Message Passing Algorithms for Network Scheduling

Devavrat Shah

Laboratory of Information and Decision Systems
Dept. of Electrical Engg. and Computer Sci.

MIT

Motivation

- Challenges for a network algorithm (system) designer
  - ○ Design algorithm that respect technological constraints, such as
    - – *low* memory-bandwidth
    - – *few* off-chip operations
    - – pipeline-able design
  - ○ Further, technology changes very fast
    - – memory speed *doubles* every 18 months
    - → can not design algorithms for a specific constraint
  - ○ Also, performance should be *excellent* so as to utilize resources well

Motivation

- In summary, we need algorithms that
  - ○ Respect changing technological constraints
    - – that are impossible to view at an abstract level
  - ○ And provide high-performance

→ This makes task of system designer extremely challenging

- We will take a pragmatic approach towards algorithm design
  - ○ First, consider algorithms that satisfy the technological constraint
  - ○ Then, optimize for performance

Motivation

- *Universal* algorithmic architecture
  - ○ Parameteric class of algorithms that can be implemented
    - – for any set of constraints by setting the parameters
  - ○ Performance adapts to the availability of resources
    - – with limited resources, provides relatively good performance
    - – with sufficient resources, provides *optimal* performance

  → Implementor can tune for performance at implementation-cost
- We will present such solutions for switch and wireless scheduling
  - ○ Based on powerful technique of Belief Propagation
  - ○ Distributed and iterative
    - – scalable and pipelineable
  - ○ Use of light-weight data-structure

Outline

- Max product belief propagation algorithm
  - ○ Some background
  - ○ Intuitive explanation
- Switch scheduling
  - ○ Background
  - ○ Algorithm
  - ○ Results
- Wireless scheduling
  - ○ Algorithm
  - ○ Results
- Discussion

Max-product Algorithm

- Background on Max-product
  - ○ It has roots in Artificial Intelligence and Statistical Physics
  - ○ Designed primarily for finding “mode” of distribution
    - – described by graphical models
  - ○ General heuristic

- Successful applications
  - ○ Decoding algorithm for codes based on graphs
    - – e.g. Low-density parity check codes
  - ○ Image processing: denoising image
  - ○ Discrete optimization problems

Max-product Algorithm

- General heuristic for discrete optimization problems
  - ○ Exact description needs elaborate notation
- I will present an intuitive description
  - ○ With limited notation
  - ○ Later, precise description for matching & independent set
- Key useful properties of algorithm
  - ○ Distributed
  - ○ Simple iterative operations
  - ○ Extremely light-weight data structure

Max-product Algorithm

- Some notation
  - ○ Variables $X_1, \ldots, X_n$ taking values in Σ (e.g. $\{0,1\}$)
  - ○ Cost
    - – $c_i : \Sigma \to \mathbb{R}_+$ for variable i
    - – total cost: $c(\bar{x}) = \sum_i c_i(x_i)$
  - ○ Constraints are pair-wise represented by a graph $G = (V, E)$
    - – $V = \{1, \ldots, n\}$, $E \subset V \times V$
    - – $(i, j) \in E$ implies constraint between X_i and X_j
    - – e.g. $(X_i, X_j) = (1, 1)$ is *not allowed*
  - ○ Optimization:

$$\text{maximize} \quad c(\bar{x}) = \sum_i c_i(x_i),$$

$$\text{subject to} \quad \bar{x} \text{ is feasible as per } G.$$

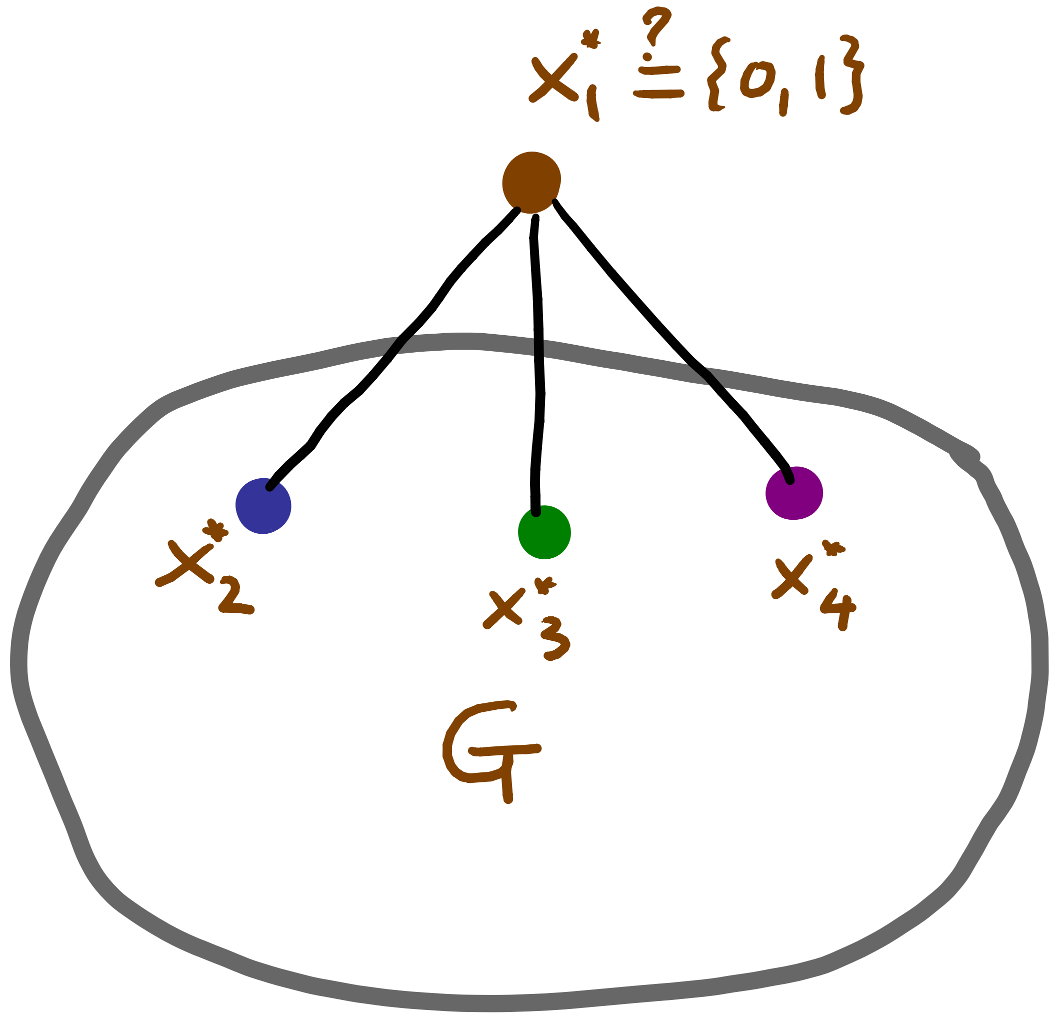
Max-product Algorithm

- Max-product is essentially parallel implementation of dynamic programming assuming G is a *tree*

