Iterative Scheduling Algorithms

Mohsen Bayati
EE, Stanford
bayati@stanford.edu

Balaji Prabhakar
EE & CS, Stanford
balaji@stanford.edu

Devavrat Shah
EECS, MIT
devavrat@mit.edu

Mayank Sharma
IBM TJ Watson Research, NY
mxsharma@us.ibm.com

***Abstract*— The input-queued switch architecture is widely used in Internet routers due to its ability to run at very high line speeds. A central problem in designing an input-queued switch is the scheduling algorithm that decides which packets to transfer from ingress ports to egress ports in a given timeslot. It is desirable that such algorithms be iterative (so as to be pipelineable), distributed (allowing flexibility in hardware implementation) and are able to deliver high performance (in terms of throughput and delay). In practice, implementable algorithms have so far had limited success in combining all of the above properties. For example, the popular iSLIP [1] algorithm is known to perform suboptimally, but it is commercially deployed mainly because it is iterative and distributed. The main contribution of this paper is the design and systematic analysis of two algorithms which, to the best of our knowledge, are the first high-performance iterative and distributed scheduling algorithms with possibility of efficient implementation.**

We first present an iterative, distributed and low-delay maximal throughput algorithm based on the celebrated "Auction Algorithm" [2], [3]. This algorithm can be seen as a natural extension of iSLIP when queue-size information is allowed to be exchanged. The standard auction algorithm can take an unbounded number of iterations to converge in the worst case. However we show that under admissible Bernoulli i.i.d. traffic, our algorithm takes $O(n^2)$ iterations, where n is the number of ingress/egress ports in the switch. Moreover for a switch with finite buffer-size, the algorithm allows for a graceful trade-off between running time and performance, which we verify by representative simulation results.

Next, we propose and analyze a throughput-optimal, iterative and distributed scheduling algorithm influenced by Max-Product Belief Propagation [4], [5]. Recently the problem of efficient transmission over multi-hop wireless networks has been formulated as that of finding an appropriate schedule over the grid-graph abstraction of the network. A key feature of the multi-hop wireless transmission problem is that while the communication subgraph is bipartite, the bi-partition is allowed to change in each scheduling epoch. We show that our algorithm can be used to efficiently schedule traffic in multi-hop wireless networks.

I. Introduction

Scheduling is an essential operational task required in any large network in order to allocate resources, like bandwidth and hardware, to various competing entities such as data flows or packets. The main challenge in designing a good scheduling algorithm is in achieving a balance between performance and implementability. Motivated by this consideration we primarily consider the problem of scheduling in an input-queued switch, and a related problem of scheduling in a multi-hop wireless network secondarily.

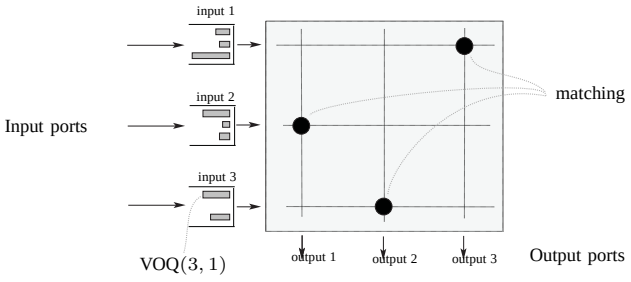

Fig. 1. An input-queued switch, and a matching of inputs to outputs.

A. Input-queued switch

Switching is an integral function in an Internet router that transfers packets arriving at ingress (input) ports to egress (output) ports. There are a variety of possible switch architectures − in this paper we are concerned with input-queued (IQ) switches and will next describe how an IQ switch operates.

Figure 1 illustrates a 3×3 IQ switch fabric, by which we mean the switch has 3 input ports and 3 output ports. (Not all ports need be used, so there is no loss in generality in assuming an equal number of input and output ports.) Packets arriving at input i destined for output j are stored in the Virtual Output Queue VOQ(i, j). In each timeslot, the switch fabric can transmit a number of packets from input ports to output ports, subject to the constraints:

i. each input can transmit at most one packet,
ii. each output can receive at most one packet.

Another way to express this is to say that, in each timeslot, the switch can choose a *matching* from inputs to outputs. For example, Figure 1 illustrates a matching in which one packet is transmitted from input port 1 to output port 3, and one from input port 2 to output port 1. The figure also shows a match from input port 3 to output port 2, but since VOQ$(3, 2)$ is empty no packet is transmitted.

The constraints (i) & (ii) mean that the buffer memory needs to be accessed only twice per timeslot (once to write an incoming packet, once to read a packet for transmission). This low memory bandwidth requirement implies that IQ switches can operate at very high speeds. The constraint (ii) means that no buffers are required at the output ports. We have assumed here and throughout this paper that all packets are of equal size, and that time is slotted so that at most one packet may arrive in any timeslot. In practice, packets are not all the same size, but they are broken up into equal-sized cells before being transmitted across the switch fabric.

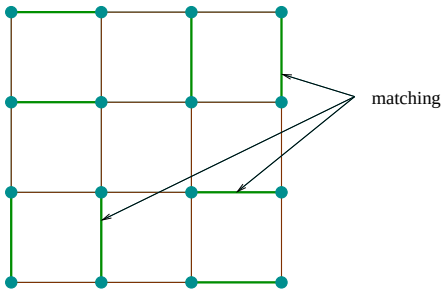

Fig. 2. A 16 node (4×4) grid graph as a model for wireless network.

Scheduling algorithm. The specific matching of inputs to outputs in each timeslot is chosen by a *scheduling algorithm*. It may take into account various kinds of information such as queue sizes, ages of packets, or quality-of-service constraints.

For the purpose of this paper, one scheduling algorithm is of particular interest: the *Maximum-Weight Matching* (MWM) algorithm. In every timeslot, this algorithm chooses a matching as follows: Let Q_{ij} be the queue size at VOQ(i, j). Given a matching that matches input i to output $o(i)$, define the *weight* of that matching to be $\sum_i Q_{i\,o(i)}$. Among all possible matchings choose one with the greatest weight (breaking ties arbitrarily).

The two main metrics for evaluating the performance of a scheduling algorithm are throughput and delay. Roughly speaking, an algorithm is said to have *100% throughput* if it can carry as much traffic as an omniscient scheduling algorithm (i.e. one which knows all future packet arrivals). This is formalized later in the paper. Delay performance is harder to quantify; we discuss it further below. Our objective in this paper is the design of scheduling algorithms that have *100%* throughput, low delay, are simple in terms of data structure and logic requirement, and are iterative and distributed.

