in N(v)} M_{u \rightarrow v}
\]

- Belief Propagation algorithm:

\[
M_{u \rightarrow v}^{t+1} = \frac{1}{1 + \prod_{w \in N(u)/v} M_{w \rightarrow u}^t}
\]

(Mu→v\(^0\) = 1/2)
Belief Propagation for Marginal Probability

- Consider a random variable $X = [X_v] \in \{0, 1\}^n$ and a tree graph $G$

  $$ p[X] = \frac{1}{Z} \prod_{(u,v) \in E} (1 - X_u X_v) $$

  - Goal: Compute marginal probabilities $p[X_v = 1]$ for all $v$

  - (Divide & Conquer) Solve similar problems in sub-trees i.e.

  $$ \frac{p[X_v = 1]}{p[X_v = 0]} = \prod_{u \in N(v)} M_{u \rightarrow v} $$

  - Belief Propagation algorithm:

    $$ M_{u \rightarrow v}^{t+1} = \frac{1}{1 + \prod_{w \in N(u) \setminus v} M_{w \rightarrow u}^t} $$

    $$ (M_{u \rightarrow v}^0 = 1/2) $$

    It is dynamic programming!
Belief Propagation (BP)

- BP is an iterative message-passing algorithm \( M^{t+1} = f_{BP}(M^t) \)
  - For tree graphical models, BP = Dynamic programming
  - BP for computing marginal probabilities is called “Sum-product BP (SBP)”
  - BP for computing MAP is called “Max-product BP (MBP)”
Belief Propagation (BP)

- BP is an iterative message-passing algorithm
  \[ M^{t+1} = f_{BP}(M^t) \]

  - For tree graphical models, BP = Dynamic programming
  - BP for computing marginal probabilities is called “Sum-product BP (SBP)”
  - BP for computing MAP is called “Max-product BP (MBP)”

- One can run BP for general graphical models
  - BP = ‘approximate’ Dynamic programming
  - However, its performance is not guaranteed if the underlying graph is not a tree
Belief Propagation (BP)

- BP is an iterative message-passing algorithm
  \[ M^{t+1} = f_{BP}(M^t) \]

  - For tree graphical models, BP = Dynamic programming
  - BP for computing marginal probabilities is called “Sum-product BP (SBP)”
  - BP for computing MAP is called “Max-product BP (MBP)”

- One can run BP for general graphical models
  - BP = ‘approximate’ Dynamic programming
  - However, its performance is not guaranteed if the underlying graph is not a tree

- Somewhat surprisingly, ‘loopy’ BPs have shown strong heuristic performance in many applications
  - e.g. error correcting codes (turbo codes), combinatorial optimization, statistical physics ...
  - It becomes a popular heuristic algorithm since it is easy to implement
Belief Propagation (BP)

- **BP** is an iterative message-passing algorithm:
  \[ M^{t+1} = f_{BP}(M^t) \]
  - For tree graphical models, **BP** = Dynamic programming
  - **BP** for computing marginal probabilities is called “Sum-product **BP** (SBP)”
  - **BP** for computing MAP is called “Max-product **BP** (MBP)”

- One can run **BP** for general graphical models
  - **BP** = ‘approximate’ Dynamic programming
  - However, its performance is not guaranteed if the underlying graph is not a tree

- Somewhat surprisingly, ‘loopy’ **BPs** have shown strong heuristic performance in many applications
  - e.g. error correcting codes (turbo codes), combinatorial optimization, statistical physics ...
  - It becomes a popular heuristic algorithm since it is **easy** to implement
How to understand Sum-product BP?

• First, \( M^c \) converges to a fixed point of \( f_{BP} \) (and how fast)?
  
  - A fixed point always exists due to the Brouwer fixed point theorem, but BP often diverges

• Second, a fixed (i.e. convergent) point of \( f_{BP} \) is good?
How to understand Sum-product BP?

