EE384Y Project Intermediate Report
Approximate Maximal Matchings and required speedup for 100% throughput

Abhishek Das
John J Kim

Table of Contents:

Background and Motivation – Page 2
Problem Statement – Page 3
Current Status and Future Work - Page 3-6
Results
Conclusions
Background and Motivation:

In order to achieve 100% throughput, the switch needs to be able to transfer a large number of cells in every slot. In order to connect a large number of inputs and outputs in any one slot:

- The scheduler should find a large-sized match for the current request graph.
- The request graph should be dense i.e the presence of queued cells at as many inputs and for as many outputs as possible, to enable matching multiple input-output pairs (to transfer many cells simultaneously).

Considering the queue occupancy as weights, we have learnt 3 very important results for switch scheduling algorithms in the course:

1. Without speedup, maximum weight matching can achieve 100% throughput for IQ switches [Achieving 100% Throughput in an Input-Queued Switch, Nick McKeown, Adisak Mekkittikul, Venkat Anantharam and Jean Walrand]
2. With speedup 2, any maximal matching can achieve 100% throughput for CIOQ switches [The Throughput of Data Switches with and without Speedup, J.G. Dai and B. Prabhakar]
3. With speedup 2, a CIOQ switch can emulate a OQ switch [Matching Output Queueing with a Combined Input Output Queued Switch, Shang-Tse Chuang, Ashish Goel, Nick McKeown, Balaji Prabhakar]

Results 2 and 3 motivate us to look for speedup s < 2, such that 100% throughput (stability) can be achieved for CIOQ switches. Since one of the main goals in designing a switch scheduling algorithm is to pick a simple algorithm, results 1 and 2 motivate us to look for maximal matching algorithms, which are k-approximation to the maximum weight matching (eg. iLQF gives a 2-approximation maximal matching). A previous work [On Achieving Throughput in an Input-Queued Switch, Saad Mneimneh and Kei-Yeung Siu] on the same topic used adversarial traffic to come up with lower bounds on the speedup required for weak throughput. However, they created the adversarial by introducing correlation amongst input arrivals. We will be looking at independent and admissible input arrivals.

The idea of k-approximation matching algorithms, along with result 2, motivate us further to look at modifying iterative algorithms like PIM, iLQF etc. As we know from practice, these algorithms need to be stopped after c iterations, which result in sub-maximal matchings. A closer look at result 2 reveals that even the maximal matching with the minimum size (or weight) amongst all maximal matchings is stable at speedup 2. We wonder if stability can be guaranteed by sub-maximal matchings of size (or weight) atleast k times larger than the maximal matching with minimum size (or weight). If so, at what speedup (s) and with how many iterations (c).

However, accounting for weights adds a lot of hardware complexity. Algorithms like iSLIP [Scheduling algorithms for input-queued cell switches] and PIM, which don’t use weights, are found to achieve good performance within 4 iterations. This motivates us to
explore practical matching algorithms that converge in a smaller number of iterations.
Our approach will be to increase the size of matching found in every iteration.

**Problem Statement:**
We want to explore the possibilities of achieving 100% throughput using k-
approximation maximal matchings, with speedup s < 2. This will involve exploring new
Lyapunov functions to account for the weight of the matchings. We also intend to look at
such Lyapunov functions (accounting weight of matchings) to work our way backwards
and design the matching algorithm, which may or may not be a maximal matching.
We want to explore fair iterative algorithms producing sub-maximal matchings in c
iterations, which account for the weight (or size) of the matching and guarantee stability
at speedup s.
We will also like to get some more insight into the existing iterative algorithms like PIM
to understand their interaction with VOQ sizes and packet arrivals. We believe this will
lead us to designing better matching algorithms and will affect the previous goal
mentioned.

**Current Status and Future work:**
We first show the progress we made in analyzing the stability of k-approximate
maximum weighted matching.

Using queue lengths as fluids, we know the following fluid equations for t ≥ 0;
1. \( Z_{ij}(t) = Z_{ij}(0) + \lambda_{ij}(t) - D_{ij}(t) \geq 0 \)
2. \( D_{ij}(t) = \pi_{ij} w I\{Z_{ij}(t) > 0\} \)

We know, for any admissible load matrix \( \lambda, \langle \lambda, Z(t) \rangle \leq W^* \), where \( W^* \) is the weight of the
maximum weighted matching. This can be shown using Birkhoff-von Neumann
decomposition.

Let \( w \) denote the weight of the matching algorithm being used i.e \( \langle \pi_{ij} w, Z(t) \rangle = w \)
Taking quadratic Lyapunov function: \( L(t) = \langle Z(t), Z(t) \rangle \)
\( L'(t) = 2\langle Z(t), Z(t) \rangle \)
\( = 2[\langle \lambda(t), Z(t) \rangle - \langle D(t), Z(t) \rangle] \)
\( \leq 2[W^* - \langle D(t), Z(t) \rangle] \)

Now, using speedup s, \( \langle D(t), Z(t) \rangle = \langle s.\pi_{ij} w, Z(t) \rangle \)
\( L(t) \leq 2[W^* - \langle s.\pi_{ij} w, Z(t) \rangle] \)
\( = 2[W^* - s.w] \)
\( \leq 0 \) if \( W^* \leq s.w \)

Now for a k-approximation matching algorithm, \( k.w \geq W^* \)
Hence, if \( k \leq s \), we satisfy \( W^* \leq k.w \leq s.w \) and achieve 100% throughput.