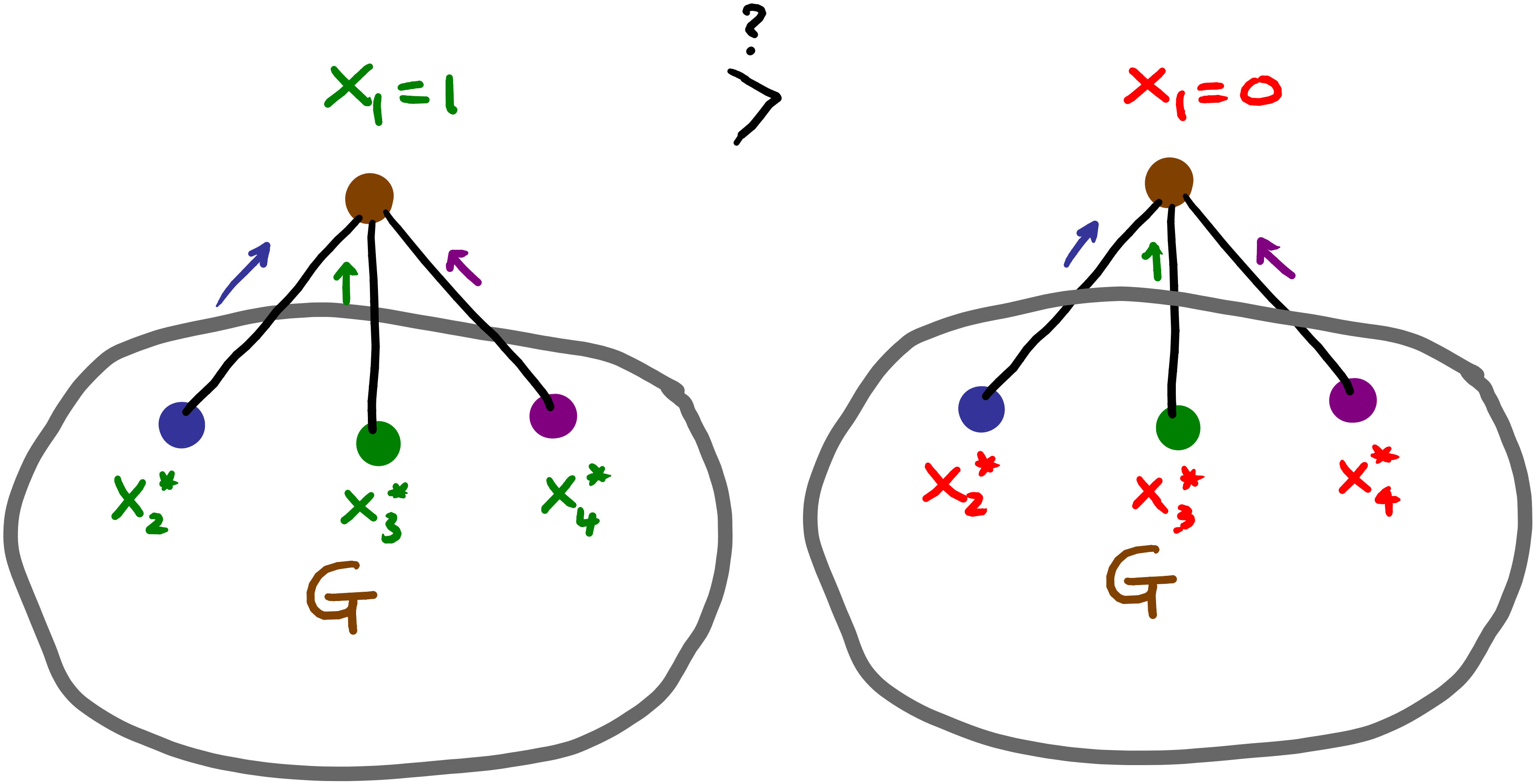
- First, let's recall key steps of dynamic programing

- In dynamic programing, to obtain optimal assignment
  - We recursively fix values of variables to one of the possible values in an arbitrary order
  - And, we compute cost of best solution given the fixed values of variables

Max-product Algorithm



Max-product Algorithm



Rest of the Talk

- Switch scheduling
  - ○ Background
  - ○ Algorithm
  - ○ Results

- Wireless scheduling
  - ○ Algorithm
  - ○ Results

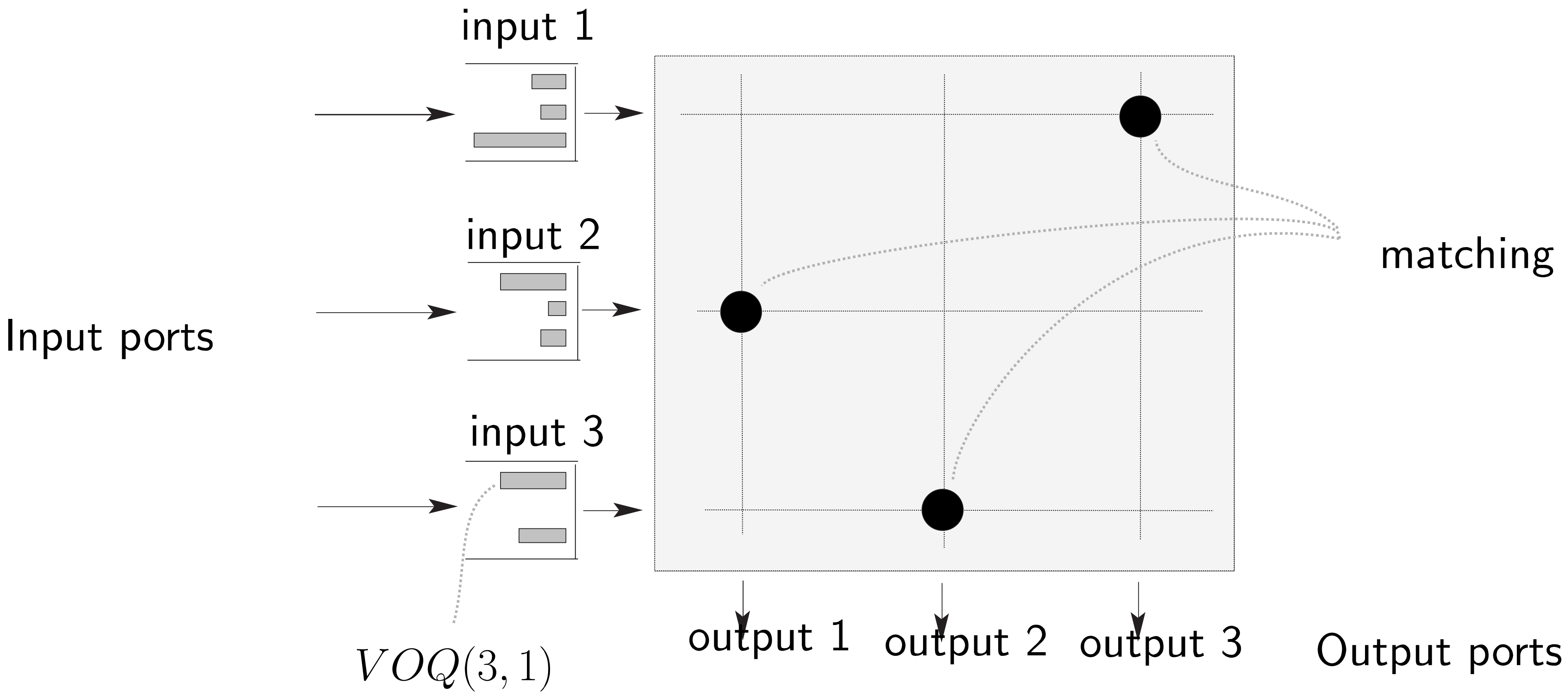
- Discussion

Switch Scheduling using Max-product

Mohsen Bayati Mayanak Sharma

Stanford IBM Research

Input-Queued Switch



- Scheduling constraint: at a given time
  - ○ Each input (output) can send (receive) at most one packet

→ Schedule is a matching in a complete bipartite graph between input and output ports

Scheduling Algorithm

- In an input-queued switch
  - ○ Scheduling algorithm is required to find a matching so that
    - – Overall network capacity is well-utilized, and
    - – Average delay is minimized

- Next, we will discuss
  - ○ Some notations
  - ○ A desirable scheduling algorithm, and
  - ○ Currently implemented algorithm

Notations

- Consider an n-port switch
    - ○ n inputs and n outputs
    - ○ n^2 queues at inputs: VOQ(i,j), $1 \leq i, j \leq n$