- First, $M^t$ converges to a fixed point of $f_{BP}$ (and how fast)?
  - A fixed point always exists due to the Brouwer fixed point theorem, but BP often diverges
  - Convergence conditions: [Weiss 00] [Tatikonda and Jordan 02] [Heskes 04] [Ihler et al. 05]
  - However, they are very sensitive w.r.t potential functions and the underlying graph structure.

- Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?
  - Several efforts: [Wainwright et al. 03] [Heskes 04] [Yedidia et al. 04] [Chertkov et al. 06]
  - However, they provide different ‘views’ instead of simple conditions.
How to understand Sum-product BP?

• First, $M^c$ converges to a fixed point of $f_{BP}$ (and how fast)?
  - A fixed point always exists due to the Brouwer fixed point theorem, but BP often diverges

• Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?
How to understand Sum-product BP?

- First, $M^t$ converges to a fixed point of $f_{BP}$ (and how fast)?
  - A fixed point always exists due to the Brouwer fixed point theorem, but BP often diverges
  - Question: Exist an algorithm which can always compute a fixed point of $f_{BP}$ in poly-time?

- Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?
How to understand Sum-product BP?

- First, $M^t$ converges to a fixed point of $f_{BP}$ (and how fast)?
  - A fixed point always exists due to the Brouwer fixed point theorem, but BP often diverges
  - Question: Exist an algorithm which can always compute a fixed point of $f_{BP}$ in poly-time?
  - Converging algorithms [Teh and Welling 01] [Yuille 02], but no convergence rate

- Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?
How to understand Sum-product BP?

• First, $M^t$ converges to a fixed point of $f_{BP}$ (and how fast)?
  - A fixed point always exists due to the Brouwer fixed point theorem, but BP often diverges
  - Question: Exist an algorithm which can always compute a fixed point of $f_{BP}$ in poly-time?
  - Converging algorithms [Teh and Welling 01] [Yuille 02], but no convergence rate

[S. 2012]* Message passing algorithm always converging to a fixed point of $f_{BP}$ in $O(2^\Delta n^2)$ iterations for general (undirected) graphical model with max-degree $\Delta$

• Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?

How to understand Sum-product BP?

- First, $M^t$ converges to a fixed point of $f_{BP}$ (and how fast)?
  - A fixed point always exists due to the Brouwer fixed point theorem, but BP often diverges
  - Question: Exist an algorithm which can always compute a fixed point of $f_{BP}$ in poly-time?
  - Converging algorithms [Teh and Welling 01] [Yuille 02], but no convergence rate

[S. 2012]* Message passing algorithm always converging to a fixed point of $f_{BP}$ in $O(2^\Delta n^2)$ iterations for general (undirected) graphical model with max-degree $\Delta$

Any algorithm should take $\Omega(n)$ iterations

- Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?

How to understand Sum-product BP?

- First, $M^t$ converges to a fixed point of $f_{BP}$ (and how fast)?
  - A fixed point always exists due to the Brouwer fixed point theorem, but BP often diverges.
  - Question: Exist an algorithm which can always compute a fixed point of $f_{BP}$ in poly-time?
  - Converging algorithms [Teh and Welling 01] [Yuille 02], but no convergence rate.

[S. 2012]* Message passing algorithm always converging to a fixed point of $f_{BP}$ in $O(2^\Delta n^2)$ iterations for general (undirected) graphical model with max-degree $\Delta$

- It is as easy to implement as BP and a strong polynomial-time algorithm if $\Delta=O(\log n)$

- Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?

How to understand Sum-product BP?

- Question: Exist an algorithm which can always compute a fixed point of $f_{BP}$ in poly-time?
- Converging algorithms [Teh and Welling 01] [Yuille 02], but no convergence rate

[S. 2012]* Message passing algorithm always converging to a fixed point of $f_{BP}$ in $O(2^\Delta n^2)$ iterations for general (undirected) graphical model with max-degree $\Delta$

- It is as easy to implement as BP and a strong polynomial-time algorithm if $\Delta=O(\log n)$

- Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?