B. Wireless Network

A multi-hop wireless network, shared by many users, arises in many situations such as a wireless mesh network. A good model for network topology is the two-dimensional grid-graph. An example of a 16 node grid-graph is depicted in Figure 2.

The need to schedule the transmissions between nodes arises due to the *interference* caused by the signals sharing the broadcast wireless medium. In other words, the transmission from one node can adversely affect the transmission of other node in a wireless environment. A popular model for interference is the *node-exclusive* model: each node can either transmit to or receive from at most one other node at any given time. That is, simultaneously transmitting nodes and receiving nodes must respectively form the two partitions of a bipartite graph, and be connected via a matching. Hence, a scheduling algorithm is required to pick schedule or matching (of transmitter-receiver pairs) at each time, with the objective of maximizing network throughput and minimizing the delay.

As shown in previous work including [6]–[9], a good candidate scheduling algorithm is the Maximum Weight Matching (MWM) algorithm, where the weight of a transmission is the difference between the queue-size at transmitting node and the receiving node (also called back-pressure policy).

In the interest of space, unlike the problem of switch scheduling we will not go into details of the wireless scheduling problem in this paper and instead point the reader to the literature cited. However, as we shall shortly see, the techniques used for both problems will be very similar.

C. Previous work

The IQ switch architecture has been studied for more than a decade [10]–[13]. A good deal is now known about the throughput charactersitics of the IQ switch. MWM has been shown to have 100 % throughput, under a 'friendly' arrival distribution [14]. A generalization of this result in the context of multi-hop networks (under the same arrival distribution) has also been shown earlier [15]. These results have been generalized to arbitrary arrival distributions [16]. A class of algorithms akin to MWM have also been shown to have 100 % throughput [17]–[19]. Further, when the appropriate function of queue-size is used as the weight, the algorithm has been shown to possess a certain delay optimality property [20].

Though MWM and related algorithms provide maximal throughput, the network-flow based algorithms such as that of Edmonds and Karp [21] which find the MWM in finite time (independent of weight) are too complex to implement since they are centralized and require the maintainenance of a lot of data structure. This has motivated the design of simpler high-performance scheduling algorithms. The iSLIP [1] algorithm has been commercially successful as it is distributed, iterative and requires simple hardware operations. However, it is not throughput optimal. Other notable algorithm [22]–[24] are simple to implement and throughput optimal. But they are either centralized or provide poor delay performance (or no guarantees on delay performance at all).

In the context of wireless networks, there has been recent work [7]–[9] that proposes a variant of iSLIP as the scheduling algorithm. Again, though simple, these algorithms are not throughput optimal.

D. Contribution

This work is motivated by the desire to design iterative, distributed and simple algorithms that have maximal throughput and low delay. The auction algorithm of Bertsekas [2] has some key similarities with the iSLIP algorithm and is therefore very appealing as a starting point. However, its running time is proportional to the largest weight (or queue-size) which may lead to undesirable performance.

In this paper, we consider a variation of the auction algorithm and show that under 'friendly' arrival traffic it takes $O(n^2)$ iterations to converge to a solution. This has the immediate implication that the algorithm has 100% throughput and has a net average queue-size of $O(n^2)$. The iterative and distributed nature of this algorithm allows for a pipelined

architecture and flexibility in hardware implementation in different components. For example, each input/output port can host a logic processor with memory and they communicate with each other to calculate the optimal schedule every time in $O(n^2)$ iterations. These results are presented in Section II.

In practice, buffers are finite and our variant of the auction algorithm suggests that in this case, a trade-off between performance and computational complexity can be obtained by tuning a single parameter. Obtaining a precise quantification of the throughput region for a system with finite buffers is known to be a hard theorectical problem. Consequently, we are unable to provide a justification for the claimed trade-off in performance. This discussion is presented in Section II-D.

The auction algorithm requires a bi-partition of the graph since it treats the partitions asymmetrically. In case of a switch, the inputs and outputs form a natural bi-partition. The wireless network when modeled as a grid-graph is bipartite. However, nodes do not know their partition a priori, and creating a partition in a distributed manner is essentially a lot of work. For this reason, we need a 'symmetric' version of the auction algorithm. In Section III, we present an iterative, distributed algorithm motivated by the Max-Product (MP) algorithm for MWM. MP is a message-passing algorithm that has been extremely successful as a heuristic for solving hard combinatorial optimization problems [25]–[27]. The roots of MP lie in statistical physics and AI [4], [27]. In recent work [28], [29], we have developed an MP-based algorithm which solves the MWM problem exactly for bipartite graphs which have a unique optimum.

II. Switch Scheduling: Auction Algorithm

In this section, we describe the auction algorithm for switch scheduling and establish that it takes $O(n^2)$ iterations. For a switch with finite buffers we discuss the possible trade-off obtainable between performance and the running time of the algorithm. We support our claims using simulation results.

A. Notation

We first specify our notation. Let $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$ and $\mathbb{Z}_+ = \{i \in \mathbb{Z} : i \geq 0\}$. Let 1_X be the indicator function: $1_{\text{true}} = 1$ and $1_{\text{false}} = 0$.

Let $\mathbb{M}$ be the set of $n \times n$ real-valued matrices, and $\mathbb{M}_+$ the subset consisting of $\mathbb{R}_+$-valued matrices. Write matrices as $\mathbf{a} = [a_{ij}]$. Denote by $\mathbf{a} \cdot \mathbf{b}$ as $\sum_{ij} \mathrm{a}_{ij} b_{ij}$. Let $\mathbf{1} = [1]$. Let $\mathbb{S} \subset \mathbb{M}_+$ be the set of matrices whose row sums and column sums are all equal to 1, i.e. the set of doubly stochastic matrices. Let $\mathbb{P} \subset \mathbb{S}$ be the set of matrices $\boldsymbol{\pi}$ for which $\pi_{ij} \in \{0, 1\}$ for all i and j, i.e. the set of permutation matrices. These correspond to matchings in the switch bipartite graph with n inputs and n outputs.

Let timeslots be indexed by $\tau \in \mathbb{Z}_+$, starting at $\tau = 0$. Let $\mathbf{Q}(\tau) = [Q_{ij}(\tau)] \in \mathbb{M}_+$ denote the matrix of the queue sizes at the end of timeslot τ. We assume $\mathbf{Q}(0) = \mathbf{0}$. Since work arrives in discrete packets, $Q_{ij}(\tau) \in \mathbb{Z}_+$ for all τ.