- Time is slotted and packets are of unit-size
    - ○ At most one packet arrives at an input in a given time-slot
    - ○ Let λ_{ij} be packet arrival rate for VOQ(i,j)
    - ○ $Q_{ij}(t)$ be number of packets in VOQ(i,j) at t

- $\lambda = [\lambda_{ij}]$ is admissible iff

$$\sum_k \lambda_{ik} < 1, \quad \sum_k \lambda_{kj} < 1, \quad \forall i, j.$$

Performance metric

- Throughput
  - An algorithm is called *stable* (or delivers 100% throughput), if
    - For any admissible λ, the average queue-size is finite

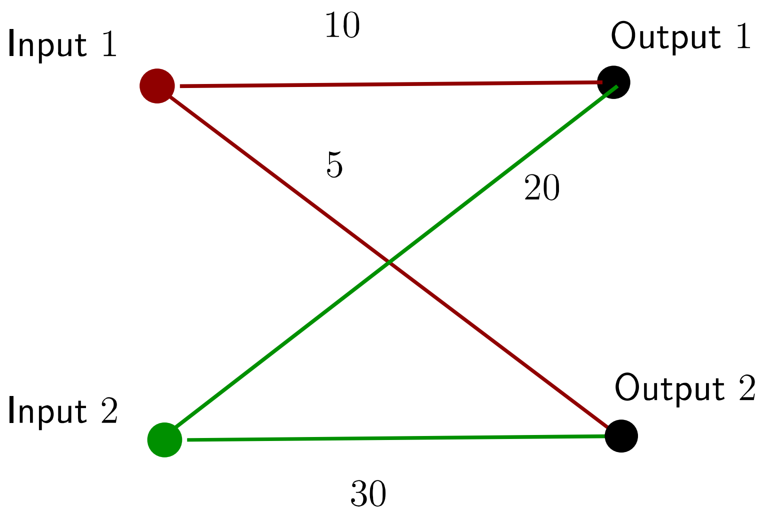
$$\sup_t \mathbb{E}\left[Q_{ij}(t)\right] < \infty, \quad \text{forall} \quad i, j.$$

- Net average queue-size:
  - $\sup_t \mathbb{E}\left[Q(t)\right]$,
    - where $Q(t) = \sum_{ij} Q_{ij}(t)$.

Maximum Weight Matching

- Consider weighted switch-bipartite graph
  - ○ Each edge is assigned weight equal to queue-size

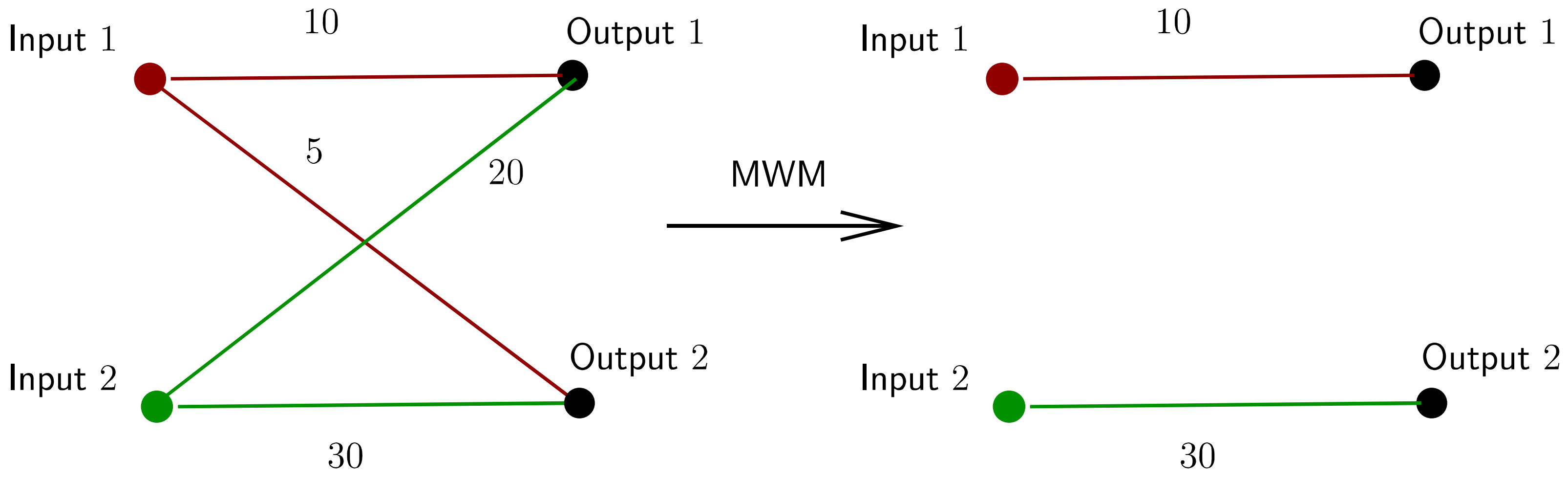
- An example of a 2-port switch



Maximum Weight Matching

- Algorithm: max. wt. matching (MWM)
  - ○ Every time, choose schedule (matching) with max. weight, and
  - ○ Transfer packets according to this schedule

- An example of a 2-port switch



Maximum Weight Matching

- The MWM algorithm has excellent performance
  - ○ It is stable
  - ○ The net average queue-size is $O(n^2)$ for admissible λ

- Standard network-flow style algorithm
  - ○ Takes $O(n^3)$ operations
  - ○ Requires complex data structure and not iterative

→ This makes it difficult to implement at very high speed

iSLIP Algorithm

- The iSLIP is a maximal matching algorithm
  - ○ Implemented in current routers (e.g. Cisco's GSR 12K)

- iSLIP Algorithm
  - ○ It is iterative
  - ○ Initially, all inputs and outputs are unmatched
  - ○ In each iteration, each unmatched input sends *request* to one of the unmatched outputs with non-empty queue
  - ○ Outputs, upon received requests, *accept* one of the requesting inputs and they get matched
  - ○ The iterations run till no more input-outputs can be matched

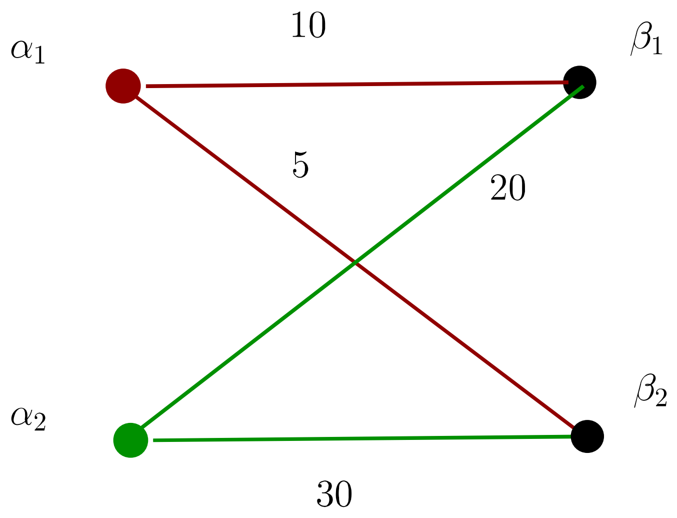
iSLIP Algorithm

- iSLIP algorithm
  - ○ It is iterative: request-grant-accept
  - ○ Only a bit of information is communicated in an iteration
  - ○ Hence, it is implementable
- Question:
  - ○ How should the iterations be performed if we could communicate more bits of information, so that
    - – performance improves with more communication, and
    - – with enough communication it becomes MWM
- We'll use max-product to obtain such a parametrized class of algorithms

Max Weight Matching: Bipartite Graph

- Bipartite graph $G = (V_1 \times V_2, E)$:
  - ○ $V_1 = \{\alpha_1, \ldots, \alpha_n\}$ and $V_2 = \{\beta_1, \ldots, \beta_n\}$
  - ○ $E = \{(\alpha_i, \beta_j) : 1 \leq i, j \leq n\}$
  - ○ Edge (α_i, β_j) has weight w_{ij}
  - ○ Goal: compute Max Wt Matching in G