How to understand Sum-product BP?

- Question: Exist an algorithm which can always compute a fixed point of $f_{BP}$ in poly-time?
- Converging algorithms [Teh and Welling 01] [Yuille 02], but no convergence rate

[S. 2012]* Message passing algorithm always converging to a fixed point of $f_{BP}$ in $O(2^\Delta n^2)$ iterations for general (undirected) graphical model with max-degree $\Delta$

- It is as easy to implement as BP and a strong polynomial-time algorithm if $\Delta=O(\log n)$

- Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?

[S. et al. 2011]† Fixed points of $f_{BP}$ are good if girth $= \Omega(\log n)$

† Computing Indep. Sets using Bethe Approximation [Shin et al.] SIDMA 2011
Proof Strategy: How to find a fixed point of SBP

- Equivalent to find a zero-gradient point of $F_{\text{Bethe}}$ [Yedidia et al. 04]
  - $F_{\text{Bethe}} : \mathbb{D} \rightarrow \mathbb{R}$ is called the Bethe free energy function [Bethe 35]
  - The underlying domain $\mathbb{D}$ is a polytope

- It is not clear whether it is easy to find (or PLS-hard) since it is non-convex
Proof Strategy: How to find a fixed point of SBP

- Equivalent to find a zero-gradient point of $F_{\text{Bethe}}$ [Yedidia et al. 04]
  - $F_{\text{Bethe}} : D \rightarrow \mathbb{R}$ is called the Bethe free energy function [Bethe 35]
  - The underlying domain $D$ is a polytope
  - It is not clear whether it is easy to find (or PLS-hard) since it is non-convex

- Natural attempt: Gradient-descent algorithm

$$x^{t+1} = x^t + \alpha \nabla F_{\text{Bethe}}(x^t)$$ for some (step-size) $\alpha > 0$

- Does it find to a zero-gradient point? If not, why?
Proof Strategy : How to find a fixed point of LBP

- When does gradient-descent algorithm work for general $F$?
  - Its domain is unbounded and $|F|, |\nabla F|, |\nabla^2 F|$ are bounded $\rightarrow$ Possible to choose $\alpha$
Proof Strategy : How to find a fixed point of LBP

- **When does gradient-descent algorithm work for general \( F \)?**
  - Its domain is unbounded and \( |F|, |\nabla F|, |\nabla^2 F| \) are bounded → Possible to choose \( \alpha \)

- **Two issues for \( x^{t+1} = x^t + \alpha \nabla F_{\text{Bethe}}(x^t) \)**
  - Domain \( D \) is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of \( D \))
Proof Strategy: How to find a fixed point of LBP

• When does gradient-descent algorithm work for general $F$?
  - Its domain is unbounded and $|F|$, $|\nabla F|$, $|\nabla^2 F|$ are bounded $\rightarrow$ Possible to choose $\alpha$

• Two issues for $x^{t+1} = x^t + \alpha \nabla F_{\text{Bethe}}(x^t)$
  - Domain $D$ is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of $D$)

• Our main idea to resolve the issues
  
  Suppose $\nabla F_{\text{Bethe}}$ always points “inside” close to boundary of $D$
Proof Strategy : How to find a fixed point of LBP

- When does gradient-descent algorithm work for general \( F \)?
  
  - Its domain is unbounded and \(|F|, |\nabla F|, |\nabla^2 F|\) are bounded → Possible to choose \( \alpha \)

- Two issues for \( x^{t+1} = x^t + \alpha \nabla F_{\text{Bethe}}(x^t) \)
  
  - Domain \( D \) is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of \( D \))

- Our main idea to resolve the issues
  
  Suppose \( \nabla F_{\text{Bethe}} \) always points “inside” close to boundary of \( D \)
  
  - Then, it is possible choose small \( \alpha \) so that
    
    \[ x^t \text{ is always far from boundary } D \text{ i.e. } x^t \in D^* \subset D \]
Proof Strategy: How to find a fixed point of LBP