Next we describe the dynamics of $\mathbf{Q}(\cdot)$, which depends on the arrival process and the scheduling algorithm. Let $\mathbf{A}(\tau)$ be the cumulative arrival process up to timeslot τ, i.e. $A_{ij}(\tau)$ is the number of packets that have arrived at input i destined for output j in the time interval $[0, \tau]$, with $\mathbf{A}(0) = \mathbf{0}$. The arrivals in timeslot τ are thus $\mathbf{a}(\tau) \stackrel{\triangle}{=} \mathbf{A}(\tau) - \mathbf{A}(\tau - 1)$. In this paper, we make the following standard assumption that the arrivals, $\mathbf{a}_{ij}(\tau)$, are Bernoulli i.i.d. across time with $\Pr(\mathrm{a}_{ij}(\tau) = 1) = \lambda_{ij}$ and the arrival rate matrix $\lambda = [\lambda_{ij}]$ is *admissible*, that is,

$$\sum_{k=1}^{n} \lambda_{ik} < 1, \quad \sum_{k=1}^{n} \lambda_{kj} < 1, \quad \forall\ i,\ j.$$

Similarly, let $\mathbf{D}(\tau)$ be the cumulative departure process from the virtual output queues. Then

$$\mathbf{Q}(\tau) = \mathbf{Q}(0) + \mathbf{A}(\tau) - \mathbf{D}(\tau) = \mathbf{A}(\tau) - \mathbf{D}(\tau), \qquad (1)$$

since $\mathbf{Q}(0) = \mathbf{0}$. Now we specify the scheduling algorithm. Let $S_{\boldsymbol{\pi}}(\tau)$ be the cumulative number of timeslots that the scheduling algorithm has devoted to matching $\boldsymbol{\pi} \in \mathbb{P}$ in the time interval $[0, \tau]$, with $S_{\boldsymbol{\pi}}(0) = 0$ for all $\boldsymbol{\pi}$. We will use the convention that departures in timeslot τ happen at the beginning of the timeslot, and that arrivals happen at the end, so that

$$D_{ij}(\tau) - D_{ij}(\tau - 1) = \sum_{\boldsymbol{\pi} \in \mathbb{P}} \pi_{ij} \big(S_{\boldsymbol{\pi}}(\tau) - S_{\boldsymbol{\pi}}(\tau - 1)\big) 1_{Q_{ij}(\tau-1)>0}. \qquad (2)$$

Before proceeding further, we recall the following well-known and well utilized fact: given an admissible λ, which is a doubly sub-stochastic matrix, by the Birkhoff-Von Neumann theorem

$$\lambda = \sum_{k=1}^{n^2} \alpha_k \pi_k, \quad \alpha_k \geq 0, \ \sum_k \alpha_k < 1, \quad \pi_k \in \mathbb{P}.$$

B. Auction Scheduling Algorithm

For ease of explanation, we denote inputs by $\alpha_1, \ldots, \alpha_n$ and outputs by $\beta_1, \ldots, \beta_n$. As noted earlier, at time τ the weight of an edge (α_i, β_j) is $Q_{ij}(\tau - 1)$ and the weight of the matching π is $\sum_{i=1}^{n} Q_{i\pi(i)}(\tau - 1)$. A Maximum Weight Matching $\pi^*(\tau)$ at time τ is such that

$$\pi^*(\tau) \in \arg\max_{\pi \in \mathbb{P}} \sum_{i=1}^{n} Q_{i\pi(i)}(\tau - 1).$$

Now we describe the auction algorithm with parameter $\varepsilon > 0$. In the description of the algorithm, we drop reference to time τ for the queue-size. Readers familiar with the iSLIP algorithm may notice a striking syntactic similarity between the iSLIP and auction algorithms: both algorithms iterate between inputs proposing and outputs accepting/refusing. This similarity suggests that the auction algorithm is likely to have a simple implementation.

ε-Auction Algorithm.

- Given queue-size matrix $\mathbf{Q}$, let $Q^* = \max_{ij} Q_{ij}$ which is determined as follows:
  - Each output β_j computes $Q^*_{\cdot j} = \max_{k=1}^{n} Q_{kj}$.

- Each input α_i obtains $Q^*_{\cdot j}$ from all outputs β_j and computes $Q^* = \max_j Q^*_{\cdot j}$.
- Each output β_j contacts input α_j to obtain Q^*.

◦ Set $\delta = \varepsilon Q^*/n$. The algorithm will find a matching in two phases. Initially, the set of matched inputs-outputs $S = \emptyset$; the set of unassigned inputs $I = \{\alpha_1, \ldots, \alpha_n\}$, and parameters $p_j = 0$ for $1 \le j \le n$.

◦ *Phase 1: Bidding* For all $\alpha_i \in I$,

(1) Find the 'weight' maximizing output β_j. Let,

$$j_i = \text{argmax}_j\{Q_{ij} - p_j\},\ v_i = \max_j\{Q_{ij} - p_j\}, \quad (3)$$

$$\text{and } u_i = \max_{j \ne j_i}\{Q_{ij} - p_j\}. \quad (4)$$

(2) Compute the 'proposal' of input α_i, denoted by $b_{\alpha_i \to \beta_j}$ as follows:

$$b_{\alpha_i \to \beta_{j_i}} = Q_{ij_i} - u_i + \delta.$$

◦ *Phase 2: Assignment.* For each output β_j,

(3) Let $P(j)$ be the set of inputs from which β_j received a 'proposal'. If $P(j) \ne \emptyset$, increase p_j to the highest bid, i.e.

$$p_j = \max_{\alpha_i \in P(j)} b_{\alpha_i \to \beta_j}.$$

(4) Remove the maximum proposing input α_{i_j} from I and add (α_{i_j}, β_j) to S. If $(\alpha_k, \beta_j) \in S,\ k \ne i_j$, then put α_k back in I.

C. Analysis

The auction algorithm described above is slight variant of Bertsekas' auction algorithm. Given a fixed weighted bipartite graph, the behavior of the auction algorithm is well understood. However, the algorithm converges only if all the weights are finite. In our setup, weights are given by $\mathbf{Q}(\cdot)$. Hence, it is not clear if the above described algorithm will maintain finite queue-sizes $Q^*(\cdot)$ with probability 1. Specifically, the size of $Q^*(\cdot)$ directly affects the number of iterations required by the algorithm to converge. We state the following result.

Theorem 1 *Given $\varepsilon > 0$, let $\lambda = \sum_k \alpha_k \pi_k$ be such that $\sum_k \alpha_k \le 1 - 2\varepsilon$. Then, for a switch operating under the ε-Auction algorithm*

$$\limsup_{\tau \to \infty} \mathbb{E}\left[\sum_{ij} Q_{ij}(\tau)\right] = O(n^2/\varepsilon).$$

Further, the ε-Auction algorithm takes $O(n^2/\varepsilon)$ iterations.