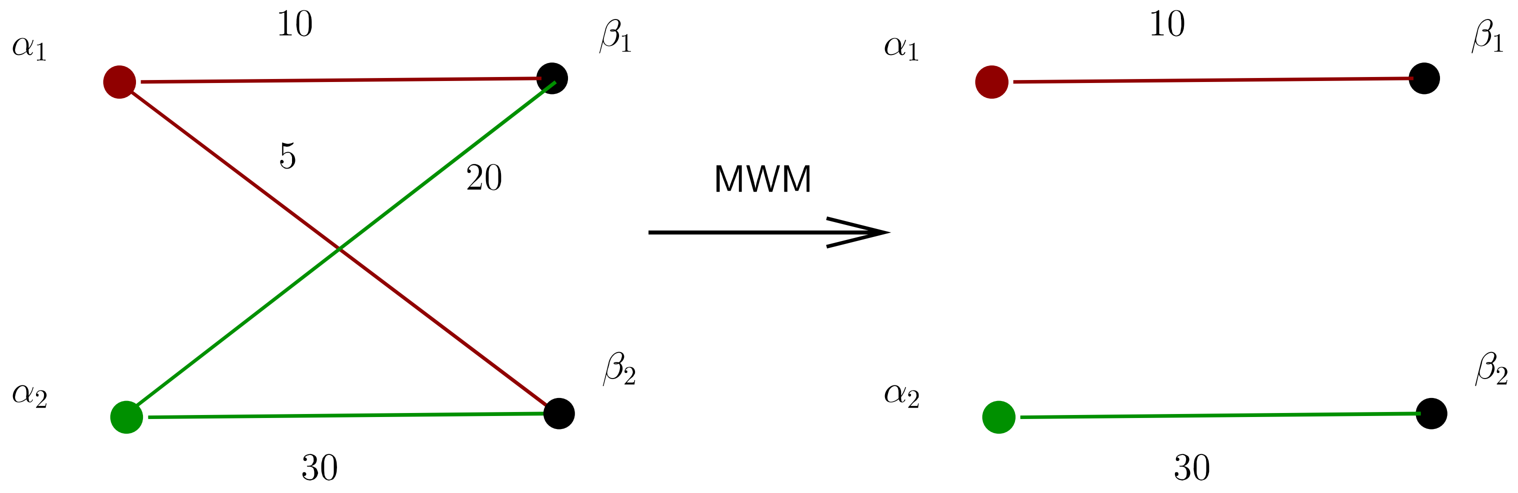
- An example with $n = 2$



Max Weight Matching: Bipartite Graph

- Bipartite graph $G = (V_1 \times V_2, E)$:
  - ○ $V_1 = \{\alpha_1, \ldots, \alpha_n\}$ and $V_2 = \{\beta_1, \ldots, \beta_n\}$
  - ○ $E = \{(\alpha_i, \beta_j) : 1 \leq i, j \leq n\}$
  - ○ Edge (α_i, β_j) has weight w_{ij}
  - ○ Goal: compute Max Wt Matching in G

- An example with $n = 2$



Pair-wise Constraints for Matching

- Bipartite graph $G = (V_1 \times V_2, E)$:
  - ○ Variables $(X_1, \ldots, X_n)$ for nodes in $V_1 = \{\alpha_1, \ldots, \alpha_n\}$
    - $X_i \in \{1, \ldots, n\}$
  - ○ Variables $(Y_1, \ldots, Y_n)$ for nodes in $V_2 = \{\beta_1, \ldots, \beta_n\}$
    - $Y_j \in \{1, \ldots, n\}$
  - ○ $E = \{(\alpha_i, \beta_j) : 1 \leq i, j \leq n\}$
    - Edge constraint

$$\psi_{\alpha_i \beta_j}(r, s) = \begin{cases} 0 & r = j \text{ and } s \neq i \\ 0 & r \neq j \text{ and } s = i \\ 1 & \text{Otherwise} \end{cases}$$

- Goal: compute $\arg\max\left(\sum_{i=1}^{n} w_{iX_i} + w_{Y_i i}\right)$, where $\prod_{i,j} \psi_{\alpha_i \beta_j}(X_i, Y_j) \; = \; 1.$

Max-Product Algorithm for MWM

- Algorithm parameters: in iteration t

  ○ Messages (numbers): $\hat{m}^t_{\alpha_i \to \beta_j}, \hat{m}^t_{\beta_j \to \alpha_i}$

- **Algorithm MP-MWM.**

  1. Initially, set: $t = 0$ and $\hat{m}^0_{\alpha_i \to \beta_j} = \hat{m}^0_{\beta_j \to \alpha_i} = w_{ij}$

  2. At iteration t:

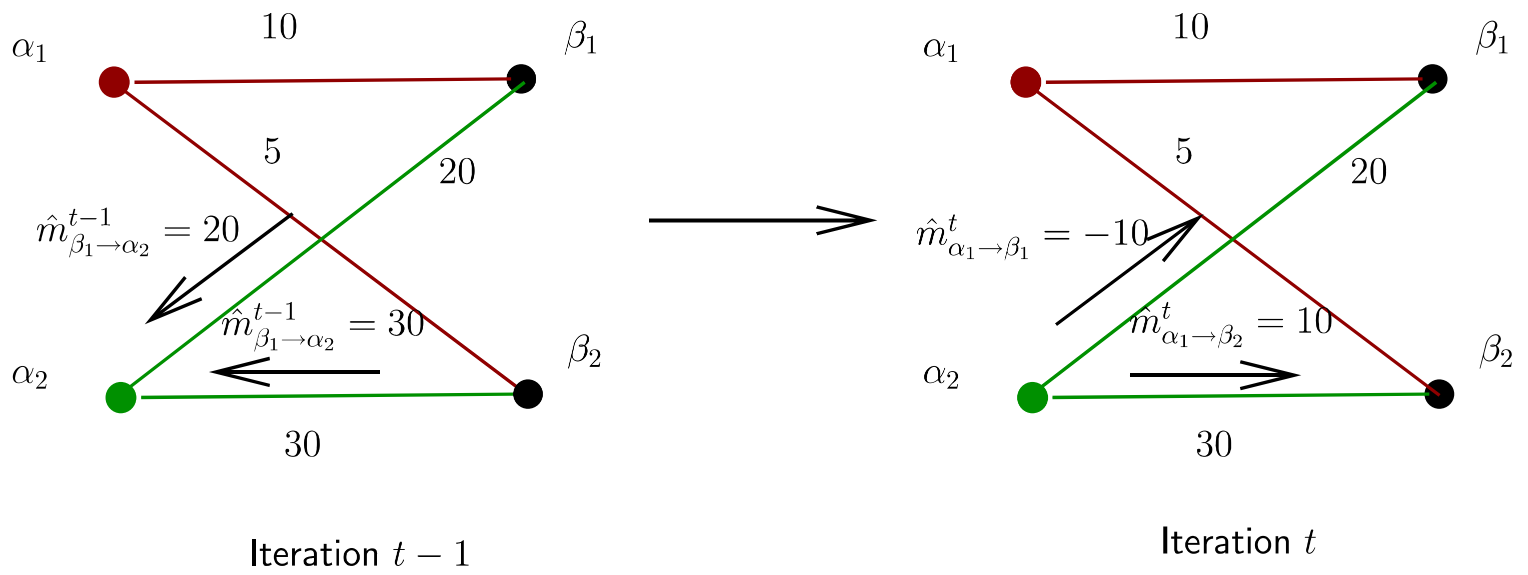
$$\begin{aligned} \hat{m}^t_{\alpha_i \to \beta_j} &= w_{ij} - \max_{\ell \neq j} \hat{m}^{t-1}_{\beta_\ell \to \alpha_i}, \\ \hat{m}^t_{\beta_j \to \alpha_i} &= w_{ij} - \max_{\ell \neq i} \hat{m}^{t-1}_{\alpha_\ell \to \beta_j}. \end{aligned}$$