- When does gradient-descent algorithm work for general $F$?
  - Its domain is unbounded and $|F|, |\nabla F|, |\nabla^2 F|$ are bounded $\rightarrow$ Possible to choose $\alpha$

- Two issues for $x^{t+1} = x^t + \alpha \nabla F_{\text{Bethe}}(x^t)$
  - Domain $D$ is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of $D$)

- Our main idea to resolve the issues
  - Suppose $\nabla F_{\text{Bethe}}$ always points “inside” close to boundary of $D$
  - Then, it is possible choose small $\alpha$ so that
    $x^t$ is always far from boundary $D$ i.e. $x^t \in D^* \subset D$
Proof Strategy: How to find a fixed point of LBP

- When does gradient-descent algorithm work for general $F$?
  - Its domain is unbounded and $|F|, |\nabla F|, |\nabla^2 F|$ are bounded $\rightarrow$ Possible to choose $\alpha$

- Two issues for $x^{t+1} = x^t + \alpha \nabla F_{\text{Bethe}}(x^t)$
  - Domain $D$ is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of $D$)

- Our main idea to resolve the issues
  
  Suppose $\nabla F_{\text{Bethe}}$ always points “inside” close to boundary of $D$
  - Then, it is possible choose small $\alpha$ so that

  $x^t$ is always far from boundary $D$ i.e. $x^t \in D^* \subset D$
Proof Strategy : How to find a fixed point of LBP

• When does gradient-descent algorithm work for general $F$?
  - Its domain is unbounded and $|F|$, $|\nabla F|$, $|\nabla^2 F|$ are bounded $\rightarrow$ Possible to choose $\alpha$

• Two issues for $x^{t+1} = x^t + \alpha \nabla F_{\text{Bethe}}(x^t)$
  - Domain $D$ is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of $D$)

• Our main idea to resolve the issues
  
  Suppose $\nabla F_{\text{Bethe}}$ always points “inside” close to boundary of $D$
  - Then, it is possible choose small $\alpha$ so that

  \[ x^t \text{ is always far from boundary } D \text{ i.e. } x^t \in D^* \subset D \]
  - Unfortunately, this (or similar) seems not true ... but
Proof Strategy: How to find a fixed point of LBP

- When does gradient-descent algorithm work for general \( F \)?
  - Its domain is unbounded and \( |F|, |\nabla F|, |\nabla^2 F| \) are bounded \( \rightarrow \) Possible to choose \( \alpha \)

- Two issues for \( x^{t+1} = x^t + \alpha \nabla F_{\text{Bethe}}(x^t) \)
  - Domain \( D \) is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of \( D \))

- Our main idea to resolve the issues
  - Suppose \( \nabla F_{\text{Bethe}} \) always points “inside” close to boundary of \( D \)
  - Then, it is possible choose small \( \alpha \) so that \( x^t \) is always far from boundary \( D \) i.e. \( x^t \in D^* \subset D \)
  - Unfortunately, this (or similar) seems not true ... but

  We found a function \( G \) such that this property holds for \( G \) and one-to-one correspondence between zero-gradients of \( G \) and \( F_{\text{Bethe}} \)
How to understand Max-product BP?
How to understand Max-product BP?

- First, $M^t$ converges to a fixed point of $f_{BP}$ (and how fast)?

- Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?
How to understand Max-product BP?

• First, $M^t$ converges to a fixed point of $f_{BP}$ (and how fast)?
  - BP converges to the solution of LP (Linear Programming) associated to several combinatorial optimization problems, i.e., BP can solve LP!

• Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?
  - [Weiss and Freeman 2001] found a generic, called “single-loop-tree”, condition

* Graphical Transformation for Belief Propagation: MaximumWeight Matchings and Odd Cycles [Shin, Gelfand and Chertkov] NIPS 2013
How to understand Max-product BP?

• First, $M^c$ converges to a fixed point of $f_{BP}$ (and how fast)?
  
