Message Passing Algorithms: Communication, Inference and Optimization

Jinwoo Shin

KAIST EE

Message Passing Algorithms in My Research

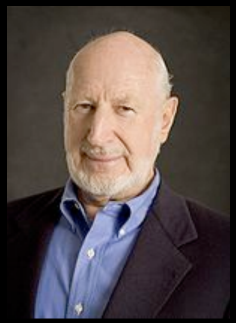
- Communication Networks (e.g. Internet)

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)



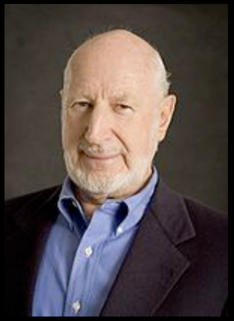
N. Abramson
(1970s)

- 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)



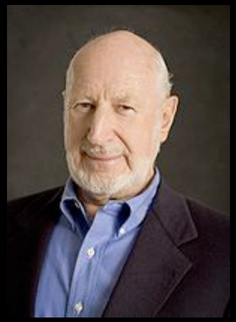
N. Abramson (1970s)

J. Tsitsiklis (1980s)

- Statistical Networks (e.g. Bayesian networks)
- Social Networks (e.g. Facebook)

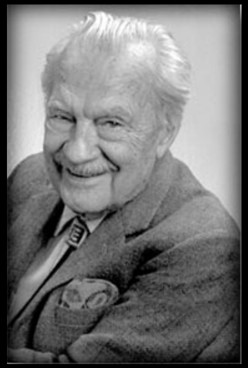
Message Passing Algorithms in My Research

- 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)



N. Abramson
(1970s)

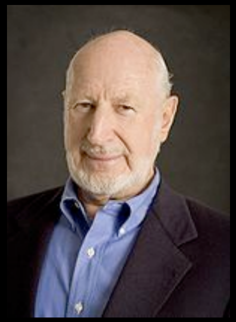
J. Tsitsiklis
(1980s)



N. Metropolis
(1950s)

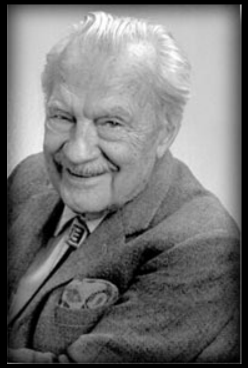
Message Passing Algorithms in My Research

- 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-respose)

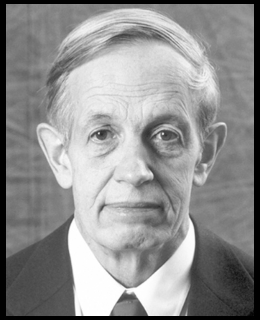


N. Abramson (1970s)

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-respose)

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)

Part I of This Talk

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

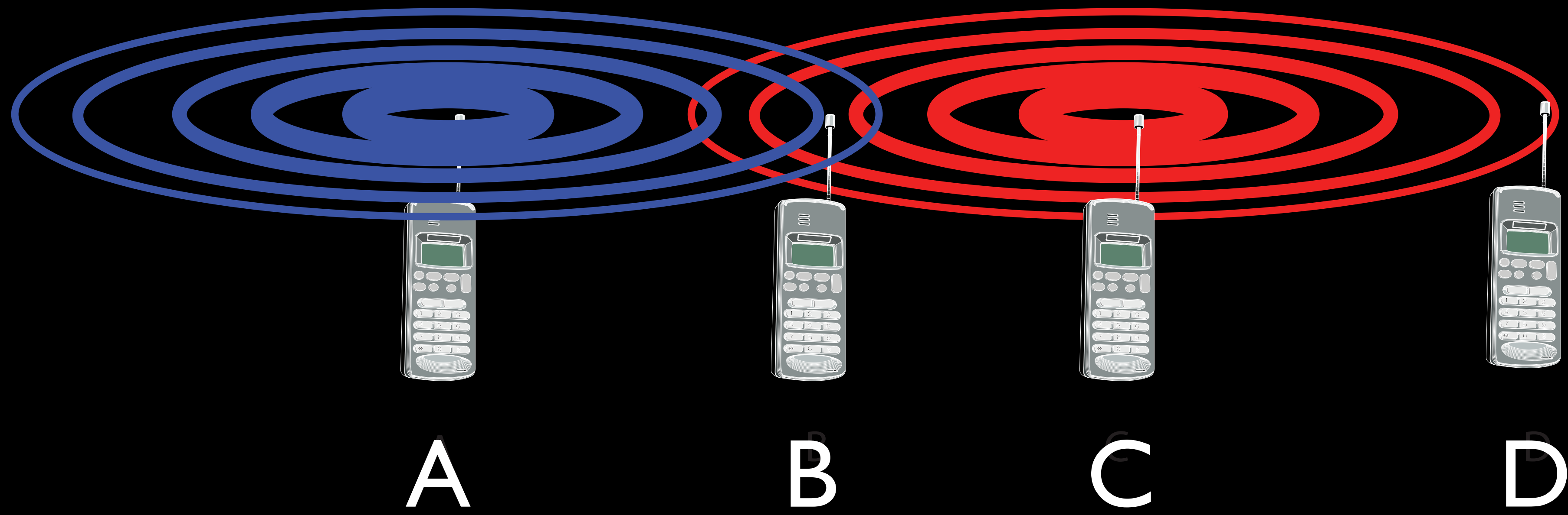
Part II of This Talk

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

- Part I -
Message Passing in Communication Networks

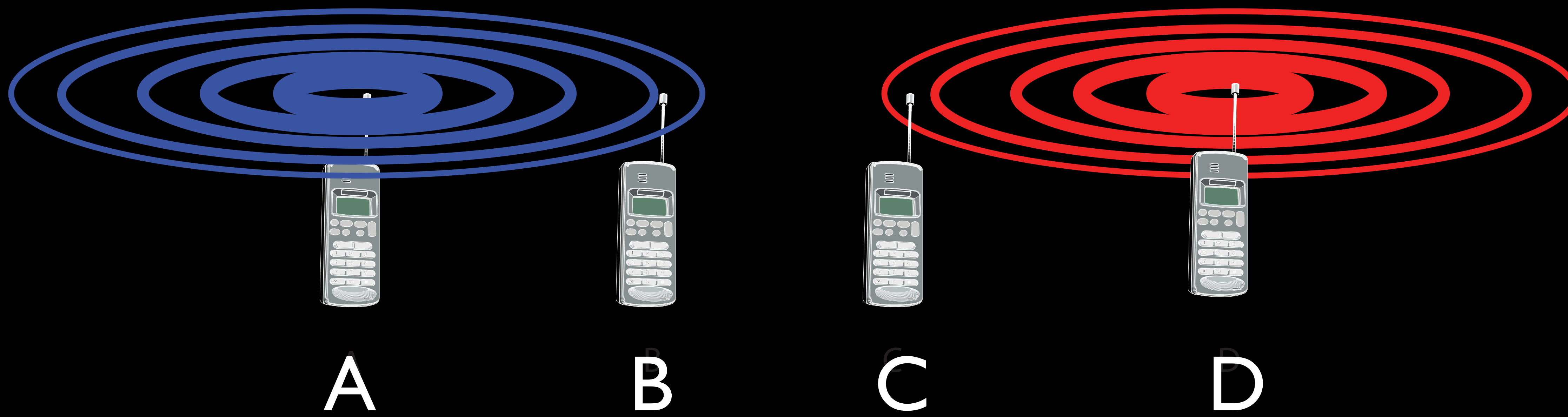
- Focus on medium access for wireless networks
  Joint work with Devavrat Shah (MIT)
- Later describe how the result is extended
  Joint work with Yung Yi (KAIST), Seyoung Yun,(MSR), Tonghoon Suk (Gatech)

Motivation : Wireless Network



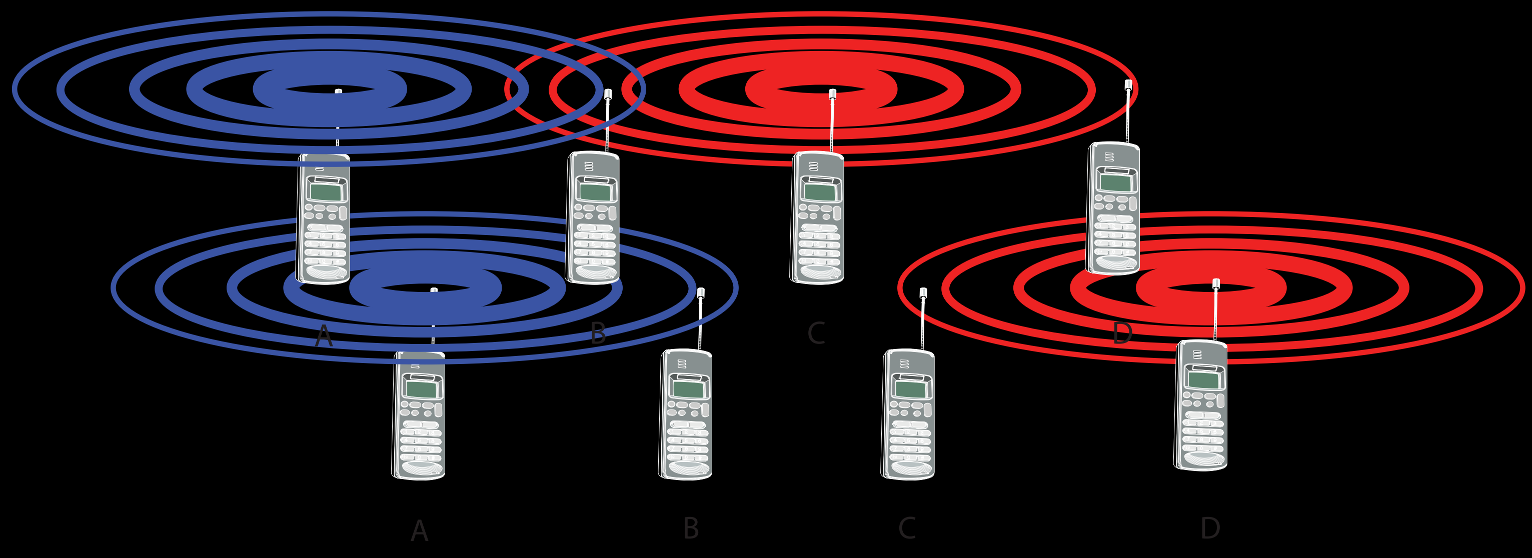
- A and C cannot send packets to B simultaneously
  - 'A->B' and 'C->B' interfere with each other & packets collide

Motivation : Wireless Network



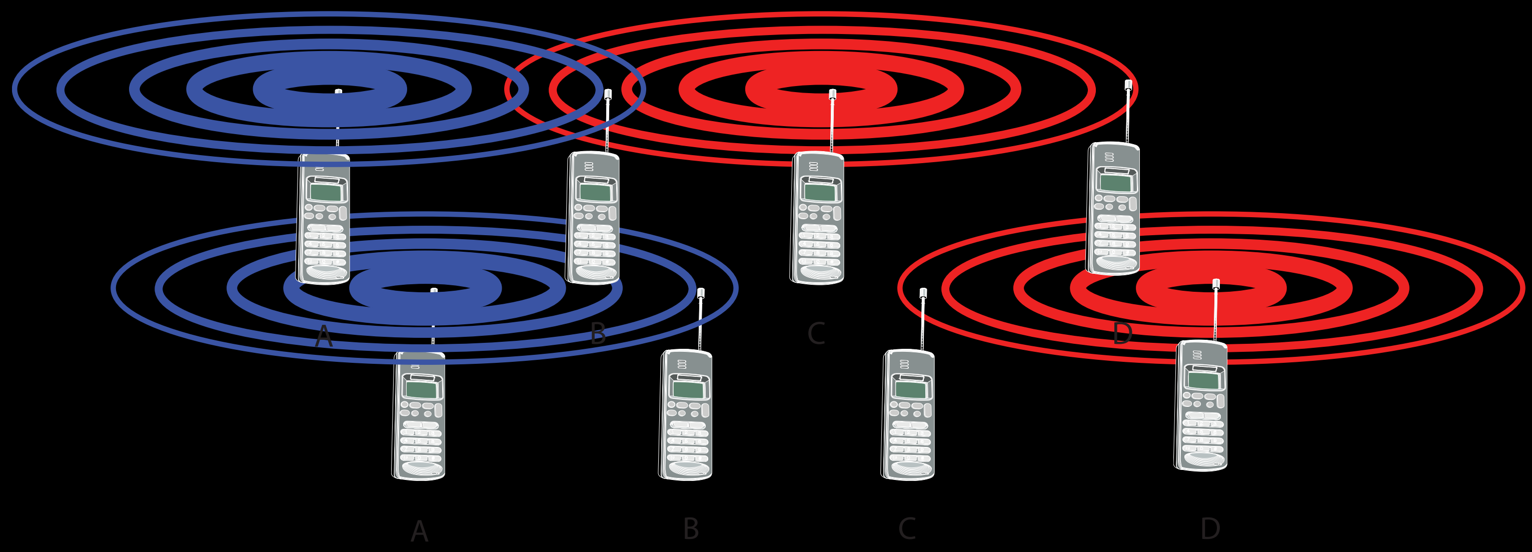
- However, ‘A->B’ and ‘D->C’ do not interfere with each other