  3. Estimate MWM π^t: node α_i estimates $\pi^t(i) = \arg\max_j \{\hat{m}^t_{\beta_j \to \alpha_i}\}$.

  4. Set $t = t + 1$ and repeat from **2** till π^t converges.

Example

- Consider an example with $n = 2$
  - ○ Figure shows messages of node α_2
  - ○ At iteration $t-1$, $\pi^{t-1}(2) = 2$ (i.e. α_2 connects to β_2)


α_1
10
β_1
5
20
$\hat{m}^{t-1}_{\beta_1 \to \alpha_2} = 20$
$\hat{m}^{t-1}_{\beta_1 \to \alpha_2} = 30$
α_2
β_2
30
Iteration $t-1$
α_1
10
β_1
5
20
$\hat{m}^{t}_{\alpha_1 \to \beta_1} = -10$
$\hat{m}^{t}_{\alpha_1 \to \beta_2} = 10$
α_2
β_2
30
Iteration t

Correctness of MP-MWM

- Notation
  - ○ Let ε be difference between weight of MWM and second MWM
    - $\rightarrow$ If MWM not unique, then $\varepsilon = 0$
  - ○ Let $w^* = \max_{ij} w_{ij}$
- **Theorem. [BSS05]** The MP-MWM algorithm (i.e. estimated matching π^t) converges to the correct MWM in $\frac{2nw^*}{\varepsilon}$ number of iterations.
- Implications: for fixed w^* and ε
  - ○ Number of iterations scale as $O(n)$
  - ○ Per-node computation in each iteration $O(n)$
  - $\rightarrow$ Total per-node computation $O(n^2)$, or
  - $\rightarrow$ Total computation cost of $O(n^3)$ overall
  - ○ Comparable performance to Edmond-Karp

MP-MWM for Scheduling

- The MP-MWM can be used to find max. wt. matching in switch
  - ◦ But, it converges only if max. wt. matching is unique
    - – This may not be the case for switch
  - ◦ Further, running time depends on queue-size
    - – The algorithm may run *forever*

- In summary,
  - ◦ Need to modify algorithm to make it convergent in bounded number of iterations

MP-MWM for Scheduling

- Given queue-size matrix $Q = [Q_{ij}]$ and $\varepsilon > 0$
  - ○ Let $Q^* = \max_{ij} Q_{ij}$
  - ○ Let $\delta = \varepsilon Q^*/n$

- Let δ_{ij} be chosen uniformly at random in $(\delta, 2\delta)$
  - ○ Define $W = [W_{ij}]$, where $W_{ij} = Q_{ij} + \delta_{ij}$

- MP-SCH algorithm:
  - ○ Run (a variant of) MP-MWM with W as weights

MP-SCH: Performance

- **Theorem. [BSS06]** The algorithm MP-SCH takes $O(n^2/\varepsilon)$ iterations to converge. For any λ such that $(1-2\varepsilon)^{-1}\lambda$ is admissible, the net average queue-size is bounded above as $O(n^2/\varepsilon)$ under *friendly* arrival process.

- Thus, algorithm MP-SCH
  - ○ Always converges, and
  - ○ Convergence time does not depend on queue-size

- However, compared to iSLIP
  - ○ MP-SCH takes a lot of iterations, and
  - ○ Too many bits are communicated in each iteration

Parametrized Approximation of MP-SCH

- The essential parameters are
  - ○ Number of bits communicated in each iteration, and
  - ○ Number of iterations
  - → We'll use them to design parametrized MP-SCH algorithm
- iSLIP allows for a single bit transmission
  - ○ It uses bits to communicate whether queue is > 0 or $= 0$
- If bit budget is k-bits, then do the following
  - ○ Convey the most significant k-bits of weights
  - → $1 - 2^{-k}$ approximation of weights
  - → In ideal setup, results in throughput loss of 2^{-k} fraction
- If iterations are limited, stop after *allowed* number of iterations

Parametrized Approximation of MP-SCH

- Additional trick : use of memory
  - ○ The weights of edges change only by 1 between time-slots
  - ○ That is, algorithm is essentially running with same instance in nearby time-slots
  - → Instead of re-starting algorithm with *new* messages, use the message values from previous time

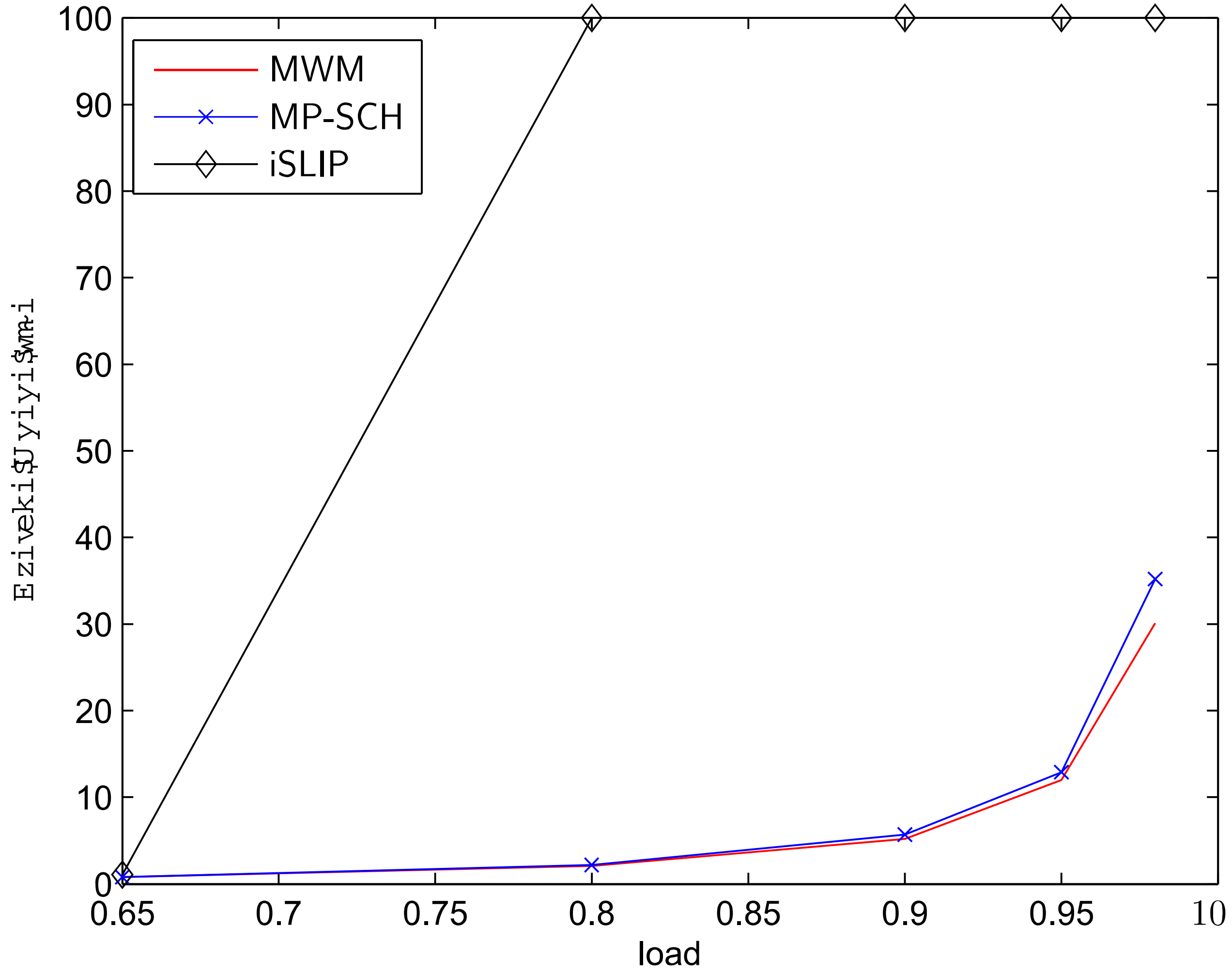
- Next, a representative set of simulation results
  - ○ To capture effect of
    - – Number of iterations
    - – Bits exchanged and
    - – Memory between time-slots