  - BP converges to the solution of LP (Linear Programming) associated to several combinatorial optimization problems, i.e., BP can solve LP!
  - [Park and S. 2015]† found a generic sufficient condition so that BP converges to the solution of LP

• Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?
  
  - [Weiss and Freeman 2001] found a generic, called “single-loop-tree”, condition

* Graphical Transformation for Belief Propagation: Maximum Weight Matchings and Odd Cycles [Shin, Gelfand and Chertkov] NIPS 2013
How to understand Max-product BP?

• First, $\mathbb{M}^t$ converges to a fixed point of $f_{BP}$ (and how fast)?
  
  - BP converges to the solution of LP (Linear Programming) associated to several combinatorial optimization problems, i.e., BP can solve LP!
  

  - [Park and S. 2015]† found a generic sufficient condition so that BP converges to the solution of LP

• Second, a fixed (i.e. convergent) point of $f_{BP}$ is good?

  - [Weiss and Freeman 2001] found a generic, called “single-loop-tree”, condition

---

* Graphical Transformation for Belief Propagation: MaximumWeight Matchings and Odd Cycles [Shin, Gelfand and Chertkov] NIPS 2013
† Max-product Belief Propagation for Linear Programming [Park and Shin] UAI 2015
Our Contribution: BP can solve LP?

- [Park and S. 2015]* BP converges to the solution of LP if
  - C1. LP has a unique and integral solution
  - C2. Each variable is associated to at most two factors
  - C3. For every factor $\psi_{\alpha}$, every $x_\alpha \in \{0, 1\}^{|\alpha|}$ with $\psi_{\alpha}(x_\alpha) = 1$, and every $i \in \alpha$ with $x_i \neq x_i^*$, there exists $\gamma \subset \alpha$ such that
    $$|\{j \in \{i\} \cup \gamma : |F_j| = 2\}| \leq 2$$
    $$\psi_{\alpha}(x'_{\alpha}) = 1,$$ where $x'_k = \begin{cases} x_k & \text{if } k \notin \{i\} \cup \gamma, \\ x_k^* & \text{otherwise} \end{cases}$$
    $$\psi_{\alpha}(x''_{\alpha}) = 1,$$ where $x''_k = \begin{cases} x_k & \text{if } k \in \{i\} \cup \gamma, \\ x_k^* & \text{otherwise} \end{cases}$$

Our Contribution: BP can solve LP?

- [Park and S. 2015] BP converges to the solution of LP if
  
  - C1. LP has a unique and integral solution
  - C2. Each variable is associated to at most two factors
  - C3. "C3 is a only non-trivial condition, but typically easy to check given GM."

\[
\begin{align*}
\text{For every factor } \psi_\alpha, \text{ every } x_\alpha \in \{0, 1\}^{\alpha} \text{ with } \psi_\alpha(x_\alpha) = 1, \text{ and every } i \in \alpha \text{ with } x_i \neq x_i^*, \text{ there exists } \gamma \subset \alpha \text{ such that } \\
|\{j \in \{i\} \cup \gamma : |E_j| = 2\}| \leq 2
\end{align*}
\]

```
\begin{align*}
\psi_\alpha(x_\alpha') = 1, \quad \text{where } x_\alpha' = \begin{cases} x_k & \text{if } k \notin \{i\} \cup \gamma, \\
          x_i^* & \text{otherwise}
\end{cases} \\
\psi_\alpha(x_\alpha'') = 1, \quad \text{where } x_\alpha'' = \begin{cases} x_k & \text{if } k \in \{i\} \cup \gamma, \\
          x_i^* & \text{otherwise}
\end{cases}
\end{align*}
```

*Max-product Belief Propagation for Linear Programming [Park and Shin] UAI 2015*
Our Contribution: BP can solve LP?