Proof: In [3], Bertsekas studied the auction algorithm where δ was independent of the weights of the bipartite graph. In our algorithm we select $\delta = \varepsilon Q^*/n$. Ignoring the specific selection of δ, the standard auction algorithm of Bertsekas with a given $\delta > 0$ has the following property.

Lemma 2 ([3]) *Given $\delta > 0$, the auction algorithm finds a matching S in $O(nQ^*/\delta)$ iterations. The weight this matching is at least $\left(\max_{\pi \in \mathbb{P}} \sum_{i\pi(i)} Q_{i\pi(i)} - n\delta\right)$.*

We skip the proof of the above lemma. The interested reader can find an elegant proof in [2], [3].

In Lemma 2, since we select $\delta = \varepsilon Q^*/n$ assuming that $Q^* < \infty$, so the algorithm is well-defined and it always converges in $O(n^2/\varepsilon)$ iterations. Further, the weight of the resulting matching is at least $(1-\varepsilon)W^*$, $W^* = \max_{\pi \in \mathbb{P}} \sum_{i\pi(i)} Q_{i\pi(i)}$, each time for the following reason: by Lemma 2 the weight of resulting matching is at least $W^* - n\delta = W^* - Q^*\varepsilon$; $Q^* \le W^*$ and hence it is at least $(1-\varepsilon)W^*$.

To complete the proof of Theorem, we show that (a) $Q^* < \infty$ with probability 1 under the ε-Auction algorithm, and (b) the claimed bound on the net average queue-size in statement of Theorem 1 holds.

For this we will use Lyapunov function based arguments. Define the Lyapunov function

$$L(\mathbf{Q}(\tau)) \quad = \quad \mathbf{Q}(\tau) \cdot \mathbf{Q}(\tau) = \sum_{ij} Q^2_{ij}(m). \quad (5)$$

From Foster's criteria (see [30]–[33]), it follows that

$$\limsup_{\tau \to \infty} \mathbb{E}[Q_{ij}(\tau)] < \infty, \forall i, j,$$

if for all τ,

$$\mathbb{E}[L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau))|\mathbf{Q}(\tau)] \quad \le \quad -\gamma\|Q(\tau)\|_1 + B, \quad (6)$$

where γ, B are some positive constants. Now, consider the following.

$$\begin{aligned} &L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau)) \\ &\quad = \sum_{i,j}[Q^2_{ij}(\tau+1) - Q^2_{ij}(\tau)] \\ &\quad = \sum_{i,j}[Q_{ij}(\tau+1) - Q_{ij}(\tau)][Q_{ij}(\tau+1) + Q_{ij}(\tau)]. \end{aligned}$$

From the dynamics of the $\mathbf{Q}(\cdot)$, we obtain the following.

$$\begin{aligned} &L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau)) \\ &\quad = \sum_{i,j} 2Q_{ij}(\tau)\left(\mathsf{a}_{ij}(\tau+1) - D_{ij}(\tau+1)\right) \\ &\qquad + \quad \sum_{i,j}(\mathsf{a}_{ij}(\tau+1) - D_{ij}(\tau+1))^2. \end{aligned}$$

Now, in a time slot, at most 1 packet arrive and 1 packet depart from a VOQ. So $(\mathsf{a}_{ij}(\tau+1) - D_{ij}(\tau+1)) \in \{-1, 0, 1\}$. Hence,

$$\sum_{i,j}(\mathsf{a}_{ij}(\tau+1) - D_{ij}(\tau+1))^2 \quad \le \quad 2n. \quad (7)$$

Let $\pi(\cdot)$ be the schedule (matching) chosen by the algorithm. Then,

$$Q_{ij}(\tau)D_{ij}(\tau+1) \quad = \quad Q_{ij}(\tau)\pi_{ij}(\tau+1). \quad (8)$$

From above, we obtain

$$\begin{aligned} &L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau)) \\ &\quad \le \sum_{i,j} 2Q_{ij}(\tau)\left(\mathsf{a}_{ij}(\tau+1) - \pi_{ij}(\tau+1)\right) + 2n, \\ &\quad = 2\mathbf{Q}(\tau) \cdot \mathbf{a}(\tau+1) - 2\mathbf{Q}(\tau) \cdot \pi(\tau+1) + 2n. \end{aligned}$$

Now, taking conditional expectation with respect to $\mathbf{Q}(\tau)$, we obtain

$$\begin{aligned}\mathbb{E}\left[L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau))|\mathbf{Q}(\tau)\right] \\ \leq 2\mathbf{Q}(\tau)\cdot\lambda - 2\mathbf{Q}(\tau)\cdot\pi(\tau+1) + 2n.\end{aligned}$$

We used the fact that arrival process is Bernoulli i.i.d. From hypothesis of Theorem 1,

$$\lambda \quad \leq \quad \left(\sum_{k=1}^{n^2}\alpha_k\pi_k\right), \tag{9}$$

where for all k, $\pi_k \in \mathbb{P}$, $\alpha_k \in \mathbb{R}_+$ and $\sum_k \alpha_k = 1-2\varepsilon$. By the property of the ε-Auction algorithm,

$$\mathbf{Q}(\tau)\cdot\pi(\tau+1) \quad \geq \quad (1-\varepsilon)W^*(\tau), \tag{10}$$

where $W^*(\tau) = \max_{\pi\in\mathbb{P}}\mathbf{Q}(\tau)\cdot\pi$. Putting the above discussion together, we have

$$\mathbb{E}\left[L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau))|\mathbf{Q}(\tau)\right] \leq -2\varepsilon W^*(\tau) + 2n.$$

It is not difficult to see that

$$\mathbf{Q}(\tau)\cdot\frac{1}{n}\mathbf{1} = \frac{1}{n}\|\mathbf{Q}(\tau)\|_1 \leq W^*(\tau).$$

Thus, we have

$$\mathbb{E}\left[L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau))|\mathbf{Q}(\tau)\right] \leq -\frac{2\varepsilon}{n}\|\mathbf{Q}(\tau)\|_1 + 2n.$$

Thus, from Foster's criteria as stated earlier, we obtain that

$$\limsup_{\tau\to\infty}\mathbb{E}[Q_{ij}(\tau)] < \infty, \quad \forall\ i,\ j.$$

Now, we prove the claimed bound on the average queue-size. Consider the following that follows from above.