Experiments

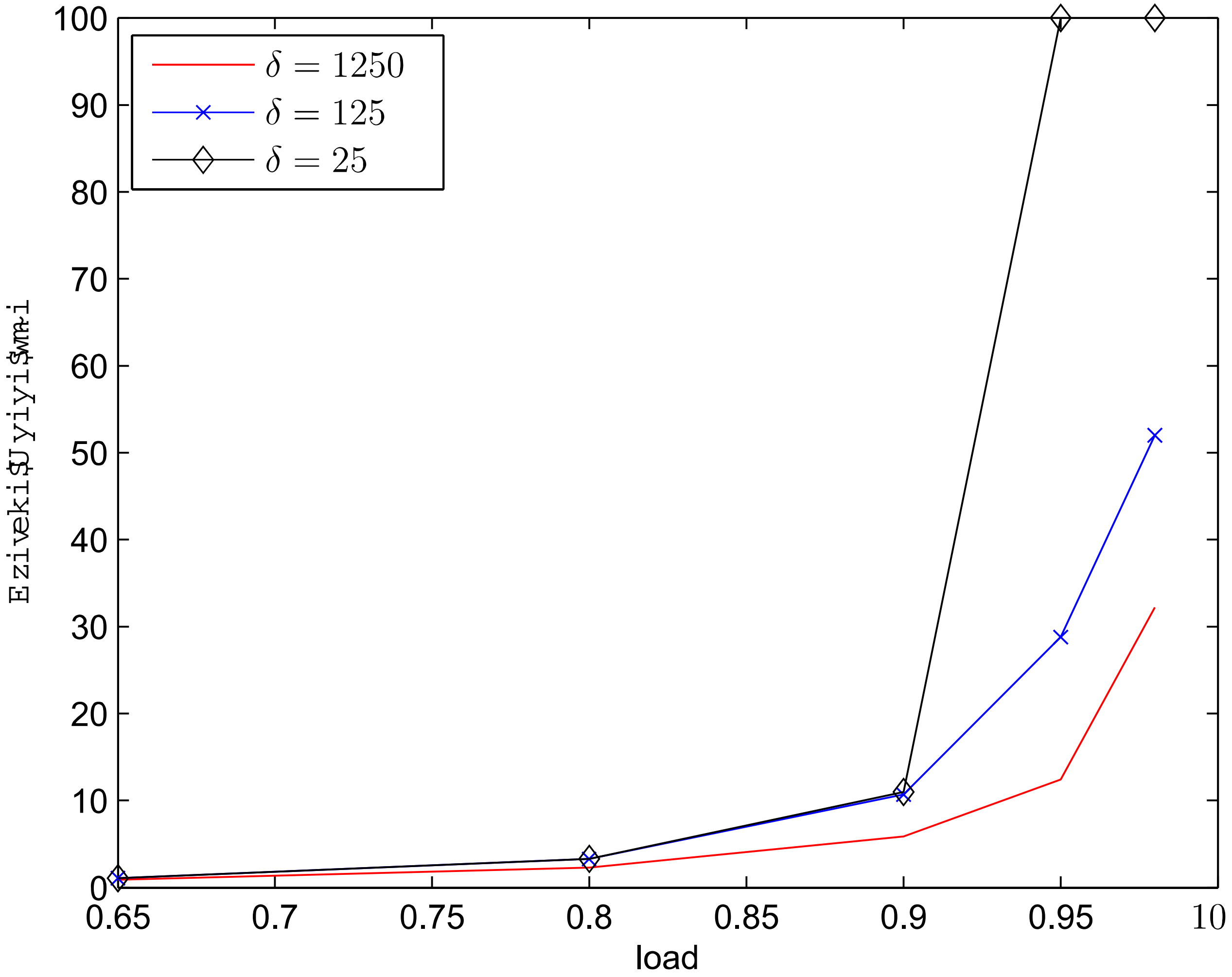
- Setup
    - ○ $n = 8$ port switch
    - ○ Traffic rate λ is *non-uniform*
        - Bernoulli i.i.d. distribution
    - ○ Finite buffer size for each VOQ is $B = 10000$
        - $\delta = \varepsilon B/n = 1250\varepsilon$
        - Recall: small δ means good performance, more iterations
    - ○ Number of iterations is equal to $n (= 8)$
        - Unless specified in some experiments