- [Park and S. 2015]∗ BP converges to the solution of LP if
  - C1. LP has a unique and integral solution
  - C2. Each variable is associated to at most two factors
  - C3. For every factor $\psi_\alpha$, every $x_\alpha \in \{0, 1\}^{\alpha}$ with $\psi_\alpha(x_\alpha) = 1$, and every $i \in \alpha$ with $x_i \neq x_i^*$, there exists $\gamma \subset \alpha$ such that
    $$|\{j \in \{i\} \cup \gamma : |E_j| = 2\}| \leq 2$$

  “C3 is a only non-trivial condition, but typically easy to check given GM.”

Why BP can be better than simplex or interior-point methods?

- BP is easy to parallelize and implement in a distributed & parallel programming model

Examples of LP solvable by BP

**Shortest Path**

\[
\begin{align*}
\text{minimize} & \quad w \cdot x \\
\text{subject to} & \quad \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e \\
& \quad = \begin{cases} 
1 & \text{if } v = s \\
-1 & \text{if } v = t \quad \forall v \in V \\
0 & \text{otherwise}
\end{cases} \\
x &= [x_e] \in [0, 1]^{|E|}.
\end{align*}
\]

**Minimum Weight (Perfect) Matching**

\[
\begin{align*}
\text{minimize} & \quad w \cdot x \\
\text{subject to} & \quad \sum_{e \in \delta(v)} x_e = 1, \quad \forall v \in V \\
x &= [x_e] \in [0, 1]^{|E|}
\end{align*}
\]

**Vertex Cover**

\[
\begin{align*}
\text{minimize} & \quad b \cdot y \\
\text{subject to} & \quad y_u + y_v \geq 1, \quad (u, v) \in E \\
y &= [y_v] \in [0, 1]^{|V|}.
\end{align*}
\]

**Cycle Packing**

\[
\begin{align*}
\text{maximize} & \quad w \cdot x \\
\text{subject to} & \quad \sum_{e \in \delta(v)} x_e = 2y_v \\
x &= [x_e] \in [0, 1]^{|E|}, y = [y_v] \in [0, 1]^{|V|}
\end{align*}
\]

**Traveling Salesman Problem**

\[
\begin{align*}
\text{minimize} & \quad w \cdot x \\
\text{subject to} & \quad \sum_{e \in \delta(v)} x_e = 2 \\
x &= [x_e] \in [0, 1]^{|E|}
\end{align*}
\]

**Network Flow**

\[
\begin{align*}
\text{minimize} & \quad w \cdot x \\
\text{subject to} & \quad \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = d_v, \quad \forall v \in V \\
x &= [x_e] \in \mathbb{R}_+^{|E|}
\end{align*}
\]
My Contribution for Belief Propagation

• Sum-product BP
  - Polynomial-time algorithm for computing a BP fixed-point
  - Large-girth condition for correctness of BP

• Max-product BP
  - BP solves the LP for minimum weight matching using odd cycle constraints
    [S., Gelfand and Chertkov] NIPS 2013
  - Generic necessary condition so that BP solves LP
    [Gelfand, Chertkov and S.] ISIT 2013
  - Generic sufficient condition so that BP solves LP
    [Park and S.] UAI 2015
  - BP solves the IP for minimum weight matching
    [Ahn, Chertkov, Park and S.] submitted
Summary

Part I

Scheduling algorithms for communication networks (e.g. wireless communication)

Part II

Efficient algorithms for Statistical Inference (e.g. belief propagation)
Summary

Part I

Scheduling algorithms for communication networks (e.g. wireless communication)

Message Passing Algorithms

Efficient algorithms for Statistical Inference (e.g. belief propagation)

Part II
Why Message Passing Algorithms?

- They are crucial for numerous fields in engineering and social science
  - Building blocks for communication (Internet) networks
e.g. medium access, packet switching
  - Efficient estimation tools for statistical (Bayesian) networks
e.g. variational (or cavity) method, Markov chain monte carlo
  - Faithful behavioral models for societal systems
e.g. markets, auctions, recommendation systems

My Research = Principles of Local Rules for Networked Systems