Motivation : Wireless Network



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

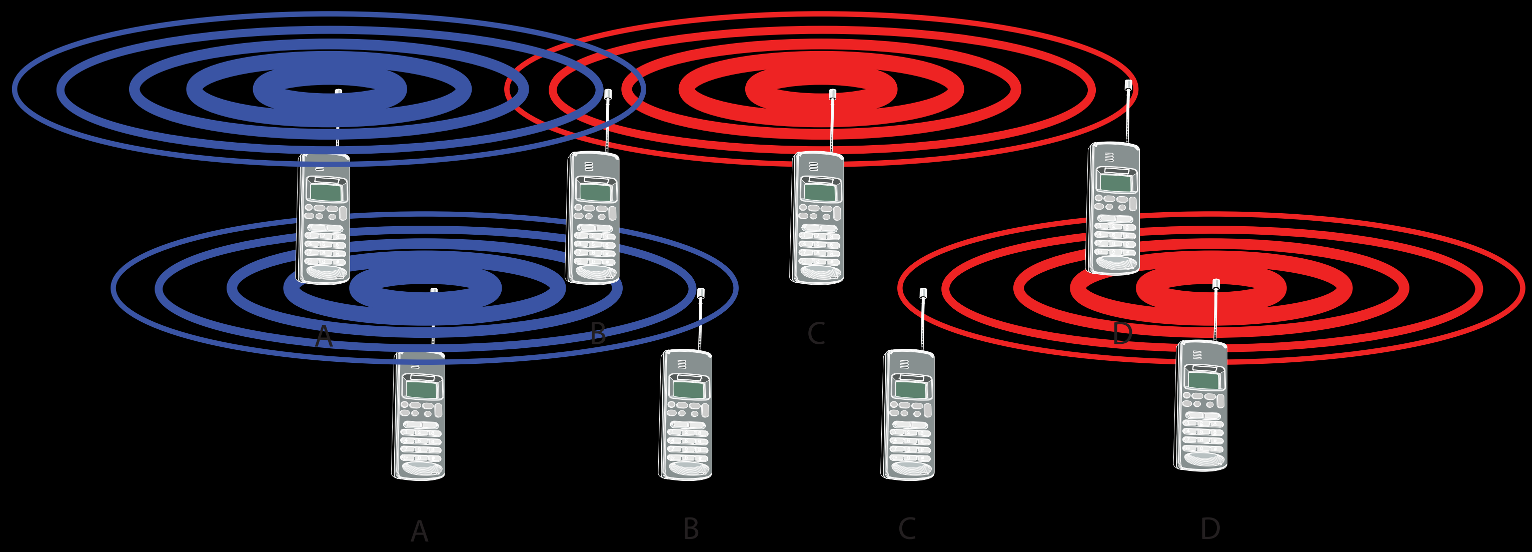
Motivation : Wireless Network



- 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

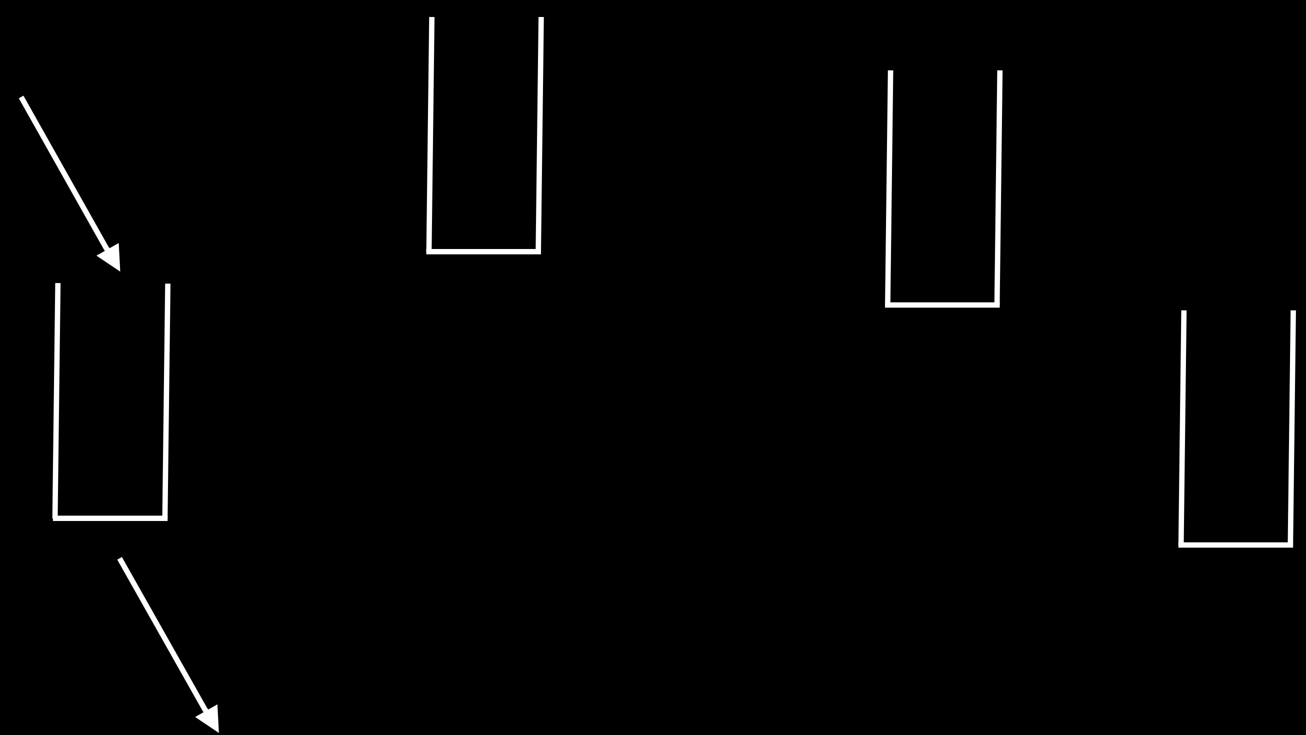
- Design a `provably’ optimal medium access algorithm

Next : Describe a mathematical model

Next : Definition of `optimality’

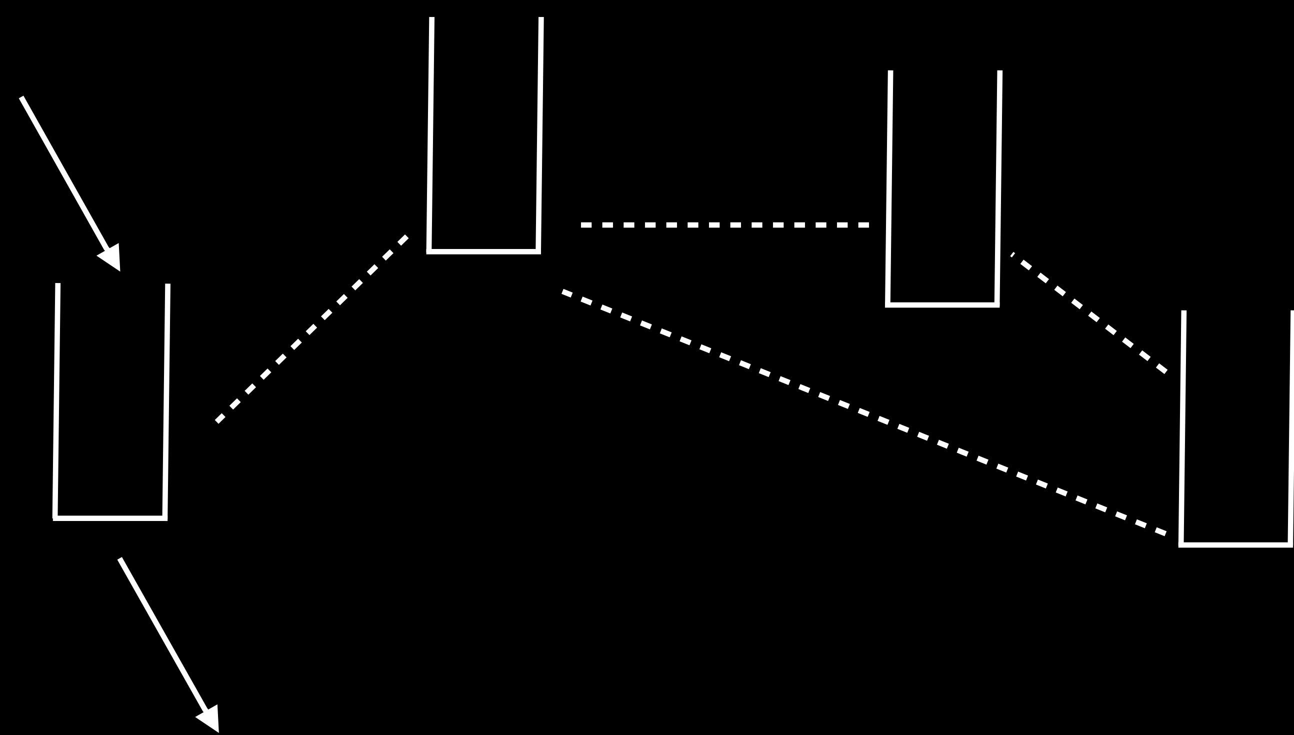
  - For simplicity, I will considers a simple model (i.e. discrete-time, single-hop, single-channel)
  - However, the same story goes through for other models (i.e. continuous-time,multi-hop, multi-channel, collisions, time-varying)

Model : Wireless Network



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

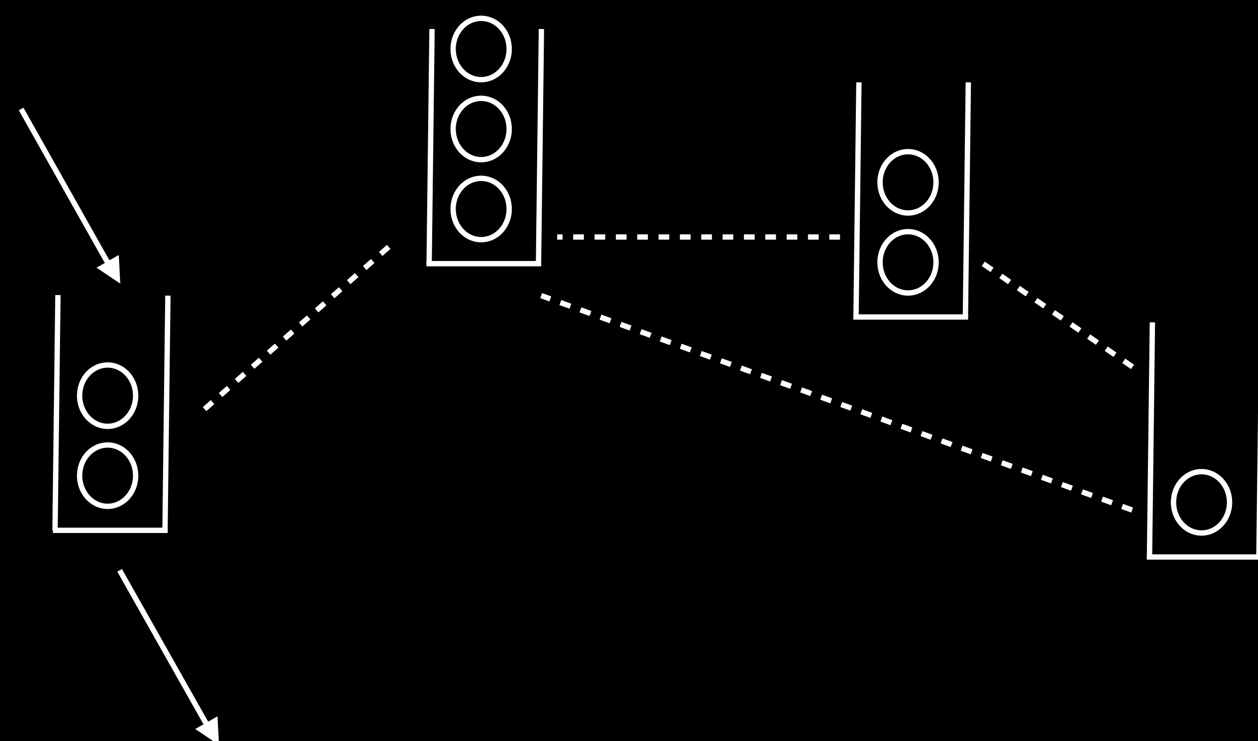
Model : Wireless Network



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

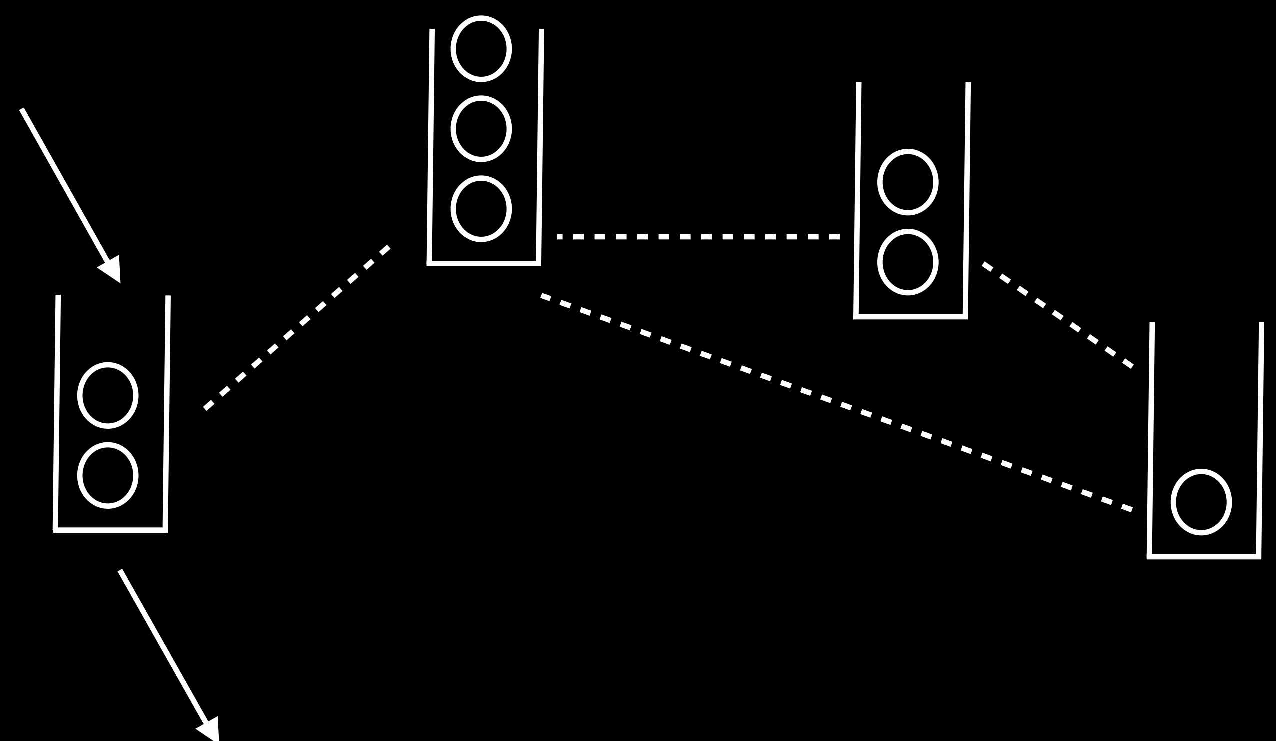
"Queues form vertices & two queues share an edge if they cannot transmit simultaneously"