$$\begin{aligned}\mathbb{E}[L(\mathbf{Q}(\tau+1))] &= \mathbb{E}\left[\mathbb{E}[L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau))|\mathbf{Q}(\tau)]\right] \\ &\quad + \mathbb{E}[L(\mathbf{Q}(\tau))] \\ &\leq \mathbb{E}[L(\mathbf{Q}(\tau))] - \frac{2\varepsilon}{n}\mathbb{E}\left[\|\mathbf{Q}(\tau)\|_1\right] + 2n.\end{aligned}$$

By telescoping the above for $\tau = 0,\ldots,T-1$ and recalling $\mathbb{E}[L(\mathbf{Q}(0))] = 0$, we obtain

$$\frac{1}{T}\mathbb{E}[L(\mathbf{Q}(T))] \quad \leq \quad -\frac{2\varepsilon}{n}\mathbb{E}\left[\frac{1}{T}\sum_{\tau=0}^{T-1}\|\mathbf{Q}(\tau)\|_1\right] + 2n. \tag{11}$$

By definition $\mathbb{E}[L(\mathbf{Q}(T))] \geq 0$. Hence,

$$\limsup_{T\to\infty}\mathbb{E}\left[\frac{1}{T}\sum_{\tau=0}^{T-1}\|\mathbf{Q}(\tau)\|_1\right] \quad \leq \quad n^2/\varepsilon. \tag{12}$$

As established by Foster's criteria earlier on, the queue-size process $\mathbf{Q}(\tau)$ is positive-Harris recurrent under the ε-Auction algorithm as long as λ satisfies the hypothesis of Theorem 1. Thus $\mathbf{Q}(\cdot)$ is an irreducible, aperiodic Markov chain and hence ergodic. That is, $\mathbf{Q}(\tau) \to \mathbf{Q}(\infty)$ where $\mathbf{Q}(\infty)$ follows the stationary distribution of this Markov chain. Now, the following completes the proof.

$$\begin{aligned}\limsup_{\tau\to\infty}\mathbb{E}\left[\|\mathbf{Q}(\tau)\|_1\right] &\leq \mathbb{E}\left[\limsup_{\tau\to\infty}\|\mathbf{Q}(\tau)\|_1\right] \\ &= \mathbb{E}\left[\|\mathbf{Q}(\infty)\|_1\right] \\ &= \mathbb{E}\left[\liminf_{T\to\infty}\frac{1}{T}\sum_{\tau=0}^{T-1}\|\mathbf{Q}(\tau)\|_1\right] \\ &\leq \liminf_{T\to\infty}\mathbb{E}\left[\frac{1}{T}\sum_{\tau=0}^{T-1}\|\mathbf{Q}(\tau)\|_1\right] \\ &\leq n^2/\varepsilon.\end{aligned}$$

In the above we have used the ergodic theorem and Fatou's lemma. ∎

D. Switch with Finite Buffers

The above sections establish that the ε-Auction algorithm is *almost* throughput maximal, takes $O(n^2/\varepsilon)$ iterations to find a matching and induces $O(n^2/\varepsilon)$ net average queue-size. This analysis assumed the standard idealized infinite buffer switch. In practice, a switch always has finite buffers. However, due to technical limitations all the known analysis has been restricted to the infinite buffer case. While the infinite buffer analysis may provide an inkling on how well a finite buffer switch may behave, it is far from being satisfactory.

For the very same reason discussed above, we are unable to deal with the precise analysis of finite buffered switches here. However, we discuss heuristics based on auction algorithm that allows for a tradeoff between performance and number of iterations algorithm needs to run. Specifically, let B be the buffer size of the switch for any VOQ. Given parameter N, set $\delta = B/N$ instead of $\varepsilon Q^*/n$ in the ε-Auction algorithm. Call this algorithm N-Auction algorithm.

Along the lines of the proof of ε-Auction algorithm, it is clear that the above algorithm will take $O(nN)$ iterations to converge. The weight of the matching found by the algorithm will be no less than nB/N amount. This will naturally affect the performance of the algorithm: as N increases, the algorithm takes longer to converge but quality of solution is expected to become better and hence the performance of algorithm is expected to be better.

E. Auction with Memory

In this section we look at the auction algorithm with memory. Consider the following slight modification of the ε-Auction algorithm from section II-B. At any time slot $\tau+1$ the parameter p_j for $1 \leq j \leq n$ instead of being initiated with zero starts with its final value from the time slot τ.

The intuition behind this modification is the following. At the end of the time slot τ the parameters p_j are optimal for the queue sizes $Q_{ij}(\tau-1)$. This means for each input i:

$$Q_{i\pi^*(i)}(\tau-1) - p_{\pi^*(i)} = \max_{j=1}^{n} Q_{ij}(\tau-1) - p_j$$

At time slot $\tau+1$ the queue sizes do not vary much since each can receive or transmit at most one packet. Hence one

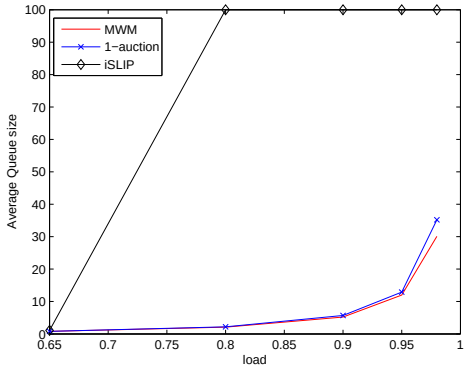

Fig. 3. Average queue sizes for MWM, 1-Auction and iSLIP.

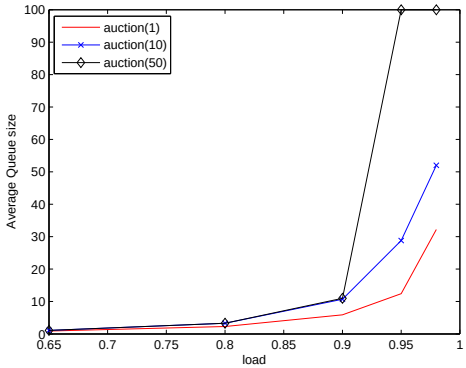

Fig. 5. Average queue sizes for auction(1), auction(10), auction (50)

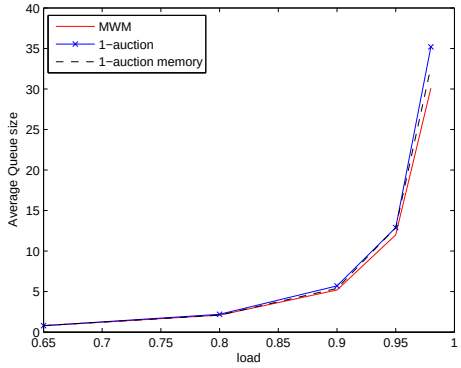

Fig. 4. Average queue sizes MWM, 1-Auction and 1-Auction with memory.