- Next, some plots

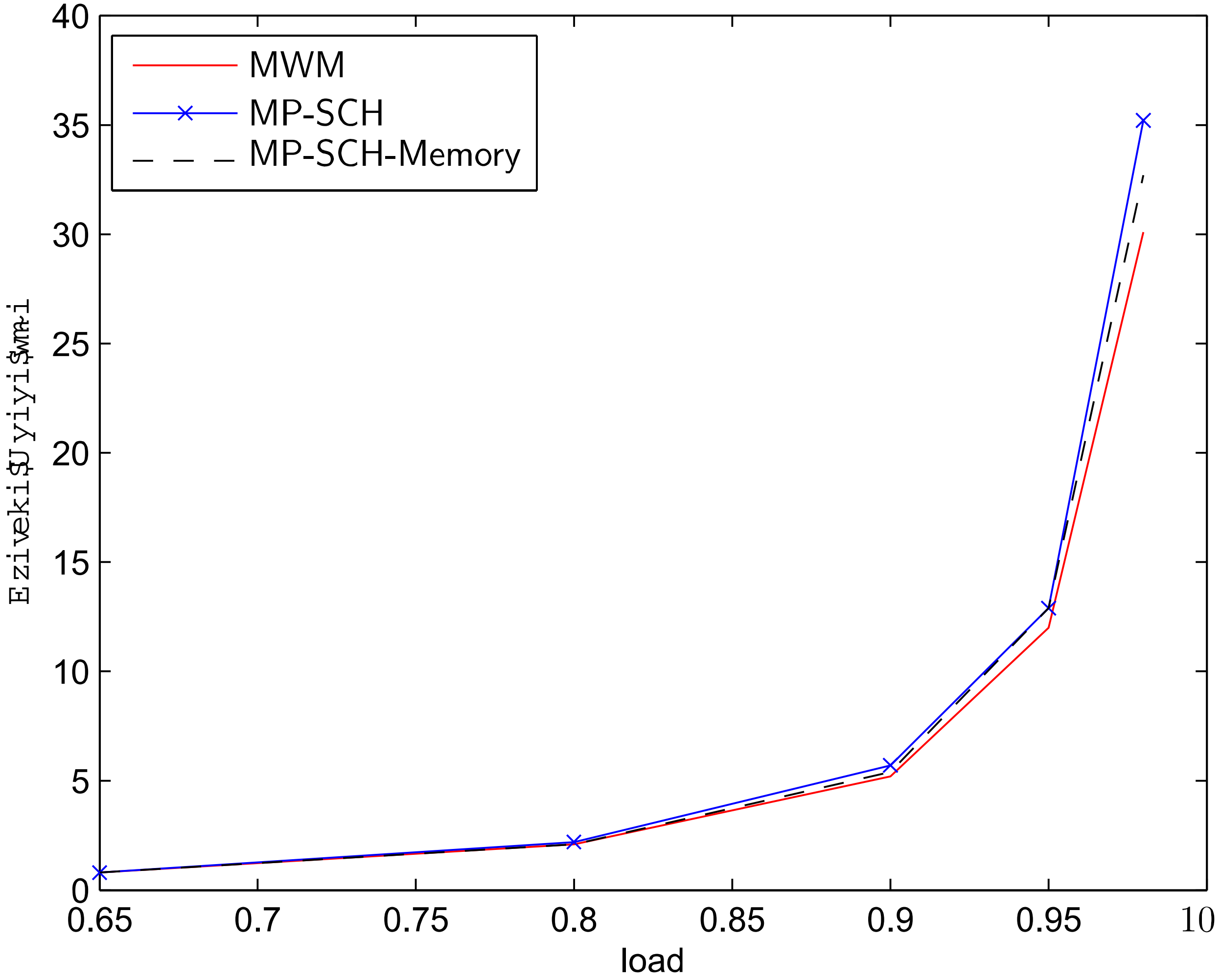
Base-case Performance



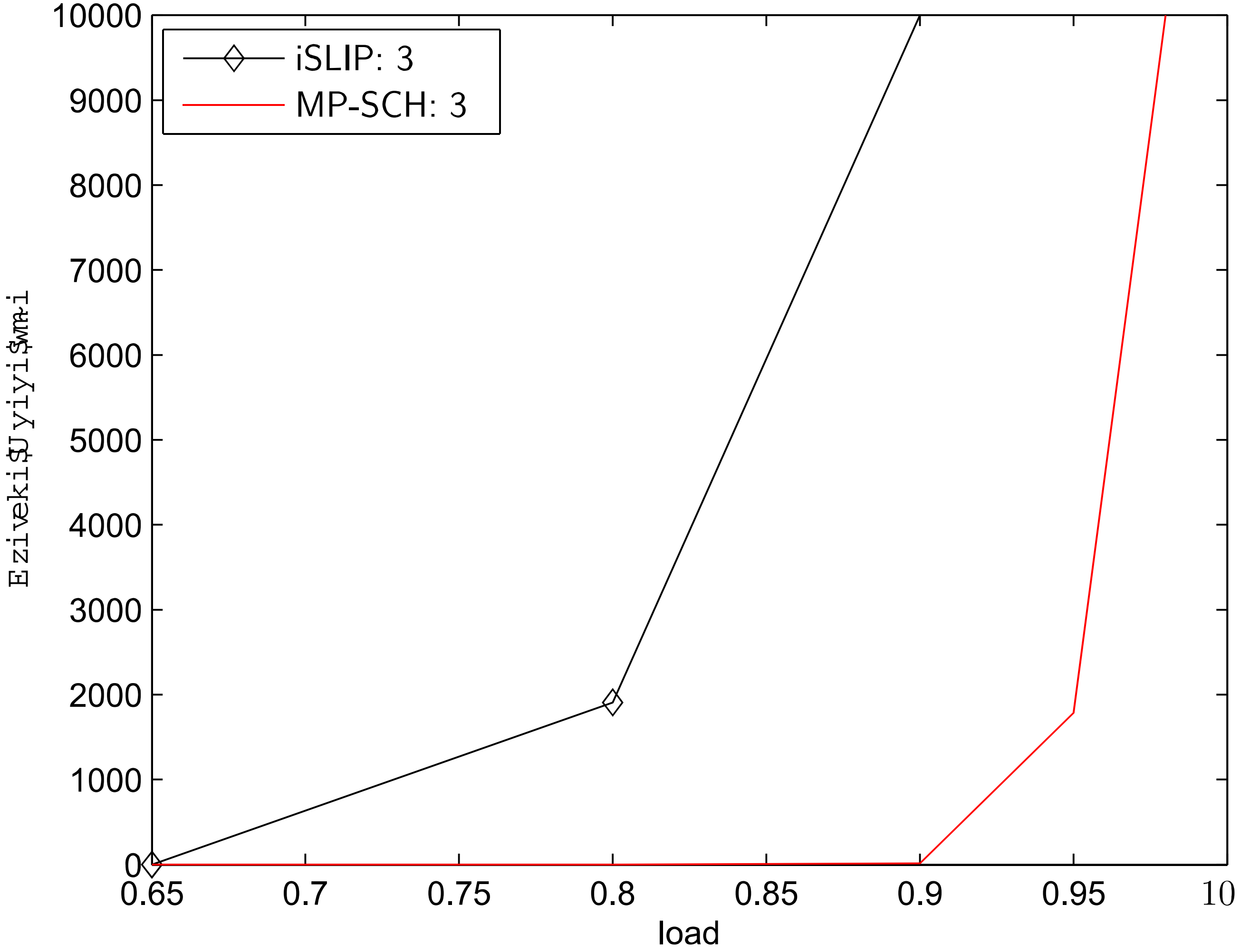
Effect of δ



Effect of Memory



Effect of Iterations

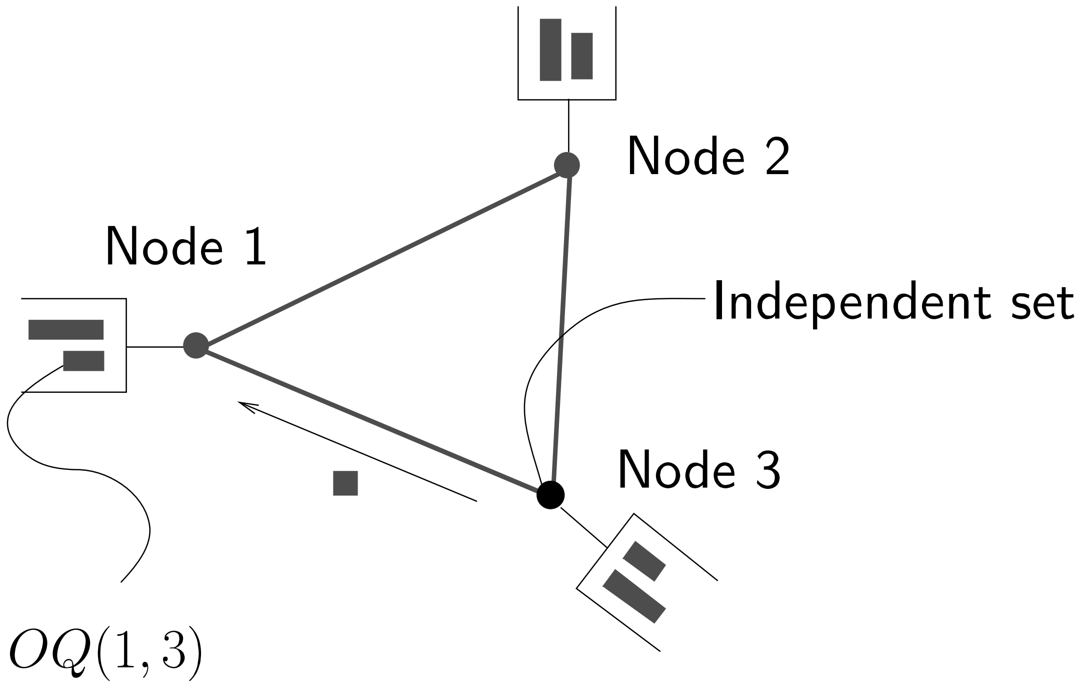


Wireless Scheduling using Max-product

Sujay Sanghavi   Alan Willsky

MIT

Wireless Network



- Scheduling constraint: at a given time
  - ○ Each node can transmit at most one packet, and
  - ○ No two neighbors can transmit simultaneously

→ Schedule is an independent set in the wireless network graph

Max Weight Independent Set

- A good scheduling algorithm
  - ○ Max. weight independent set (MWIS), where
    - – weight of a node is appropriate queue-size

- Performance of MWIS
  - ○ Provides 100 % throughput
  - ○ $O(n^2)$ net average queue-size

- What about algorithm to find MWIS ?