Model : Wireless Network



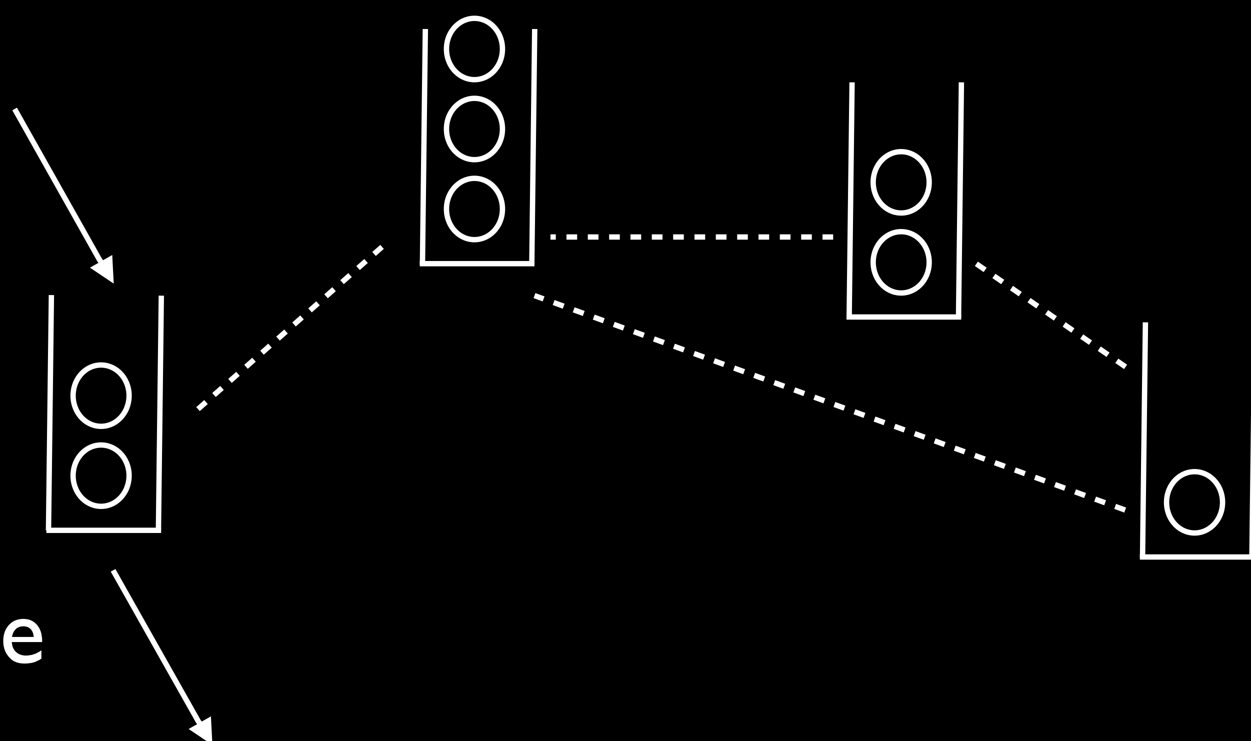
- We assume
  - Packets arrive at queue i with rate $\lambda(i)$ e.g. Bernoulli stochastic process
  - Time is discrete & at most one packet can depart from each queue at each time instance

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
- 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

Model : Wireless Network



- We assume
  - Packets arrive at queue i with rate λ(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

How to decide the length of the time interval ?

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

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 ?

- 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

How to decide the length of the time interval ?

If any algorithm can stabilize the network, the back-off protocol can also do it

- 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

How to decide the length of the time interval ?

If any algorithm can stabilize the network, the back-off protocol can also do it

- 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 from1970s 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 from1970s 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]"

    0 T 2T 3T 4T

  - 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]"

    0 T 2T 3T 4T

No proof

  - 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]"

    No proof

    0 T 2T 3T 4T

  - 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 ε

* Distributed Random Access Algorithm [Jiang, Shah, Shin and Walrand] IEEE Transactions on Information Theory 2010

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]"

    0 T 2T 3T 4T

    No proof

  - 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 ε
- 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

* Distributed Random Access Algorithm [Jiang, Shah, Shin and Walrand] IEEE Transactions on Information Theory 2010

How to Choose Access Probability in CSMA : Queue-based

$$\frac{1}{f(Q_i(t))}$$

How to Choose Access Probability in CSMA : Queue-based

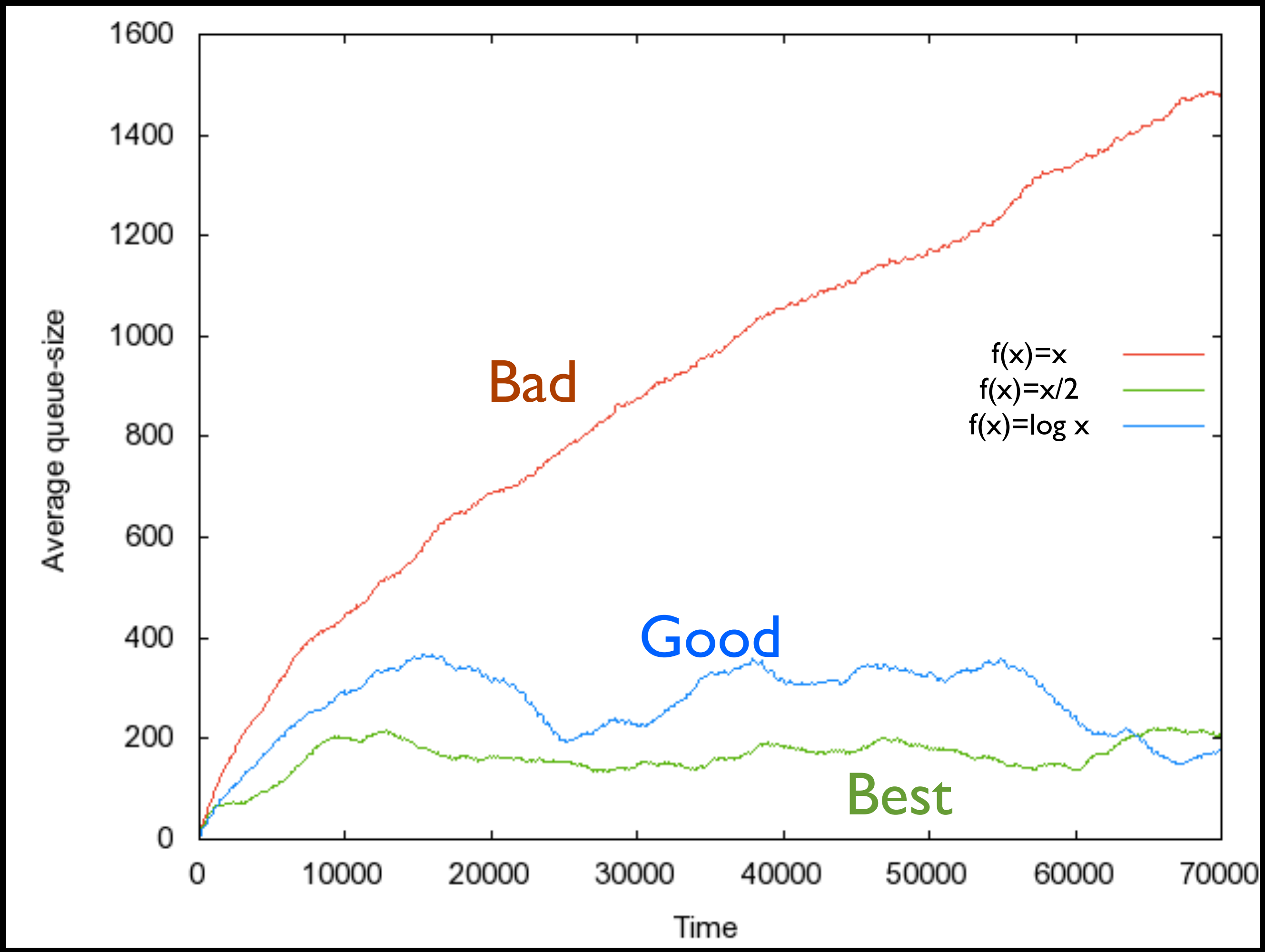
- Another approach is choosing access probability $p_i(t) = 1 - \frac{1}{f(Q_i(t))}$
  - $Q_i(t)$ = Queue-size at time t and f is some increasing function
  - It is called Queue-based CSMA

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

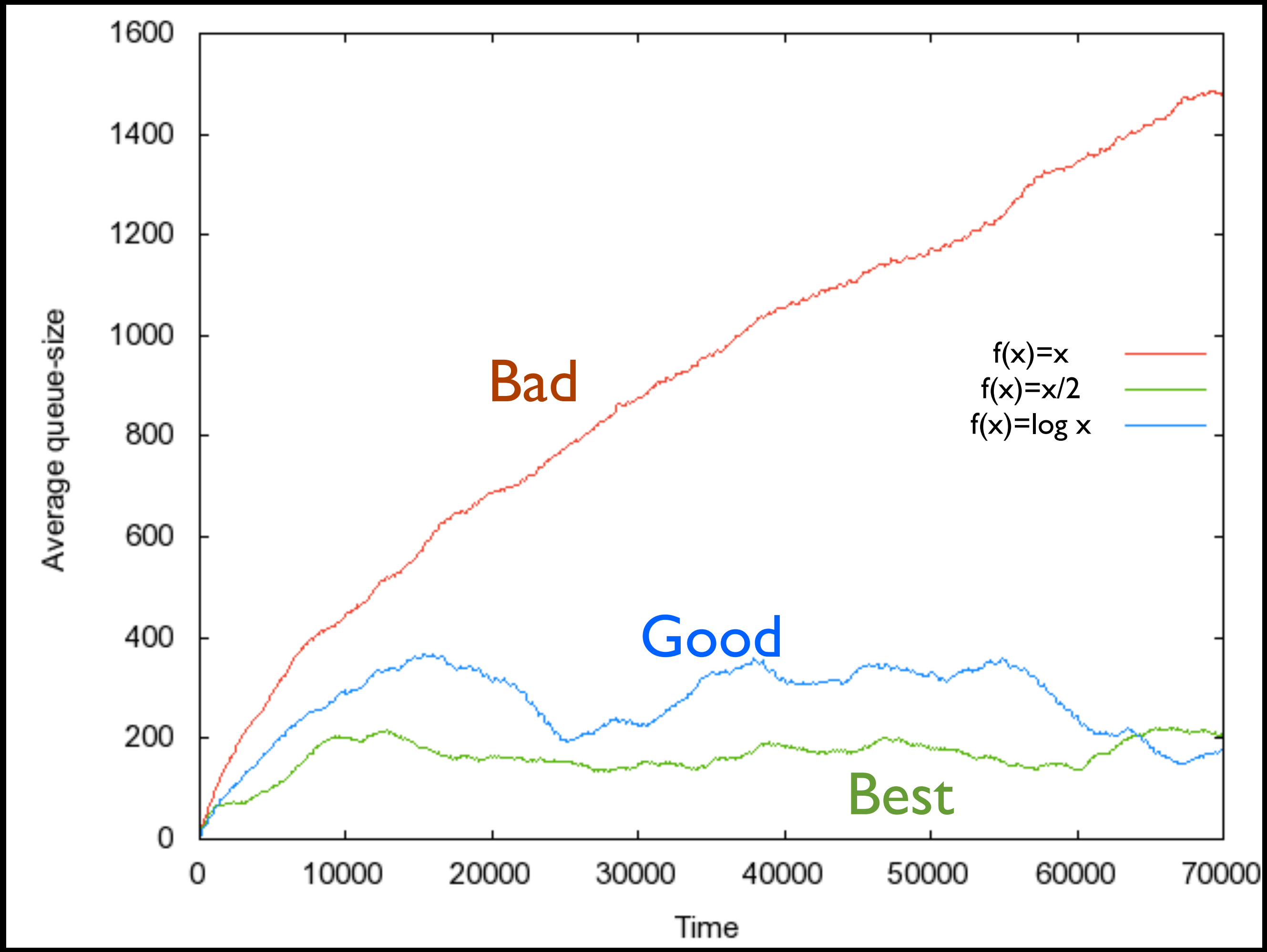
- Another approach is choosing access probability $p_i(t) = 1 - \frac{1}{f(Q_i(t))}$
  - $Q_i(t)$ = Queue-size at time t and f is some increasing function
  - It is called Queue-based CSMA
  - However, it is known to be harder to analyze
  - Simulations on 10 by 10 grid graph



How to Choose Access Probability in CSMA : Queue-based

- Another approach is choosing access probability $p_i(t) = 1 - \dfrac{1}{f(Q_i(t))}$
  - $Q_i(t)$ = 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 ?



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

	Rate-based CSMA	Queue-based CSMA
Pros	Easier to analyze	Easier to implement
Cons	Less robust	Harder to analyze
Main Question	How to design updating rules ?	How to choose a function f ?