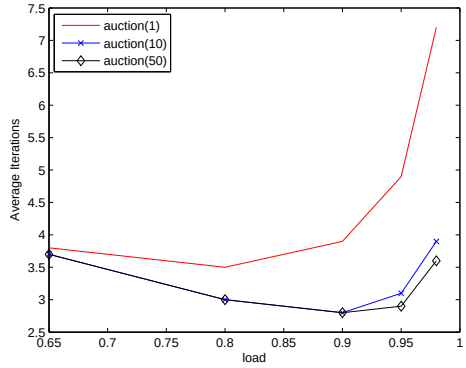

Fig. 6. Average iterations to converge for auction(1), auction(10), auction(50)

expects the parameters p_j to be near optimal for queue sizes $Q_{ij}(\tau)$. From [3] we expect that in time slot $\tau+1$ starting with values of p_j from the time slot τ the number of iterations for convergence of the algorithm to be relatively small. Simulation results of the next section support this intuition as well.

F. Representative Simulation Results

In this section we provide simulation results for an 8×8 input queued switch with a non-uniform admissible arrival matrix. The traffic load takes one of the values from the set $\{.65, 8, .9, .95, .98\}$. All simulations are done for one million time slots. In this section "auction(c)" denotes the auction algorithm with $\delta = c$ where c is a constant. For the ε-Auction algorithm we use $\varepsilon = 1$ and hence denote it by 1-Auction. When the number of iterations of the iSLIP algorithm is not mentioned it is understood to have run all the way to the end, i.e. it runs $n = 8$ iterations.

Figure 3 shows that the 1-Auction algorithm performs much better than the iSLIP algorithm and is as good as MWM. The next plot, Figure 4, shows that 1-Auction with memory has better performance than 1-Auction.

Figures 5 and 6 show the trade-off that was referred to in section II-D. Here the value $\delta = B/N$ takes one of the values $1, 10, 50$. As mentioned before larger values of δ yield less number of iterations but at the expense of greater queue sizes. Figure 7 shows a comparison between 1-Auction and iSLIP when both run only three iterations in each time slot. In practice sometimes only a few iterations of the iSLIP algorithm are used instead of the full iSLIP. This figure shows that the 1-Auction algorithm can also be used for a fewer number of iterations and it still outperforms iSLIP.

III. Scheduling for Wireless Networks

In this section, we describe a simple, iterative, distributed scheduling algorithm for a multi-hop wireless network, which is modeled as a grid-graph as mentioned earlier in this paper. First, we introduce the specific model.

A. Setup

Consider a grid-graph on n nodes with V denoting the vertex-set and E denoting its edge-set. Let $\mathcal{N}(i)$ denote the neighbors of the node i. For $j \in \mathcal{N}(i)$, let $Q_{ij}(\tau)$ denote the queue-size corresponding to the packets waiting at node i to go to node j at time τ. The matching constraints require that at any given time each node i can either transmit to or receive from at most one other node. As before, let $A_{ij}(\tau)$ denote the cumulative arrival process corresponding to packets arriving at i and going to j till time τ. Again, we assume that arrival

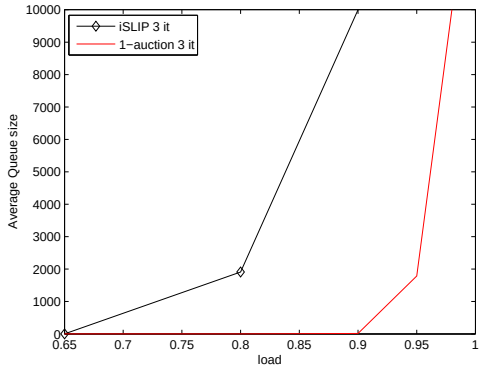

Fig. 7. Average queue sizes for iSLIP with 3 iterations and 1-Auction with 3 iterations

process is 'friendly', i.e. it is a Bernoulli i.i.d. process. Given the constraints, the *admissible* arrival rate-matrix $\lambda = [\lambda_{ij}]$ must belong to the convex hull of the rates induced by the set of matchings of grid graph (an edge (i,j) in matching can induce unit rate along $i \to j$ or $j \to i$ and we are interested in the convex hull of all such possibilities).

In this setup, the scheduling algorithm corresponds to first finding a matching in the graph and then for each edge in the matching deciding which node transmits and which node will receive data. Again, the result of Tassiulas and Ephremides [15] immediately implies that the following max-weight scheduling algorithm is throughput optimal: (i) choose maximum weight matching with the weight of an edge (i,j), denoted by w_{ij}, being $\max\{Q_{ij}, Q_{ji}\}$; (ii) for an edge (i,j) in the chosen matching, node i transmits to node j if $Q_{ij} \geq Q_{ji}$ (ties broken arbitrarily).

Thus, it is desirable to find the MWM in the graph with edge weights as described above. In case of a switch, the graph was bipartite. The wireless network, modeled as grid graph (generally any such 'product graph' in d dimensions) is also bipartite, i.e. the graph nodes can be divided into two parts so that edges are only between nodes of the different partitions. This is because these graphs do not have cycles of odd length. Thus, this situation is the same as that of switch setup. The only difference is as follows: in a switch, the partition of nodes is known in terms of inputs and outputs. In case of a grid-graph, it is not known a priori. An ad-hoc fix to this situation is that nodes co-operate and form a bipartition and use the above described auction algorithm for finding an optimal schedule. A more natural and scalable approach is to have an algorithm that *does not* require prior knowledge of the bipartition. That is, we need an algorithm, which is 'symmetric', unlike the standard auction, and that does not treat nodes of the two partitions differently. Next, we describe such an algorithm based on the Max-Product Belief propagation algorithm.

B. Symmetric Auction via Max-Product

The following algorithm is an adaption of the Min-Sum (a version of Max-Product) algorithm described in [29] that operates very similarly to the auction algorithm.