Max Weight Independent Set

- Maximum weight independent set problem
  - ○ Known to be computationally hard
  - ○ It is even hard to approximate

- In the context of wireless network
  - ○ It was shown that, it is computationally hard
    - – to have polynomial in n average queue-size
    - – Shah and Tsitsiklis (2007)

- We will use max-product as a heuristic
  - ○ As we shall see, it is a good approximation

Max. Weight Independent Set

- Given weighted graph $G = (V, E)$ of n nodes
  - ○ $V = \{1, \ldots, n\}$ and E being edges
  - ○ Let w_i be weight of node i
  - ○ Variable $x_i \in \{0, 1\}$ for node i
    - $x_i = 1$ implies i in independent set

- Optimization: IP

$$\begin{aligned} \text{maximize} \quad & \sum_i w_i x_i, \\ \text{subject to} \quad & x_i + x_j \leq 1, \quad \text{for all} \quad (i, j) \in E, \\ & x_i \in \{0, 1\} \quad \text{for all} \quad i \in V. \end{aligned}$$

Max. Weight Independent Set

- Relaxation of IP: LP

$$\begin{aligned} \text{maximize } & \sum_i w_i x_i, \\ \text{subject to } & x_i + x_j \le 1, \quad \text{for all } \ (i,j) \in E, \\ & x_i \ge 0, \quad \text{for all } \ i \in V. \end{aligned}$$

- The dual of LP

$$\begin{aligned} \text{minimize } & \sum_{(ij) \in E} \gamma_{ij}, \\ \text{subject to } & \sum_{k \in \mathcal{N}(i)} \gamma_{ik} \ge w_i, \quad \text{for all } \ i \in V, \\ & \gamma_{ij} \ge 0, \quad \text{for all } \ (i,j) \in E. \end{aligned}$$

- By strong duality
  - ○ Value of LP $=$ value of it's dual

Max-Product Algorithm for MWIS

- Algorithm parameters: in iteration t
  - ○ Messages (numbers): $\hat{m}^t_{(i,j)}$ for each $(i,j) \in E$

- **Algorithm MP-MWIS.**

  1. Initially, set: $t = 0$ and $\hat{m}^0_{(i,j)} = \max\{0, w_i, w_j\}$ for all $(i,j) \in E$

  2. At iteration t:

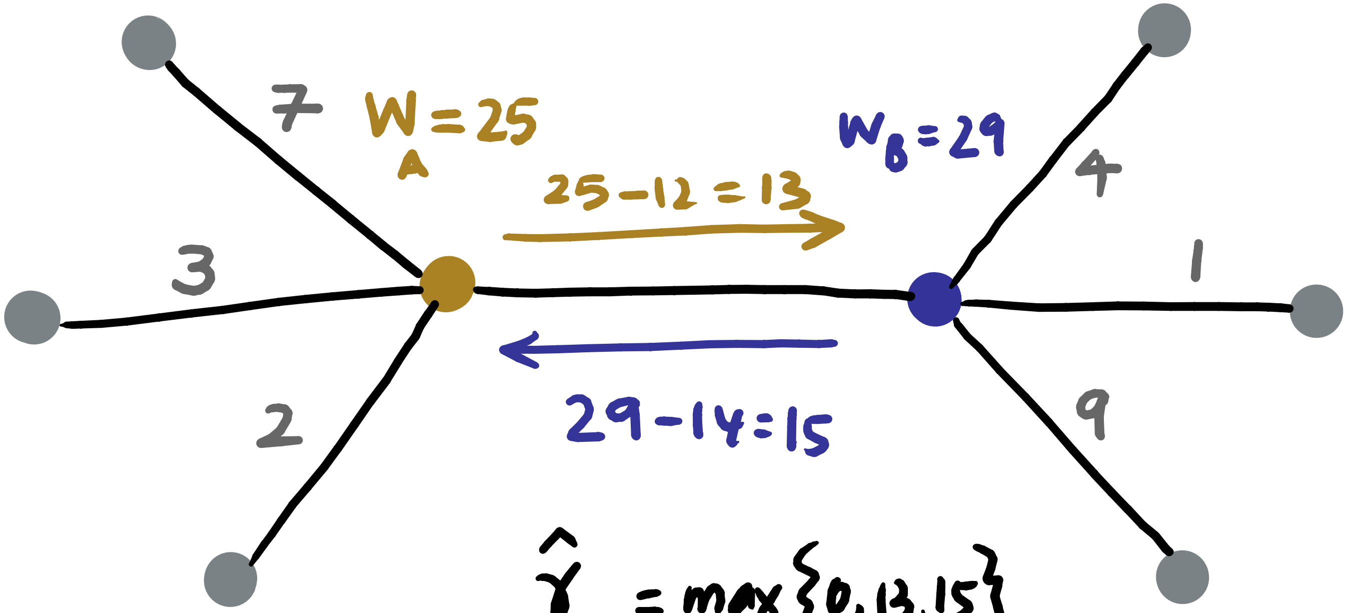
$$\hat{m}^t_{(i,j)} \;=\; \max\left\{0, w_j - \max_{\ell \neq i}\; \hat{m}^{t-1}_{(\ell,j)}, w_i - \max_{\ell \neq j}\; \hat{m}^{t-1}_{(\ell,i)}\right\}.$$

  3. Estimate MWIS $\mathbf{x}^t$:

$$\mathbf{x}_i = \begin{cases} 1 & \text{if } 0 < \sum_{\ell \in \mathcal{N}(i)} \hat{m}^t_{(\ell,i)} \leq w_i \\ 0 & \text{otherwise.} \end{cases}$$

  4. Set $t = t + 1$ and repeat from **2** till $\mathbf{x}^t$ converges.

Max-product for MWIS



$$\hat{\gamma}_{AB} = \max\{0, 13, 15\}$$

$$= 15$$

Max-product for MWIS

- The MP-MWIS algorithm
  - ○ Is a co-ordinate descent algorithm for dual optimization
  - ○ It may not converge to dual optimal solution

- Consider the convergent modification of MP-MWIS
  - ○ Add penalty function for $\sum_k \gamma_{ik} \geq w_i$:

$$\varepsilon \times \log \left(\sum_k \gamma_{ik} - w_i \right)$$

  - ○ Corresponding modification in MP-MWIS is of the form:

$$\hat{m}^t_{(i,j)} \;=\; \eta(\varepsilon) + \max \left\{ 0, w_j - \max_{\ell \neq i} \; \hat{m}^{t-1}_{(\ell,j)}, w_i - \max_{\ell \neq j} \; \hat{m}^{t-1}_{(\ell,i)} \right\},$$

    – where $\eta(\varepsilon) \in (\varepsilon/2, \varepsilon)$, which is locally computable

Max-product for MWIS

- In summary,
  - ○ The MP-MWIS is a co-ordinate descent for the dual
  - ○ When modified, it converges to (approximate) solution of dual
- Implications
  - ○ Provides a *good* implementable solution for any graph
  - ○ For bipartite graphs
    - there is lack of integrality gap
    - optimal dual solution allows for finding optimal MWIS, when it's unique
    - for small ε, it's possible to recover the unique optimal MWIS by means of modified MP-MWIS
- The MP-MWIS algorithm can be used for scheduling

Discussion

- Design philosophy for network algorithms
  - ○ Parametrized class of algorithms that work for a range of constraints
    - – Rather than an optimal algorithm for given constraints
- Iterative message passing algorithms such as max-product
  - ○ Excellent method for designing such parametrized algorithms
  - ○ This was demonstrated in the context of switch and wireless scheduling
- Other implications
  - ○ Convergence and correctness of max-product algorithm
  - ○ Precise relation with co-ordinate descent algorithm for dual problem
  - ○ Extends for k-factors