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

- Two recent approaches

	Rate-based CSMA	Queue-based CSMA
Pros	Easier to analyze	Easier to implement
Cons	Less robust	Harder to analyze
Main Question	How to design updating rules ?	How to choose a function f ?
My Contribution : First provable answers for both questions		

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

- Two recent approaches

	Rate-based CSMA	Queue-based CSMA
Pros	Easier to analyze	Easier to implement
Cons	Less robust	Harder to analyze
Main Question	How to design updating rules ?	How to choose a function f ?
My Contribution : First provable answers for both questions	Previous slide : First provable throughput-optimal updating rule*	

* Distributed Random Access Algorithm [Jiang, Shah, Shin and Walrand] IEEE Transactions on Information Theory 2010

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

- Two recent approaches

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

* Distributed Random Access Algorithm [Jiang, Shah, Shin and Walrand] IEEE Transactions on Information Theory 2010

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

Network Adiabatic Theorem

- Theorem [Shah and S. 2009, 2010, 2011]*

  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

* Medium Access using Queues [Shah, Shin and Prasad] IEEE Symposium on Foundations of Computer Science 2011
* Randomized Algorithms for Queueing Networks [Shah and Shin] Annals of Applied Probability 2012
* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009

Network Adiabatic Theorem

- Theorem [Shah and S. 2009, 2010, 2011]*

  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

* Medium Access using Queues [Shah, Shin and Prasad] IEEE Symposium on Foundations of Computer Science 2011
* Randomized Algorithms for Queueing Networks [Shah and Shin] Annals of Applied Probability 2012
* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009

Network Adiabatic Theorem

- Theorem [Shah and S. 2009, 2010, 2011]*

  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)

* Medium Access using Queues [Shah, Shin and Prasad] IEEE Symposium on Foundations of Computer Science 2011
* Randomized Algorithms for Queueing Networks [Shah and Shin] Annals of Applied Probability 2012
* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009

Network Adiabatic Theorem

- Theorem [Shah and S. 2009, 2010, 2011]*

  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 ?

* Medium Access using Queues [Shah, Shin and Prasad] IEEE Symposium on Foundations of Computer Science 2011
* Randomized Algorithms for Queueing Networks [Shah and Shin] Annals of Applied Probability 2012
* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009

Network Adiabatic Theorem

- Theorem [Shah and S. 2009, 2010, 2011]*

  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 ?

why not $f(x)=x$, $x^{1/2}$ or loglog x ?

* Medium Access using Queues [Shah, Shin and Prasad] IEEE Symposium on Foundations of Computer Science 2011
* Randomized Algorithms for Queueing Networks [Shah and Shin] Annals of Applied Probability 2012
* Network Adiabatic Theorem [Rajagopalan, Shah and Shin] ACM SIGMETRICS 2009

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

$$Q(t) \rightleftarrows A(t)$$

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

$$Q(t) \rightleftarrows A(t)$$

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

$$Q(t) \rightleftarrows A(t)$$

Want Q(t) is stable

- 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

$$Q(t) \rightleftarrows A(t)$$

Want Q(t) is stable

- Proof Strategy
  - Step I : A(t) is 'stable'

    This talk
  - 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

$$Q(t) \rightleftarrows A(t)$$

Want Q(t) is stable

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

This talk

- 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

$$Q(t) \rightleftarrows A(t)$$

Want Q(t) is stable

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

This talk

- 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

$$Q(t) \rightleftarrows A(t)$$

Want Q(t) is stable

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

This talk

- 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

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

$$Q(t) \rightleftarrows A(t)$$

Want Q(t) is stable

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

This talk

- 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

$$Q(t) \rightleftarrows A(t)$$

Want Q(t) is stable

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

This talk

- 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 $P(t)$

Next : Holds if f=log

Main issue :
P(t) is time-varying

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

Proof Strategy for Positive Recurrence

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 π(t) of P(t) if

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

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)

||

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 π(t) of P(t) if

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

$$\text{mixing speed of } P(t) = \text{spectral gap of } P(t)=P(Q(t)) = \frac{1}{f(Q(t))^n}$$

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

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

mixing speed of $P(t)$ = spectral gap of $P(t)=P(Q(t))$ = $\frac{1}{f(Q(t))^n}$

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

Each non-zero entry of $P(t)$ is $\frac{1}{f(Q(t))}$ 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

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

mixing speed of P(t) = spectral gap of $P(t)=P(Q(t))$ = $\frac{1}{f(Q(t))^n}$

changing speed of P(t) = changing speed of its entry $\frac{1}{f(Q(t))}$ = $\frac{d}{dt}\frac{1}{f(Q(t))}$

Each non-zero entry of P(t) is $\frac{1}{f(Q(t))}$ 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

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

$$\text{mixing speed of } P(t) = \text{spectral gap of } P(t)=P(Q(t)) = \frac{1}{f(Q(t))^n}$$

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

$$\text{changing speed of } P(t) = \text{changing speed of its entry } \frac{1}{f(Q(t))} = \frac{d}{dt}\frac{1}{f(Q(t))} = \frac{f'(Q(t))}{f(Q(t))^2}\frac{dQ(t)}{dt}$$

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)

$$\text{mixing speed of } P(t) = \text{spectral gap of } P(t)=P(Q(t)) = \frac{1}{f(Q(t))^n}$$

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

$$\text{changing speed of } P(t) = \text{changing speed of its entry } \frac{1}{f(Q(t))} = \frac{d}{dt}\frac{1}{f(Q(t))} \overset{\text{chain rule}}{=} \frac{f'(Q(t))}{f(Q(t))^2}\,\boxed{\frac{d\,Q(t)}{dt} = 1}$$

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)

mixing speed of P(t) = spectral gap of P(t)=P(Q(t)) = $\frac{1}{f(Q(t))^n}$

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

changing speed of P(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

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)

$$\text{mixing speed of } P(t) = \text{spectral gap of } P(t)=P(Q(t)) = \frac{1}{f(Q(t))^n} = \frac{1}{Q(t)^n} \not> \frac{1}{Q(t)^2}$$

$$\text{changing speed of } P(t) = \text{changing speed of its entry } \frac{1}{f(Q(t))} = \frac{d}{dt}\frac{1}{f(Q(t))} = \frac{f'(Q(t))}{f(Q(t))^2} = \frac{1}{Q(t)^2}$$

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

Choose f(x)=x

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

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

mixing speed of $P(t)$ = spectral gap of $P(t)=P(Q(t))$ = $\dfrac{1}{f(Q(t))^n}$

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

changing speed of $P(t)$ = changing speed of its entry $\dfrac{1}{f(Q(t))}$ = $\dfrac{d}{dt}\dfrac{1}{f(Q(t))}$ = $\dfrac{f'(Q(t))}{f(Q(t))^2}$

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)

$$\text{mixing speed of } P(t) = \text{spectral gap of } P(t)=P(Q(t)) = \frac{1}{f(Q(t))^n} = \frac{1}{(\log Q(t))^n}$$

$$\text{changing speed of } P(t) = \text{changing speed of its entry } \frac{1}{f(Q(t))} = \frac{d}{dt}\frac{1}{f(Q(t))} = \frac{f'(Q(t))}{f(Q(t))^2}$$

$$\frac{1}{(\log Q(t))^n} > \frac{1}{Q(t)(\log Q(t))^2} = \frac{f'(Q(t))}{f(Q(t))^2}$$

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

Choose f(x)=log x

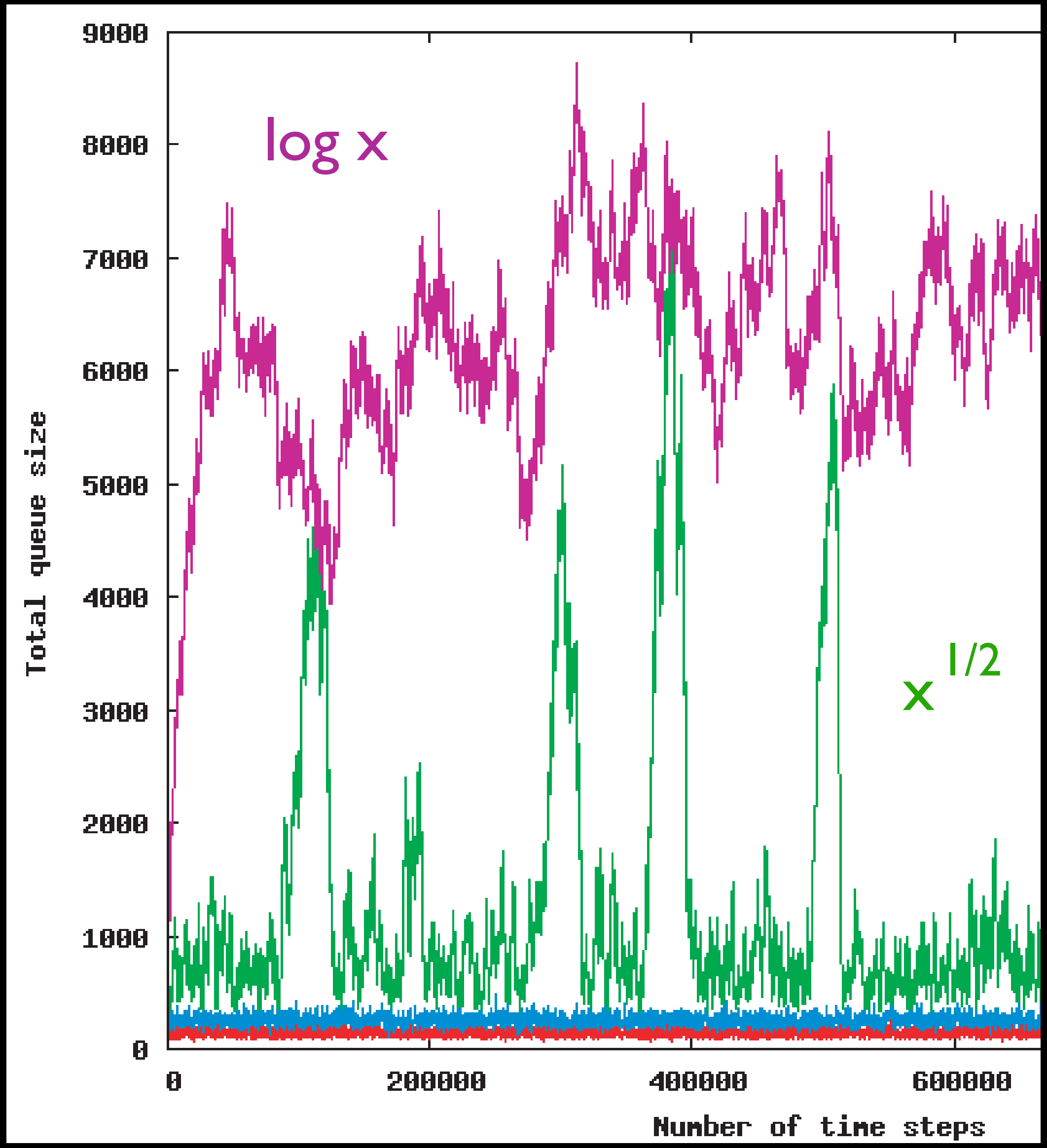
chain rule

Which function is best for throughput and delay?

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

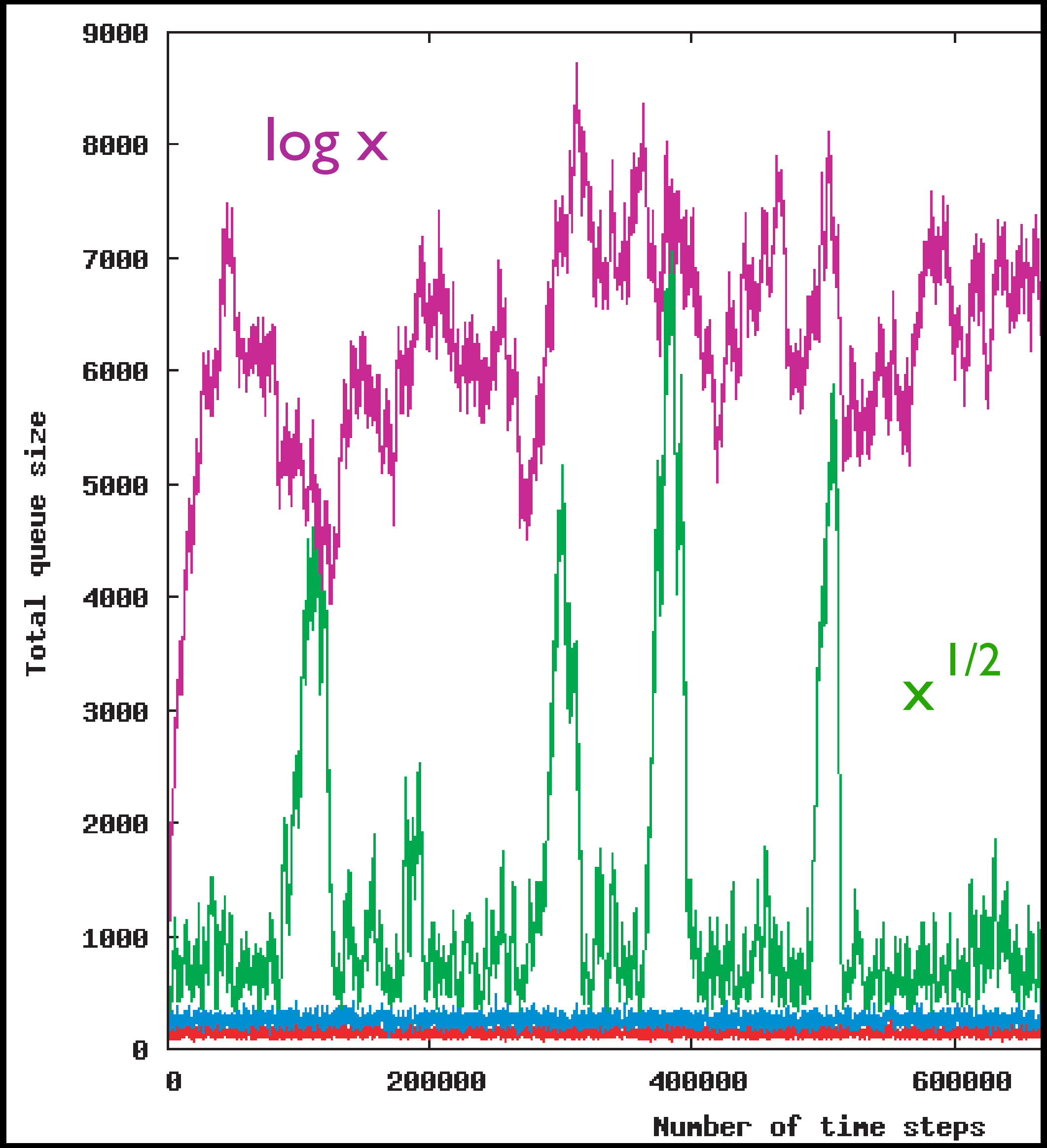
Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than x^{ε} e.g. $\log \mathbf{x}, \log\log \mathbf{x}, e^{\sqrt{\log \mathbf{x}}}$ ...
- Simulation on grid interference graph



Which function is best for throughput and delay?