ε-Min-Sum Algorithm

- Let $Q^* = \max_{ij} Q_{ij}$, which can be quickly computed in a distributed manner.
- Set $\delta = \varepsilon Q^*/n$.
- Given queue-size matrix $\mathbf{Q}$, define a symmetric weight matrix $\mathbf{W} = [W_{ij}]$ as follows: for all $(i,j) \notin E$, set $W_{ij} = 0$ and for all $(i,j) \in E$ set $W_{ij} = \max\{Q_{ij}, Q_{ji}\} + \delta_{ij}$. Where δ_{ij} is a randomly chosen number from the interval $(0, \delta)$ and can be selected by one communication between i, j.
- The algorithm variables are message that are exchanged between neighboring nodes. Let $\hat{m}^k_{i\to j} \in \mathbb{R}$ denote message from node i to node j in iteration k.
- Initialize $k = 0$ and set the messages as follows:

$$\hat{m}^0_{i\to j} = W_{ij}; \quad \hat{m}^0_{j\to i} = W_{ij}$$

- For $k \geq 1$, iterate as follows:
  (a) Update messages as follows:

$$\begin{aligned} \hat{m}^k_{\alpha_i\to\beta_j} &= W_{ij} - \max_{\ell\neq j} \hat{m}^{k-1}_{\beta_\ell\to\alpha_i}, \\ \hat{m}^k_{\beta_j\to\alpha_i} &= W_{ij} - \max_{\ell\neq i} \hat{m}^{k-1}_{\alpha_\ell\to\beta_j}. \end{aligned} \quad (13)$$

  (b) The estimated MWM at the end of iteration k is π^k, where $\pi^k(i) = \arg\max_{j\in\mathcal{N}(i)}\{\hat{m}^k_{\beta_j\to\alpha_i}\}$ for $1 \leq i \leq n$. But when $\max_{j\in\mathcal{N}(i)}\{\hat{m}^k_{\beta_j\to\alpha_i}\} < 0$ then let $\pi^k(i) =$ "null" which means node i chooses not to connect to any of its neighbors.
  (c) Repeat (a)-(b) till $\pi^k(i)$ converges, i.e. for each $1 \leq i \leq n$, $\pi^k(\pi^k(i)) = i$ or $\pi^k(i) =$ "null" for all k large enough.

We will show that the above algorithm converges to the MWM with probability one in finite number of iterations.

1) Analysis of the ε-Min-Sum Algorithm: The ε-min-sum algorithm described above is a minor variant of the simplified min-sum algorithm described in [29][1]. In fact there are two main differences between the two algorithm: i) In the simplified min-sum algorithm at every iteration each node is matched, but in the ε-min-sum the nodes have the option of remaining unmatched. ii) In the ε-Auction algorithm, the parameter ε gives us a trade-off between converging to a good matching versus a fewer number of iterations to converge.

Let $\mathbf{Q}' = [Q'_{ij}]$ be a symmetric matrix of queue seizes defined by $Q'_{ij} = \max\{Q_{ij}, Q_{ji}\}$. Also, let π^* denote the MWM of matrix $\mathbf{Q}'$ and let W^* denote weight of π^*. We will prove the following result.

Theorem 3 *Given $\varepsilon > 0$, with probability one the algorithm ε-min-sum will converge to a matching with weight at least $W^* - \varepsilon Q^*$.*

[1]A longer version of this paper can be found at http://www.stanford.edu/∼bayati/papers/mpmwm.ps

Proof: Consider the set of weights $\mathbf{W}$. Let $\hat{\pi}$ be the MWM of the matrix $\mathbf{W}$ and let $\hat{W}$ denotes its weight. Note that $W^* \leq \hat{W}$ since for all i, j we have $Q'_{ij} \leq W_{ij}$. Hence

$$\begin{aligned} \sum_i Q'_{i\hat{\pi}(i)} &= \sum_i \left(W_{i\hat{\pi}(i)} - \delta_{i\hat{\pi}(i)}\right) \\ &\geq \hat{W} - n\delta \\ &\geq W^* - \varepsilon Q^*. \end{aligned}$$

Now we only need to show that the algorithm ε-min-sum converges to the matching $\hat{\pi}$ with probability one.

Note that since the numbers δ_{ij} are chosen randomly from the interval $(0, \delta)$, with probability one $\hat{\pi}$ is unique. Now all we need is to show that if $\hat{\pi}$ is unique then ε-min-sum algorithm will converge to it. The proof of this fact is very similar to the proof of Theorem 1 in [29]. Here we summarize the main steps of the proof: i) Consider the min-sum algorithm defined in sections I.A and I.B of [29]. For each vertex add another state which corresponds to being unmatched and update the compatibility functions accordingly. Hence each message and belief vector will be in $\mathbb{R}^{(n+1)}$. ii) Apply the same procedure as in the proof of Lemma 2 in [34] to show that the min-sum algorithm finds the MWM as long as it is unique. iii) Similar to the proof of Lemma 2 in [29] each message vector of the min-sum algorithm still has two distinct values with the coordinate corresponding to the new state being equal to the $n-1$ remaining coordinates. iv) Subtract the two distinct values of each message from each other to obtain equations (13) of the ε-min-sum algorithm described above. Moreover by the same argument as in Lemma 2 of [29] this algorithm will be equivalent to the min-sum algorithm with additional state. This will finish the proof. ■

Theorem 3 shows that the algorithm ε-min-sum converges, but it does not provide any bound on the number of iterations required for convergence. From the main theorem in [34] we know that the ε-min-sum algorithm converges in at most $\mathrm{O}(nW_{\max}/\gamma)$ iterations where $W_{\max} = \max_{ij}\{W_{ij}\}$ and γ is the difference between the weight of MWM and second MWM in the matrix $\boldsymbol{\delta} = [\delta_{ij}]$. Therefore using $\delta = \varepsilon Q^*/n$ the expected number of iterations for convergence of the ε-min-sum algorithm is expected to be at most $\mathrm{O}\big(n^2/(\varepsilon\rho)\big)$ iterations where ρ is the maximum expected gap between the weight of the MWM and second MWM of an $n \times n$ matrix whose entries are i.i.d. random variables from the interval $(0, 1)$. Simulation results suggest that $\rho \approx \frac{1}{n \log n}$ for the uniform distribution. Hence expected number of iterations for convergence of the ε-min-sum algorithm will be at most $\mathrm{O}(n^3 \log n/\varepsilon)$.

2) Simulation results for the ε-Min-Sum Algorithm: In section III-B.1 it was shown that similar to the ε-Auction algorithm the ε-min-sum algorithm finds matchings that have weight very close to the weight of the MWM. This means the ε-min-sum algorithm is throughput optimal and has very low delay. We have simulated this algorithm for an 8×8 input queued switch and compared it with the ε-Auction algorithm and MWM. The result as is shown in Figures 8 and 9 indicates that the ε-min-sum algorithm gives lower delay than the ε-Auction algorithm at the expense of more convergence time. For this simulation we assume $\varepsilon = 1$.