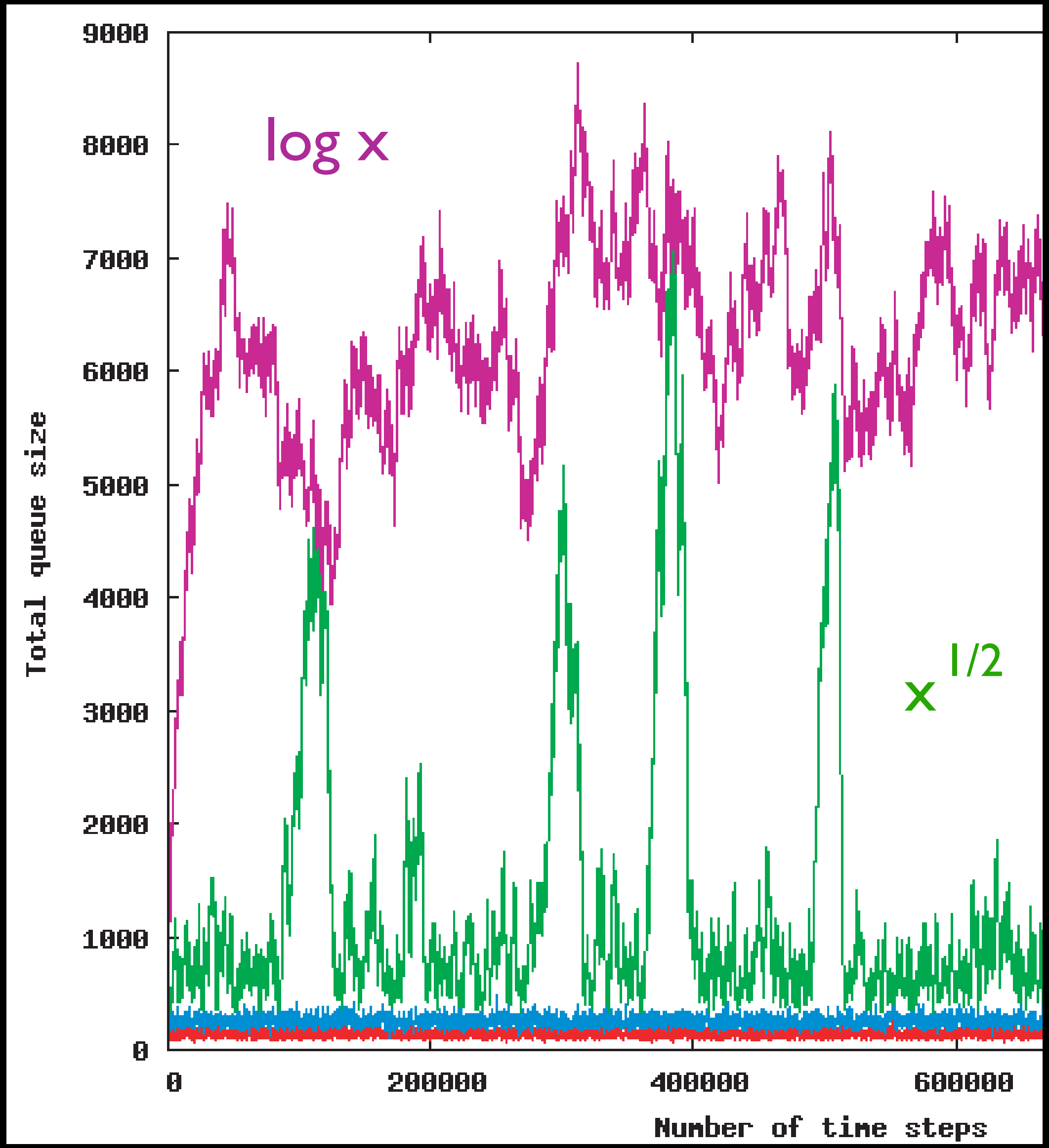
- Any sub-poly function works for throughput-optimality
  - Growing slower than x^{ε} e.g. $\log \mathbf{x}, \log\log \mathbf{x}, \mathrm{e}^{\sqrt{\log \mathbf{x}}}$ ...
- Simulation on grid interference graph



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

Which function is best for throughput and delay?

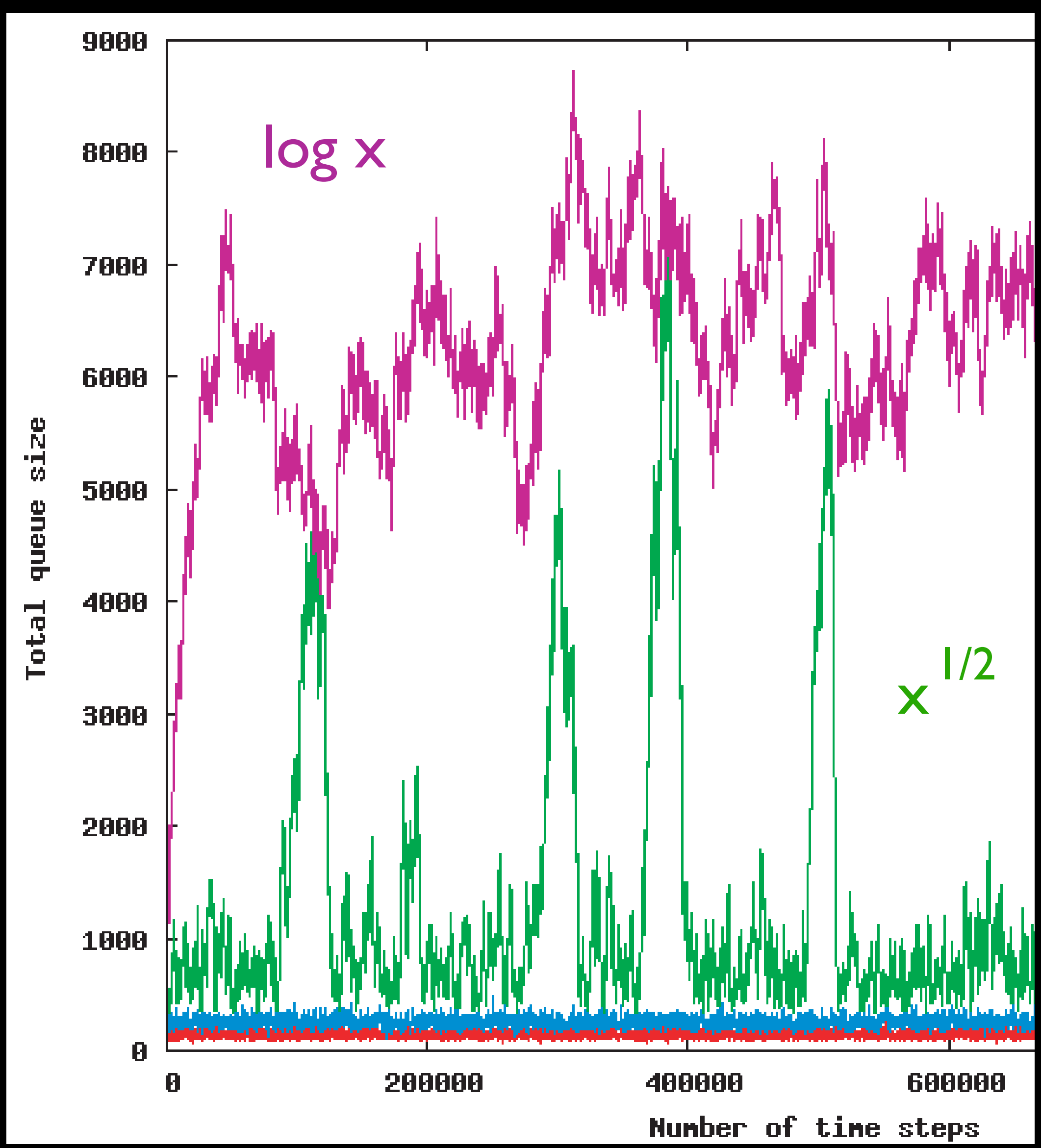
- Any sub-poly function works for throughput-optimality
  - Growing slower than x^{ε} e.g. $\log \mathbf{x}, \log\log \mathbf{x}, \mathrm{e}^{\sqrt{\log \mathbf{x}}}$ ...
- Simulation on grid interference graph



- 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.

Which function is best for throughput and delay?

- Any sub-poly function works for throughput-optimality
  - Growing slower than x^{ε} e.g. $\log \mathbf{x}, \log\log \mathbf{x}, e^{\sqrt{\log \mathbf{x}}}$ ...
- Simulation on grid interference graph

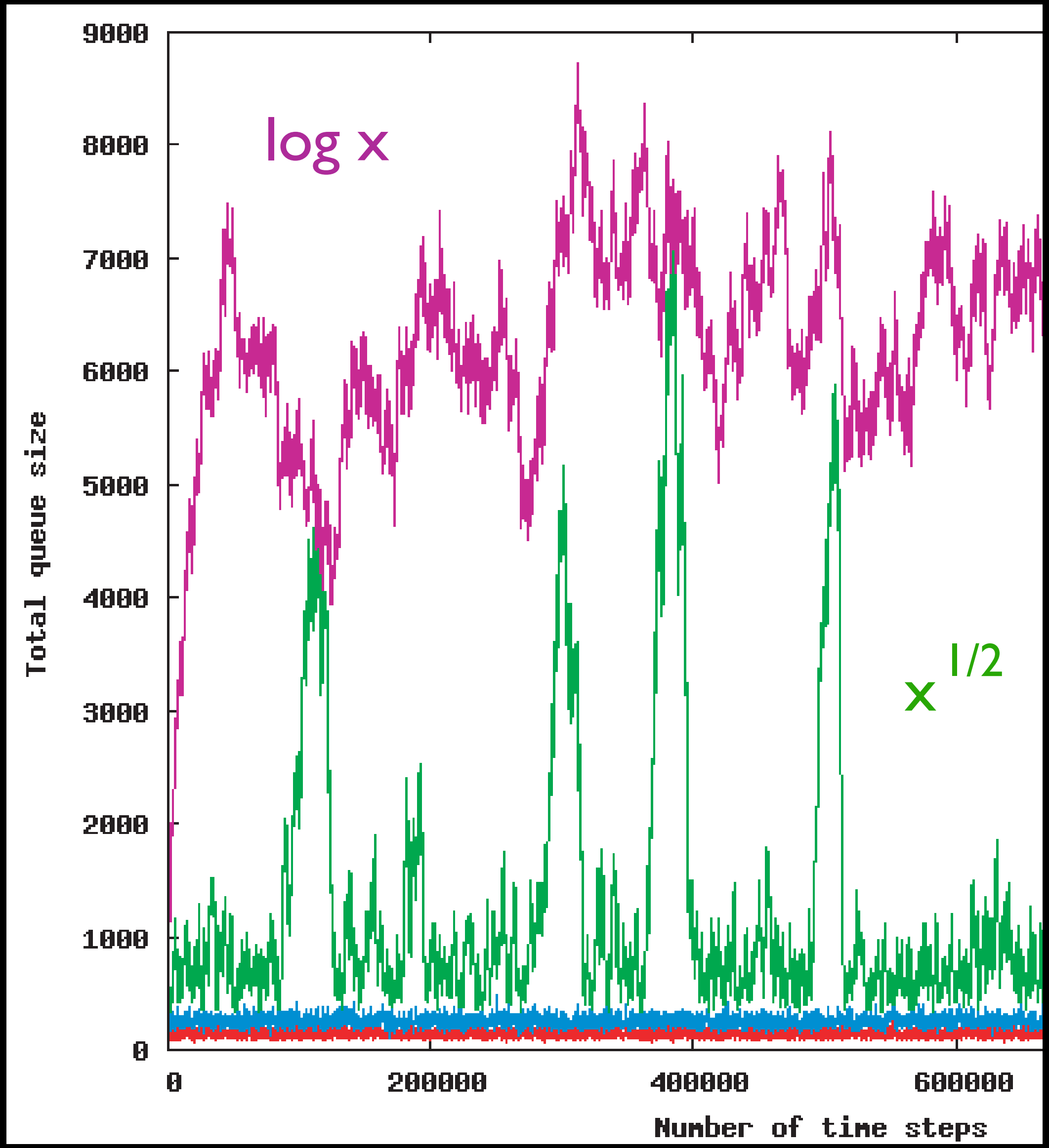


- Tradeoff : Faster growing function is better for queue-size but worse for stability
- Hence, the best function is the fastest growing one as long as it guarantees stability i.e.

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

Which function is best for throughput and delay?

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

$$\text{Queue-size} \approx \frac{1}{\text{Spectral gap of Glauber dynamics}}$$ 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.

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^{ε} e.g. $\log \mathbf{x}, \log\log \mathbf{x}, \mathbf{e}^{\sqrt{\log \mathbf{x}}}$ ...