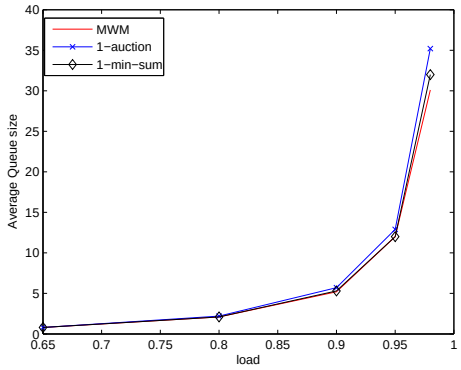

Fig. 8. Average queue sizes for MWM, 1-Auction and 1-min-sum

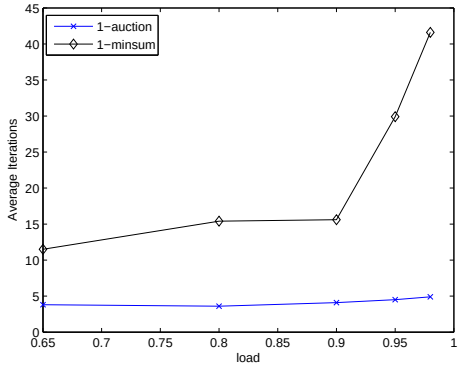

Fig. 9. Average iterations to converge for 1-Auction and 1-min-sum

IV. Conclusion

In this paper we introduced two simple, iterative, distributed algorithms for switch scheduling and wireless network applications. Iterative algorithms are powerful solutions for such problems because their iterative and distributed nature allow for many implementational simplifications (such as a pipelined architecture and eliminating centralized schedulers), taking them from the realm of theoretical constructs towards possible commercial deployment.

We introduced the ε-Auction algorithm and the ε-min-sum algorithm which are both iterative and very simple. They are new variations of the auction and the min-sum algorithm respectively. We have shown that the ε-Auction and the ε-min-sum algorithm solve the problem of finding the MWM for a switch/network. Therefore, both these algorithms are throughput and delay optimal which makes them most suited for scheduling purposes. We also show (via simulations) that these algorithms possess a very attractive feature that even when run for a few number of iterations, their throughput and delay properties are better than the current algorithms used in practice.

We believe that our algorithms represent an effective and, in some aspects, radical line of attack on the traditional problem of scheduling. Further work is required to establish the auxiliary properties and to better understand this family of iterative scheduling algorithms.

REFERENCES

[1] N. McKeown, "iSLIP: a scheduling algorithm for input-queued switches," *IEEE Transaction on Networking*, vol. 7, no. 2, pp. 188–201, 1999.

[2] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, 1988.

[3] ——, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, vol. 1, pp. 7–66, 1992.

[4] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann, 1988.

[5] J. Yedidia, W. Freeman, and Y. Weiss, "Generalized belief propagation," *Mitsubishi Elect. Res. Lab.*, vol. TR-2000-26, 2000.

[6] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *Submitted, available through* `csl.uiuc.edu/rsrikant`, 2006.

[7] X. Wu, R. Srikant, and J. R. Perkins, "Queue-length stability of maximal greedy schedules in wireless networks," in *Workshop on Information Theory and Applications, UCSD*, 2006.

[8] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks," in *43rd Allerton conference on Comm. Control and computing*, 2005.

[9] X. Lin and N. B. Shroff, "Impact of imperfect scheduling in wireless networks," in *IEEE INFOCOM*, 2005.

[10] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space division packet switch," *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1347–1356, 1987.

[11] Y. Tamir and H. Chi, "Symmetric crossbar arbiters for vlsi communication switches," *IEEE Transaction on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 13–27, 1993.

[12] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Transactions on Computer Systems*, vol. 11, pp. 319–351, 1993.

[13] M. Karol, K. Eng, and H. Obara, "Improving the performance of input-queued atm packet switch," in *IEEE INFOCOM*, 1992, pp. 110–115.

[14] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proceedings of IEEE Infocom*, 1996, pp. 296–302.

[15] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1936–1948, 1992.

[16] J. Dai and B. Prabhakar, "The throughput of switches with and without speed-up," in *Proceedings of IEEE Infocom*, 2000, pp. 556–564.

[17] D. Shah, "Stable algorithms for input queued switches," in *Proceedings of Allerton Conference on Communication, Control and Computing*, 2001. [Online]. Available: http://www.stanford.edu/~devavrat/ilqf.ps

[18] I. Keslassy and N. McKeown, "Analysis of scheduling algorithms that provide 100% throughput in input-queued switches," in *Proceedings of Allerton Conference on Communication, Control and Computing*, 2001.

[19] D. Shah and M. Kopikare, "Delay bounds for the approximate Maximum Weight matching algorithm for input queued switches," in *Proceedings of IEEE Infocom*, 2002.

[20] D. Shah and D. J. Wischik, "An optimal scheduling algorithm for input queued switch," in *IEEE INFOCOM*, 2006.

[21] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of ACM*, vol. 18, pp. 264–284, 1972.

[22] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *IEEE INFOCOM*, vol. 2, 1998, pp. 533–539.

[23] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *IEEE Journal on Selected Areas in Communications High-performance electronic switches/routers for high-speed internet*, vol. 21, no. 4, pp. 546–559, 2003.

[24] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," in *ACM SIGMETRICS/Performance*, 2006.

[25] R. G. Gallager, *Low Density Parity Check Codes*. Monograph, MIT Press, 1963.

[26] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," vol. 47, pp. 599–618, 2001.

[27] A. Braunstein, M. Mezard, and R. Zecchina, "Survey propagation: an algorithm for satisfiability," *Random Structures and Algorithms*, vol. 27, pp. 201–226, 2005.

[28] M. Bayati, D. Shah, and M. Sharma, "Maximum weight matching via max-product belief propagation," *Preliminary version appeared at IEEE ISIT 2005. Longer version* Submitted *and available at http://www.stanford.edu/∼bayati/papers/mpmwm.ps*, 2005.

[29] ——, "A simpler max-product maximum weight matching algorithm and the auction algorithm," in *IEEE Int. Symp. Information Theory*, 2006.

[30] P. R. Kumar and S. P. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, pp. 251–260, 1995.

[31] S. Asmussen, *Applied Probability and Queues*. New York: Wiley, 1987.

[32] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. Springer-Verlag, London, 1993.

[33] ——, "Stability of markovian processes ii: Continuous time processes and sampled chains," *Advances of Applied Probability*, vol. 25, pp. 487–517, 1993.

[34] M. Bayati, D. Shah, and M. Sharma, "Maximum weight matching via max-product belief propagation," in *EEE Int. Symp. Information Theory*, 2005.