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

- 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

$\downarrow$ Step II

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

$\downarrow$ Step I

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

Summary of Network Adiabatic Theorem

- In summary,

Designing a high performance medium access algorithm

↓ Step II

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

↓ Step I

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

Distributed Scheduling $\xRightarrow{\text{Step II}}$ Distributed (time-varying) Sampling $\xRightarrow{\text{Step I}}$ Time-varying Metropolis-Hastings Algorithm

Summary of Network Adiabatic Theorem

- In summary,

Designing a high performance medium access algorithm

↓ Step II

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

↓ Step I

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

Distributed Scheduling ⇒ (Step II) Distributed (time-varying) Sampling ⇒ (Step I) Time-varying Metropolis-Hastings Algorithm

- 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

$\downarrow$ Step II

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

$\downarrow$ Step I

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

↓ Step II

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

↓ 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

* 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

↓ Step II

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

↓ 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

↓ Step II

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

↓ 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 Exhaustic 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

$\downarrow$ Step II

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

$\downarrow$ Step I

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

Step II

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

Step I

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

Step II

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

Step I

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

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

My Contribution for Distributed Scheduling

Queue-based Algorithms

Network adiabatic theorem for medium access
[Shah and S.] SIGMETRICS 2009 and Annals of Applied Probability 2012

Generalized network adaibatic 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
[Shah and S.] SIGMETRICS 2009 and Annals of Applied Probability 2012

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

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

Local stability is enough for global stability
for multi-hop queueing networks
[Dieker and S.] Mathematics of Operations Research 2013

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
[Shah and S.] SIGMETRICS 2009 and Annals of Applied Probability 2012

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

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

Local stability is enough for global stability
for multi-hop queueing networks
[Dieker and S.] Mathematics of Operations Research 2013

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

Throughput optimality for rate-based CSMA
[Jiang, Shah, S. and Walrand] IEEE Transactions
on Information Theory 2010

Belief Propagation for medium access
[Yun, S. and Yi] to appear in IEEE
Transactions on Information Theory

Rate-based Algorithms

- Part II -
Message Passing in Statistical Networks

- Convergence and Correctness of Belief Propagation
  Joint work with Sungsoo Ahn (KAIST), Michael Cherktov (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

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

- Factor Graph for GM

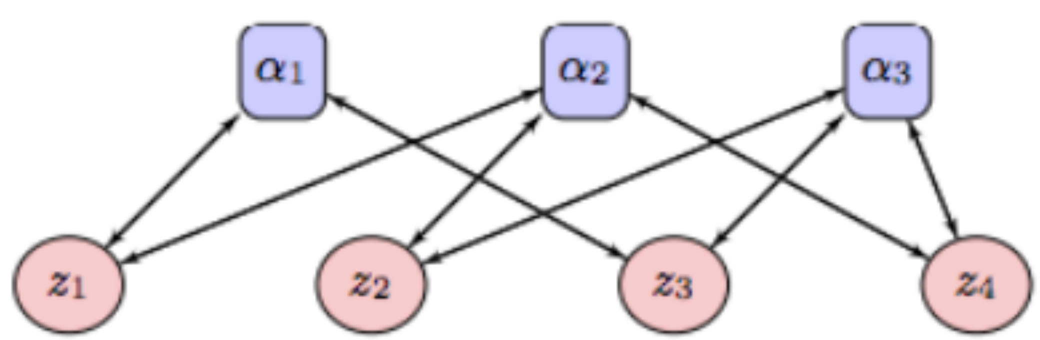


Figure 1: Factor graph for the graphical model $\Pr[z] \propto \psi_{\alpha_1}(z_1, z_3)\psi_{\alpha_2}(z_1, z_2, z_4)\psi_{\alpha_3}(z_2, z_3, z_4)$, i.e., $F = \{\alpha_1, \alpha_2, \alpha_3\}$ and $n = 4$. Each α_j selects a subset of z. For example, α_1 selects $\{z_1, z_3\}$.

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 $z = [z_i] \in \Omega^n$,

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

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

$$F = \{\alpha_1, \alpha_2, \dots, \alpha_k\} \subset 2^{\{1,2,\dots,n\}}$$

- 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

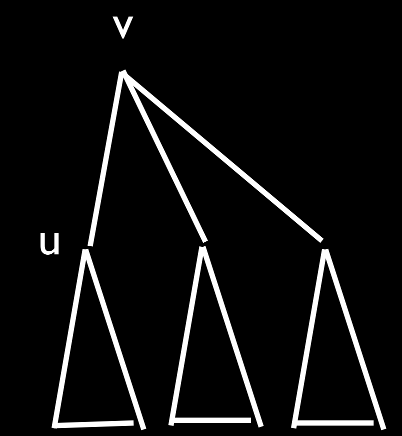
Need some heuristics or approximation algorithms !

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

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

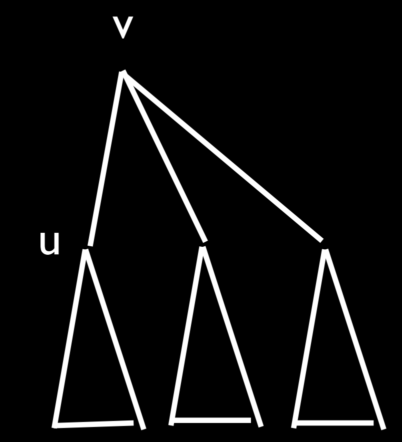


  - Goal : Compute marginal probabilities $p[X_v=1]$ for all v

Belief Propagation for Marginal Probability

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

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

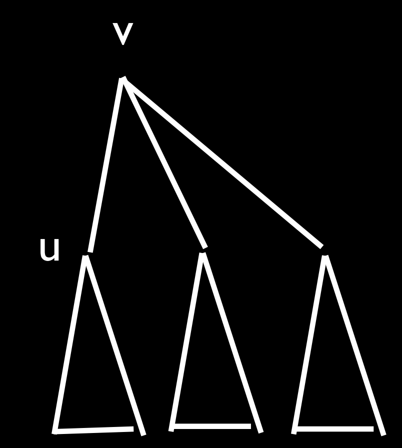


  - Goal : Compute marginal probabilities $p[X_v=1]$ for all v
  - (Divide & Conquer) Solve similar problems in sub-trees i.e.

Belief Propagation for Marginal Probability

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

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



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

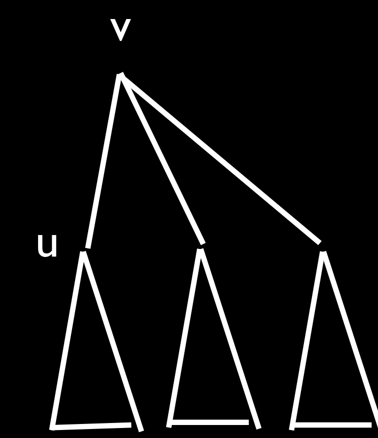
  - (Divide & Conquer) Solve similar problems in sub-trees i.e.

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

Belief Propagation for Marginal Probability

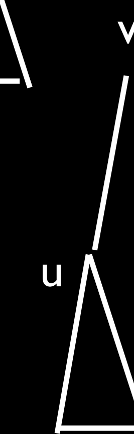
- Consider a random variable $X=[X_v]\in\{0,1\}^n$ and a tree graph G

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



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

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

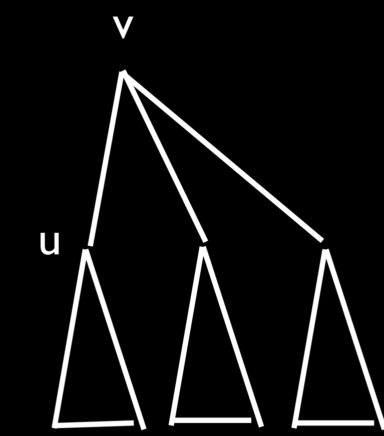


$$\frac{p[X_v=1]}{p[X_v=0]} = \prod_{u\in N(v)} M_{u\to v}$$

Belief Propagation for Marginal Probability

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

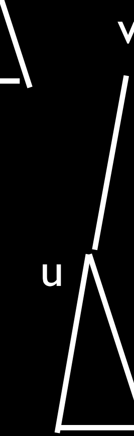
$$p[X] = \frac{1}{Z}\prod_{(u,v)\in E} 1-X_uX_v$$



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

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

$M_{u\to v}$ ⟵ Marginal ratio $\frac{p[X_v=1]}{p[X_v=0]}$ in sub-tree



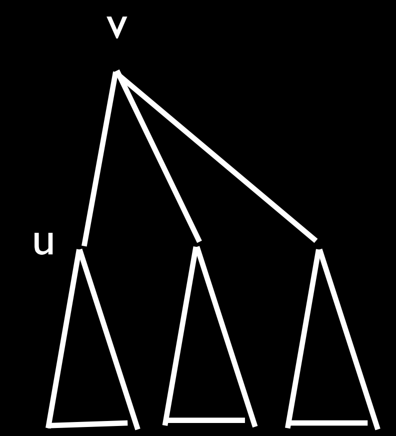
$$M_{u\to v} = \frac{1}{1+\prod_{w\in N(u)/v} M_{w\to u}}$$

$M_{w\to u}$ ⟵ 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

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

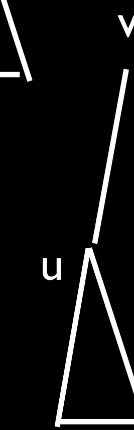


  - Goal : Compute marginal probabilities $p[X_v=1]$ for all v
  - (Divide & Conquer) Solve similar problems in sub-trees i.e.

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

$$\boxed{M_{u\to v} = \frac{1}{1 + \prod_{w\in N(u)/v} M_{w\to u}}}$$

$M_{u\to v}$ ⟵ Marginal ratio $\frac{p[X_v=1]}{p[X_v=0]}$ in sub-tree

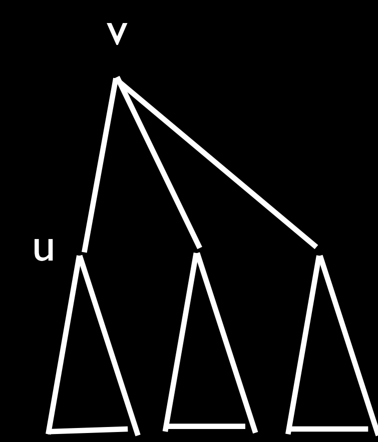


$\prod_{w\in N(u)/v} M_{w\to u}$ ⟵ Marginal ratio in sub-sub-tree

Belief Propagation for Marginal Probability

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

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

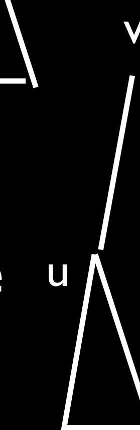


  - Goal : Compute marginal probabilities $p[X_v=1]$ for all v
  - (Divide & Conquer) Solve similar problems in sub-trees i.e.

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

$$\boxed{M_{u\to v} = \frac{1}{1+\prod_{w\in N(u)/v} M_{w\to u}}}$$

$M_{u\to v}$ ← Marginal ratio $\frac{p[X_v=1]}{p[X_v=0]}$ in sub-tree

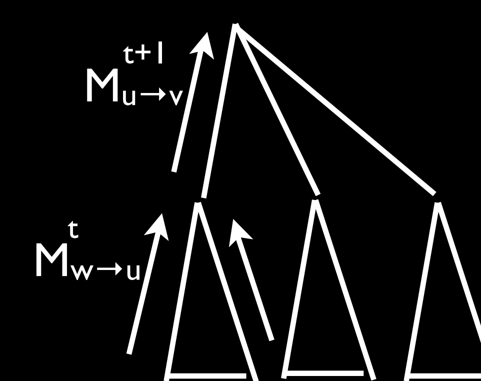


$\prod_{w\in N(u)/v} M_{w\to u}$ ← Marginal ratio in sub-sub-tree

- Belief Propagation algorithm :

$$M^{t+1}_{u\to v} = \frac{1}{1+\prod_{w\in N(u)/v} M^{t}_{w\to u}}$$

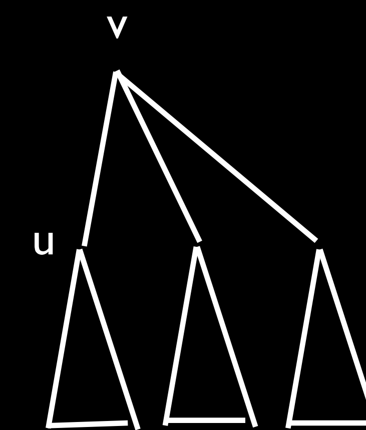
$(M^{0}_{u\to v} = 1/2)$



Belief Propagation for Marginal Probability

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

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

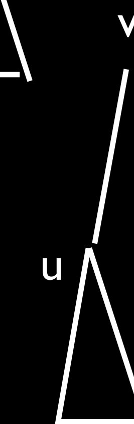


  - Goal : Compute marginal probabilities $p[X_v=1]$ for all v
  - (Divide & Conquer) Solve similar problems in sub-trees i.e.

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

$$M_{u\to v} = \frac{1}{1+\prod_{w\in N(u)/v} M_{w\to u}}$$

$M_{u\to v}$ ← Marginal ratio $\frac{p[X_v=1]}{p[X_v=0]}$ in sub-tree



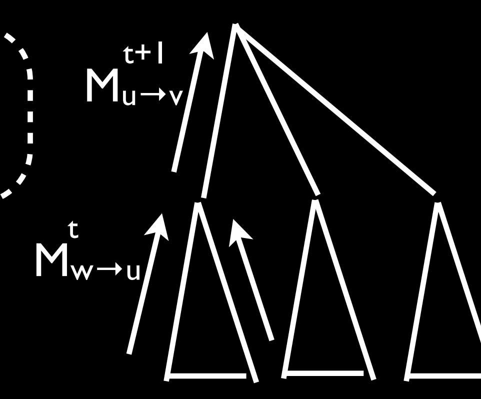
$\prod_{w\in N(u)/v} M_{w\to u}$ ← Marginal ratio in sub-sub-tree

- Belief Propagation algorithm :

$$M^{t+1}_{u\to v} = \frac{1}{1+\prod_{w\in N(u)/v} M^{t}_{w\to u}}$$

It is dynamic programming !

$(M^{0}_{u\to v} = 1/2)$



Belief Propagation (BP)

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

Why ?

- 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^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

- 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^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

- 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 Δ

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

* Complexity of Bethe Approximation [Shin] IEEE Transactions on Information Theory 2014

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 Δ

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

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

* Complexity of Bethe Approximation [Shin] IEEE Transactions on Information Theory 2014

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 Δ

  - 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 ?

* Complexity of Bethe Approximation [Shin] IEEE Transactions on Information Theory 2014

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 Δ

- 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 ?

* Complexity of Bethe Approximation [Shin] IEEE Transactions on Information Theory 2014

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 Δ

- 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 ?

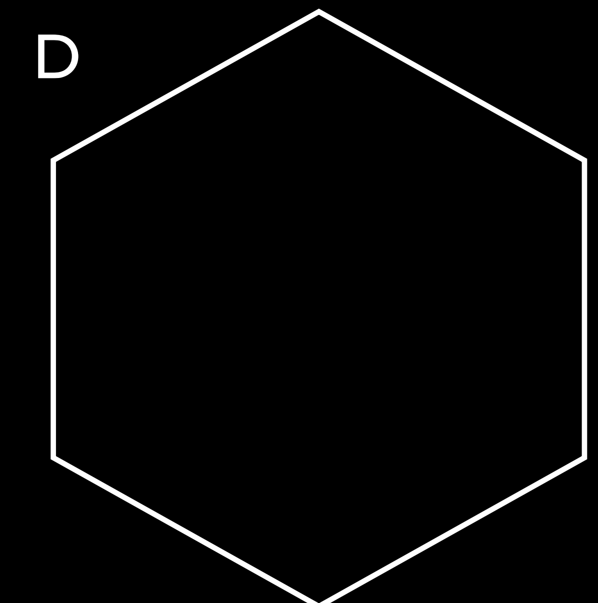
[S. et al. 2011]† Fixed points of f_{BP} are good if girth = $\Omega(\log n)$

† Computing Indep. Sets using Bethe Approximation [Shin et al.] SIDMA 2011

* Complexity of Bethe Approximation [Shin] IEEE Transactions on Information Theory 2014

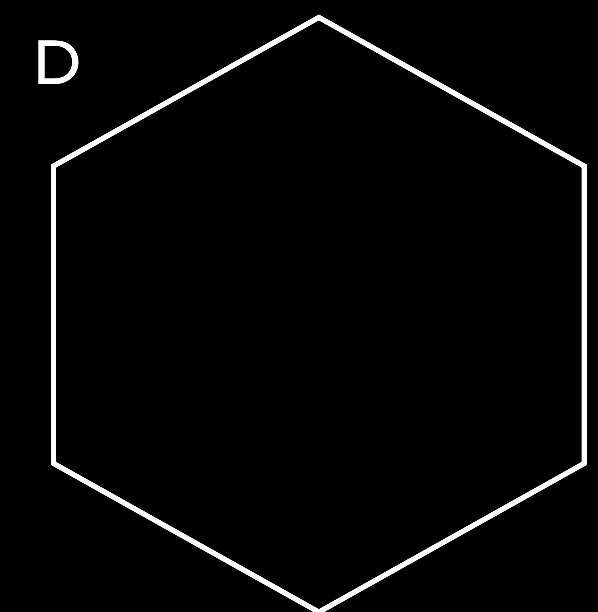
Proof Strategy : How to find a fixed point of SBP

- Equivalent to find a zero-gradient point of F_{Bethe} [Yedidia et al. 04]
  - $F_{Bethe} : D \rightarrow R$ 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



Proof Strategy : How to find a fixed point of SBP

- Equivalent to find a zero-gradient point of F_{Bethe} [Yedidia et al. 04]
  - $F_{Bethe} : D \to R$ 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

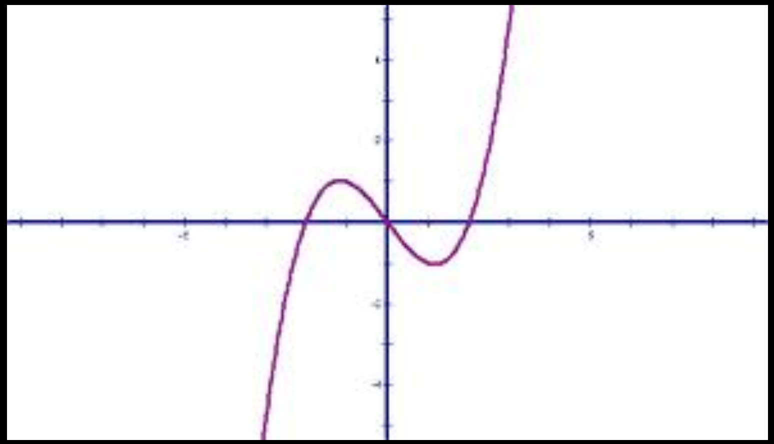
- Natural attempt : Gradient-descent algorithm

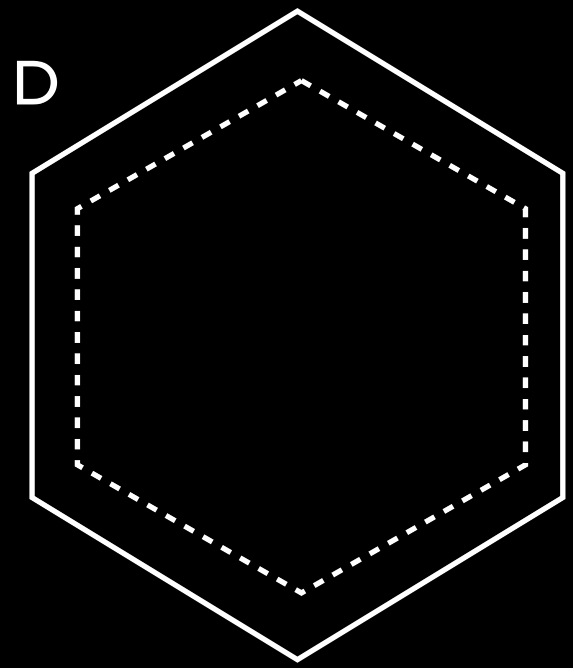
$$x^{t+1} = x^t + \alpha \nabla F_{Bethe}(x^t)$$ for some (step-size) $\alpha>0$

  - Does it find to a zero-gradient point? If not, why?

Proof Strategy : How to find a fixed point of LBP

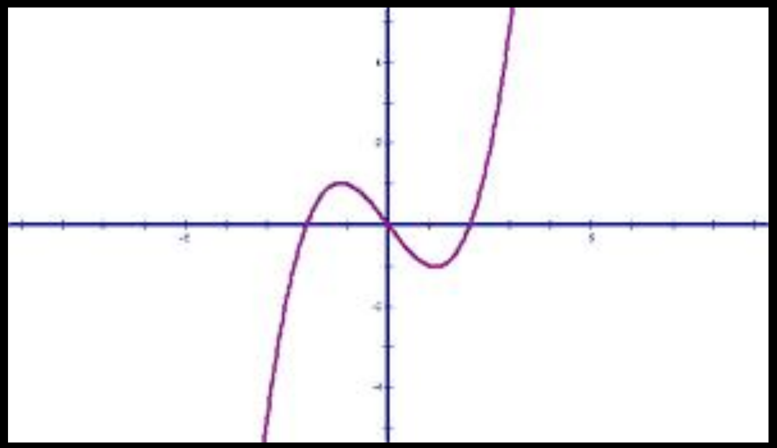
- When does gradient-descent algorithm work for general F ?
  - Its domain is unbounded and $|F|, |\nabla F|, |\nabla^2 F|$ are bounded $\rightarrow$ Possible to choose α

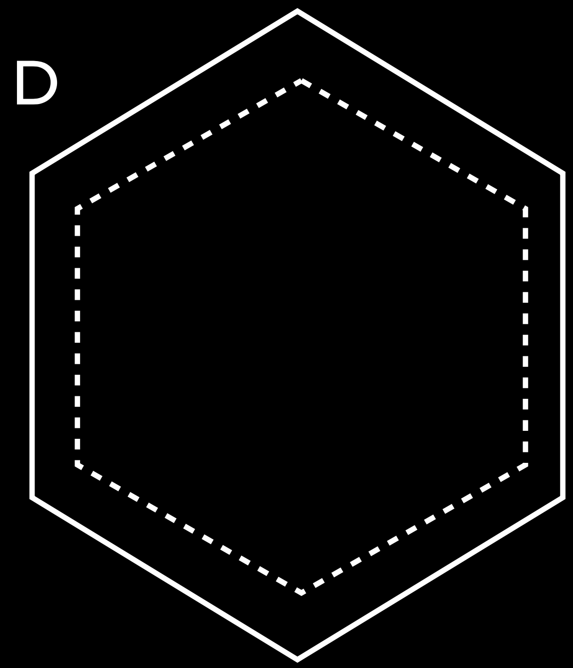




Proof Strategy : How to find a fixed point of LBP

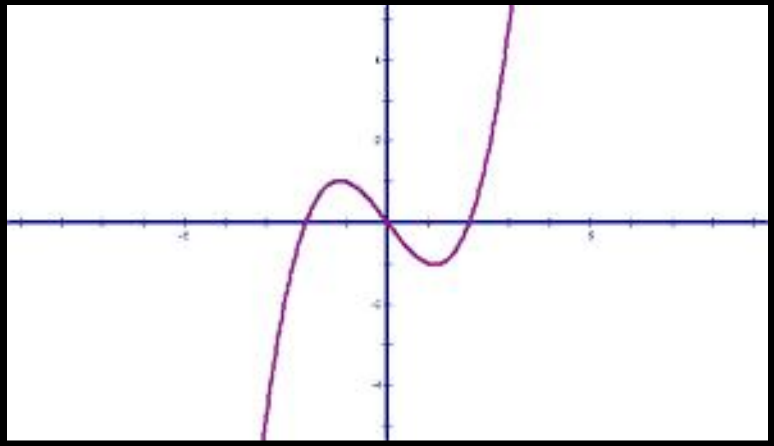
- When does gradient-descent algorithm work for general F ?
  - Its domain is unbounded and $|F|$, $|\nabla F|$, $|\nabla^2 F|$ are bounded → Possible to choose α
- Two issues for $x^{t+1} = x^t + \alpha \nabla F_{Bethe}(x^t)$
  - Domain D is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of D)





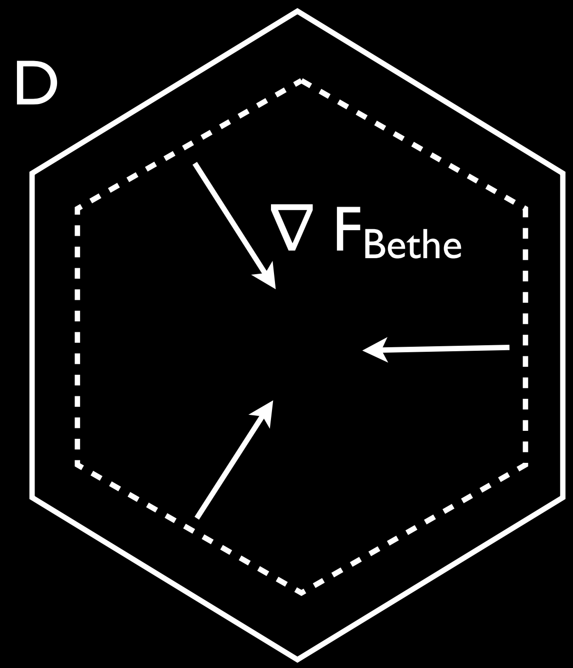
Proof Strategy : How to find a fixed point of LBP

- 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 α
- Two issues for $x^{t+1} = x^t + \alpha \nabla F_{Bethe}(x^t)$



  - 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

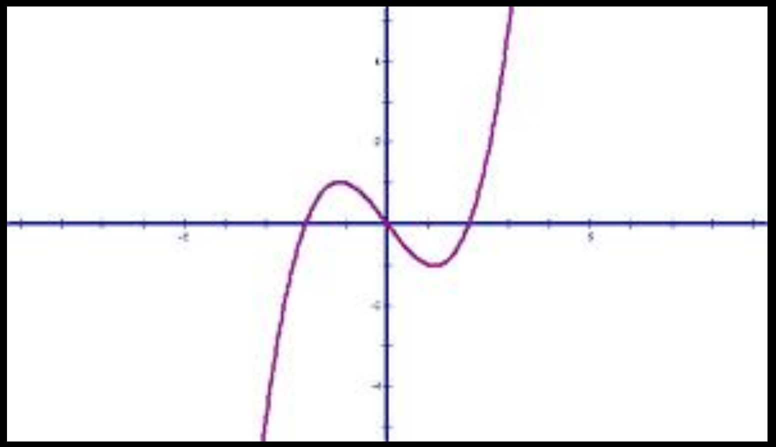
Suppose ∇F_{Bethe} always points "inside" close to boundary of D



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 α

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

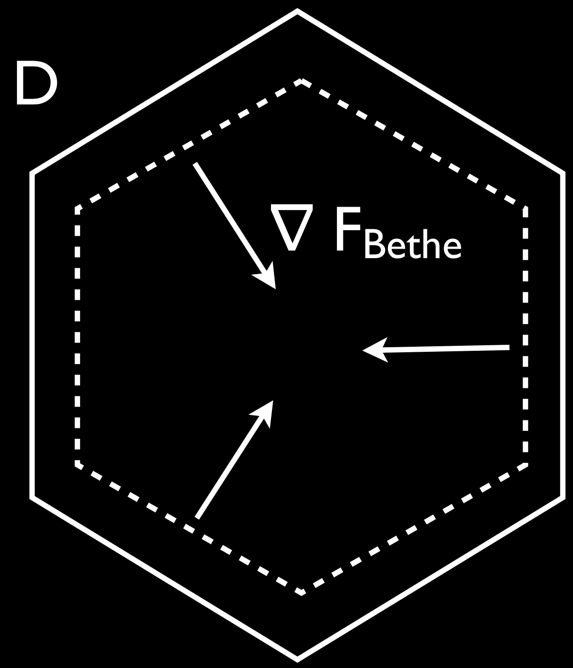


- Our main idea to resolve the issues

  Suppose ∇F_{Bethe} always points "inside" close to boundary of D

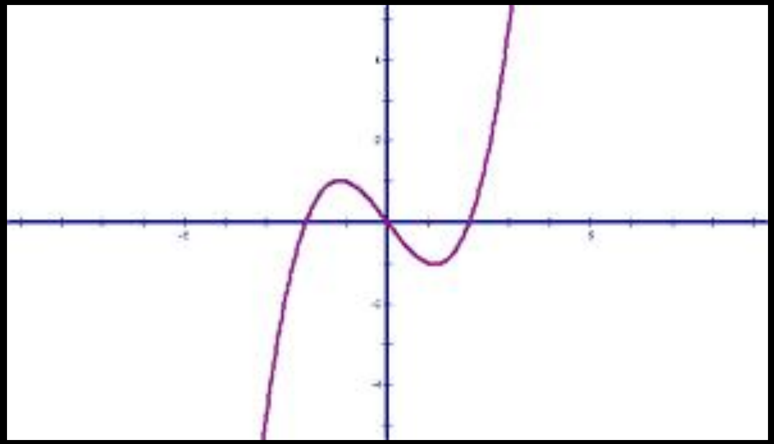
  - Then, it is possible choose small α so that

    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 → Possible to choose α



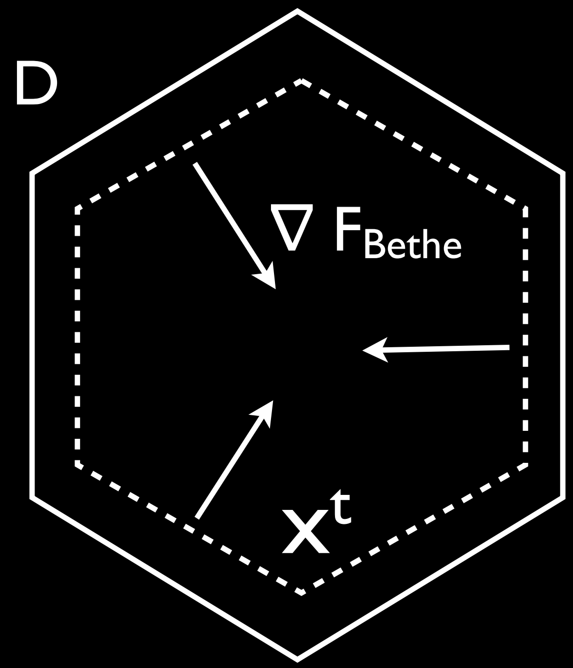
- Two issues for $x^{t+1} = x^t + \alpha \nabla F_{Bethe}(x^t)$
  - Domain D is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of D)

- Our main idea to resolve the issues

  Suppose ∇F_{Bethe} always points "inside" close to boundary of D

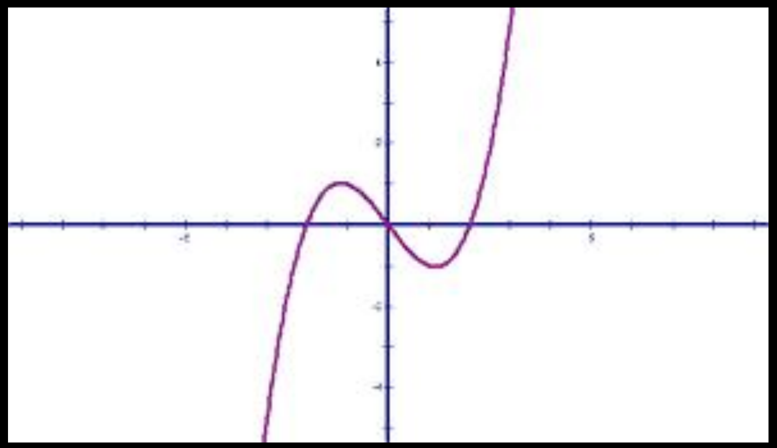
  - Then, it is possible choose small α so that

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 → Possible to choose α



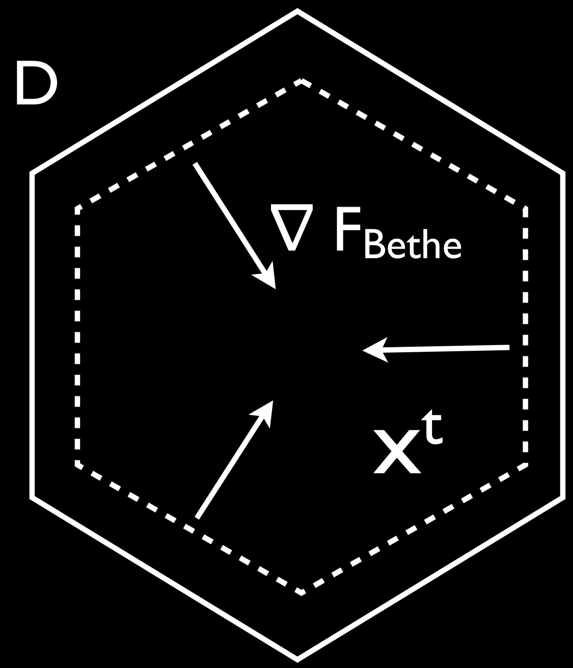
- Two issues for $x^{t+1} = x^t + \alpha \nabla F_{Bethe}(x^t)$
  - Domain D is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of D)

- Our main idea to resolve the issues

  Suppose ∇F_{Bethe} always points “inside” close to boundary of D

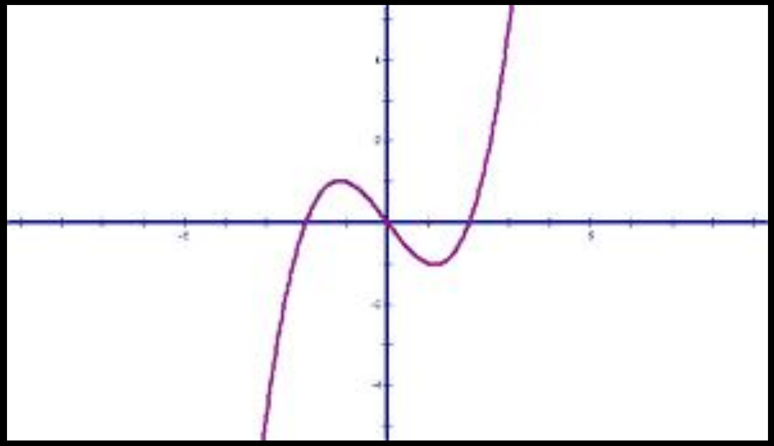
  - Then, it is possible choose small α so that

    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 → Possible to choose α



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

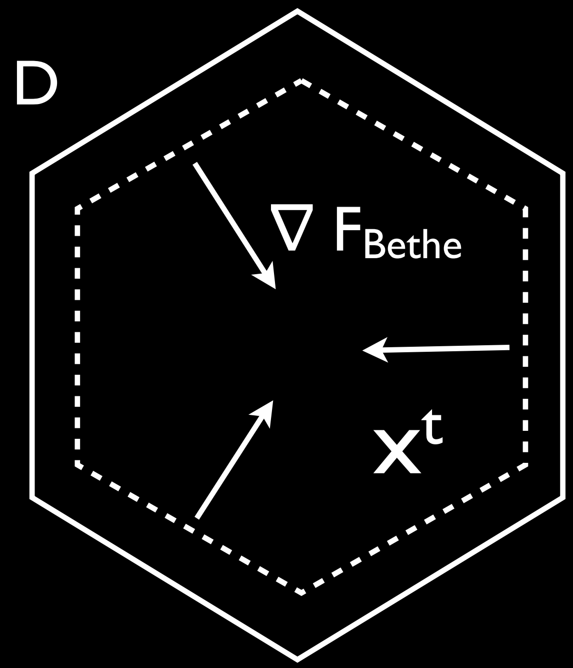
- Our main idea to resolve the issues

  Suppose ∇F_{Bethe} always points "inside" close to boundary of D

  - Then, it is possible choose small α so that

    x^t is always far from boundary D i.e. $x^t \in D^* \subset D$

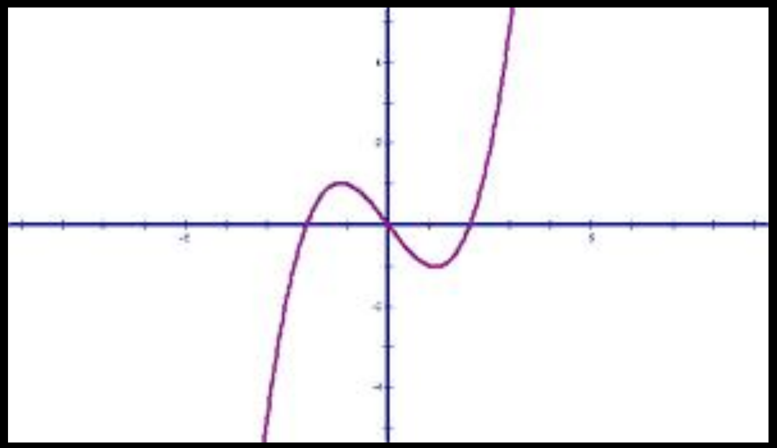
  - Unfortunately, this (or similar) seems not true ... but



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 α

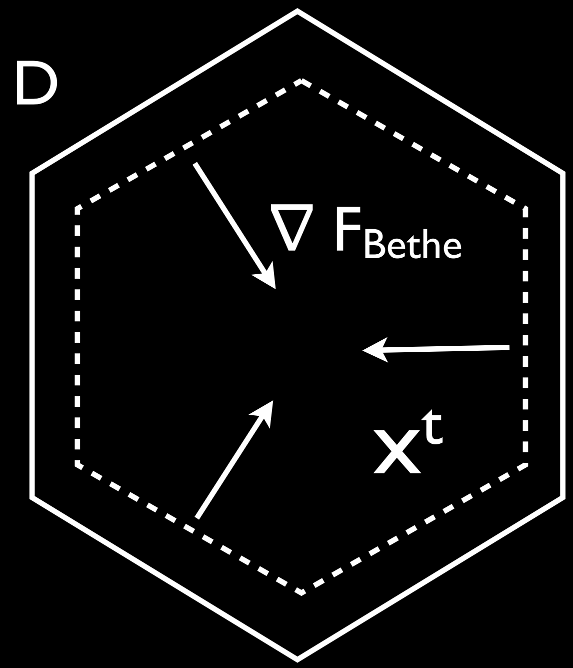
- Two issues for $x^{t+1} = x^t + \alpha \nabla F_{Bethe}(x^t)$
  - Domain D is bounded i.e. a projection may be required
  - Derivatives are unbounded (close to boundary of D)



- Our main idea to resolve the issues

  Suppose ∇F_{Bethe} always points "inside" close to boundary of D
  - Then, it is possible choose small α 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_{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 !
  - For example, Bipartite Matchings [Bayati, Shah and Sharma 2006], Matchings [Sanghavi, Malioutov and Willsky 2007] Perfect Matchings [Bayati, Borgs, Chayes and Zecchina 2008] Shortest Path [Ruozzi and Tatikonda 2008] Independent Sets [Sanghavi, Shah and Willsky 2008] Min-cost Network Flow [Gamarnik, Shah and Wei 2011] Matchings with Odd Cycles [S., Chertkov and Gelfand 2013]*

- 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, 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 !
  - For example, Bipartite Matchings [Bayati, Shah and Sharma 2006], Matchings [Sanghavi, Malioutov and Willsky 2007] Perfect Matchings [Bayati, Borgs, Chayes and Zecchina 2008] Shortest Path [Ruozzi and Tatikonda 2008] Independent Sets [Sanghavi, Shah and Willsky 2008] Min-cost Network Flow [Gamarnik, Shah and Wei 2011] Matchings with Odd Cycles [S., Chertkov and Gelfand 2013]*
  - [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, 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 !
  - For example, Bipartite Matchings [Bayati, Shah and Sharma 2006], Matchings [Sanghavi, Malioutov and Willsky 2007] Perfect Matchings [Bayati, Borgs, Chayes and Zecchina 2008] Shortest Path [Ruozzi and Tatikonda 2008] Independent Sets [Sanghavi, Shah and Willsky 2008] Min-cost Network Flow [Gamarnik, Shah and Wei 2011] Matchings with Odd Cycles [S., Chertkov and Gelfand 2013]*
  - [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

† 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 ψ_α, 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*

$$|\{j \in \{i\} \cup \gamma : |F_j| = 2\}| \leq 2$$

$$\psi_\alpha(x'_\alpha) = 1, \qquad \textit{where } x'_k = \begin{cases} x_k & \textit{if } k \notin \{i\} \cup \gamma \\ x_k^* & \textit{otherwise} \end{cases}.$$

$$\psi_\alpha(x''_\alpha) = 1, \qquad \textit{where } x''_k = \begin{cases} x_k & \textit{if } k \in \{i\} \cup \gamma \\ x_k^* & \textit{otherwise} \end{cases}.$$

* 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 ψ_α, 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*

$$|\{j \in \{i\} \cup \gamma : |F_j| = 2\}| \leq 2$$

$$\psi_\alpha(x'_\alpha) = 1, \qquad \text{where } x'_k = \begin{cases} x_k & \text{if } k \notin \{i\} \cup \gamma \\ x^*_k & \text{otherwise} \end{cases}.$$

$$\psi_\alpha(x''_\alpha) = 1, \qquad \text{where } x''_k = \begin{cases} x_k & \text{if } k \in \{i\} \cup \gamma \\ x^*_k & \text{otherwise} \end{cases}.$$

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

* 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 ψ_α, 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*

$$|\{j \in \{i\} \cup \gamma : |F_j| = 2\}| \leq 2$$

$$\psi_\alpha(x'_\alpha) = 1, \qquad \text{where } x'_k = \begin{cases} x_k & \text{if } k \notin \{i\} \cup \gamma \\ x_k^* & \text{otherwise} \end{cases}.$$

$$\psi_\alpha(x''_\alpha) = 1, \qquad \text{where } x''_k = \begin{cases} x_k & \text{if } k \in \{i\} \cup \gamma \\ x_k^* & \text{otherwise} \end{cases}.$$

"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

* Max-product Belief Propagation for Linear Programming [Park and Shin] UAI 2015

Examples of LP solvable by BP

Shortest Path

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

Minimum Weight (Perfect) Matching

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

Vertex Cover

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

Cycle Packing

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

Traveling Salesman Problem

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

Network Flow

$$\begin{array}{ll} \text{minimize} & w \cdot x \\ \text{subject to} & \displaystyle\sum_{e \in \delta^o(v)} x_e - \sum_{e \in \delta^i(v)} x_e = d_v, \ \forall v \in V \\ & x = [x_e] \in \mathbb{R}_+^{|E|} \end{array}$$

My Contribution for Belief Propagation

- Sum-product BP
  - Polynomial-time algorithm for computing a BP fixed-point
    [S.] IEEE Transactions on Information Theory 2014
  - Large-girth condition for correctness of BP
    [S. et al.] SIAM Journal on Discrete Mathematics 2011
- 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

Part II

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

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