**Result 1:** In order to run at speedup s, using a k-approximate maximum weight matching
is sufficient to provide 100% throughput if \( k \leq s \).
We are currently unaware of many practical approximation maximum weighted matching algorithms. There are many known algorithms for \(k=2\) eg. the greedy iLQF. Other known work is in *Distributed algorithm for better approximation of the maximum matching*, A. Czygrinow and M. Hanckowiak, which finds \(k\)-approximate matchings for graphs that don’t have odd cycles.

Let’s now look at size of matching algorithms since we know considering weights add a lot of hardware complexity.

Taking Linear Lyapunov function: \(L(t) = \langle Z(t), 1 \rangle\)
\[
L'(t) = \langle Z'(t), 1 \rangle = \langle \lambda(t), 1 \rangle - \langle D(t), 1 \rangle
\]

Now, since we consider admissible traffic i.e \(\sum_{i} \lambda_{ij} \leq 1\) and \(\sum_{j} \lambda_{ij} \leq 1\)

Thus, \(\langle \lambda(t), 1 \rangle \leq N\)

Also, \(D(t), 1 \rangle \geq \langle \pi_{ij}^w, 1 \rangle\) since \(D_{ij}(t) = \pi_{ij}^w I_{\{Z_{ij}(t)>0\}}\)

Combining, at speedup \(s\), we get
\[
L(t) \leq N - \langle s \cdot \pi_{ij}^w, 1 \rangle = N - s|M|, \text{ where } |M| \text{ is the size of the matching}
\]

\[\leq 0\]

if \(s|M| \geq N\), thus achieving 100% throughput.

**Result 2:** In order to run at speedup \(s\), a matching with size \(|M| \geq N/s\) is sufficient to provide 100% throughput.

Note that the sub-maximal matching algorithm PIM [High-Speed Switch Scheduling for Local-Area Networks, T. Anderson, S. Owicki, J. Saxe, and C. Thacker] was shown to generate a matching of size \(3R/4\) on average, in the first iteration, where \(R\) is the number of inputs requesting a match.

This leads us to our next work, which focused on practical iterative matching algorithms, producing sub-maximal matches. In order to achieve a better matching in an iterative algorithm such as PIM, we need to achieve a “better” matching – e.g. increase the chances of grants of being accepted. In PIM, grants are chosen randomly and among the grants, the input selects one of the grants randomly and accepts it. To improve on this selection process, we propose HIM (Head-of-line parallel Iterative Matching) where each input request is assigned weights based on the fanout of the input in the bipartite graph. The fanout corresponds to the number of VOQ’s in the input which has a packet at the head of the line. The input with the smallest fanout is assigned the highest weight since it will have the best chance of accepting the grants. Thus, the input with the highest fanout will have the smallest weight since it has the highest chance of receiving multiple grants. Note that the hardware complexity to obtain the weights is much simpler than other weighted algorithms.
Based on the weights described above, there are several different ways to implement the HIM algorithm:

1) output grants the input with the highest weight – with the ties being broken randomly
2) output grants the input based on the weighted probability based on the weights above

Although (1) option is simpler, it can result in unfairness on certain traffic patterns where few inputs have all of their VOQ HOL occupied and the other inputs only have a few VOQ occupied. The heavily occupied inputs will likely be starved with this approach. With the (2) option, the smaller fanout will be favored but the randomness added in selecting the inputs based on the weights prevents starvation from occurring.

The algorithm has been implemented on the SIM simulator and under uniform traffic, this modification was shown to have some improvement in the throughput for 1 iteration of the PIM algorithm.

![Figure 1: Latency vs Offered Load for the Different Iterative Algorithm](image)

Although the performance of the PIM improves with this modification, it does not achieve the goal of either bounding the number of iteration or achieving stability with a speedup of s. Since iSLIP is known to achieve 100% throughput for uniform Bernoulli IID traffic, in 1 iteration, can these weights used in HIM be applied to iSLIP to give us a better matching? From figure 3.4 of [2], at high loads the number of iteration required drops for iSLIP because of the characteristics of the iSLIP algorithm and the way it updates its pointer. However, between 60% and 90% loads, the number of iterations steadily increases, which is caused by sub-maximal matching as shown in figure 3.6[2]. By accounting for these weights, the expectation is that the size of the matching will increase, thus able to bound the number of iterations required. From Figure 1, at
intermediate loads, the HIM has lower latency and combining the load with the iSLIP, we hope to show there is an improvement in the matching of iSLIP. However, it is unknown how this will effect the synchronization between the different points within iSLIP but at high loads where the synchronization matters, the HIM weights should be identical since the fanout of most of the inputs will be N.

Work left to do:

1) After verifying and implementing the algorithm above, collect results and simulate with non-uniform random traffic.
2) It has been shown that the average number of iterations required for the iterative algorithms to converge is \( \log(n) \) [1] With the modifications to the iterative algorithms, can we show that there is a bound on the number of iterations required?
3) In addition to the number of iterations, if speedup is added to iterative algorithms, can we obtain stability, even in sub-maximal matching? We plan to simulate the iterative algorithm with various speedup and try to combine this simulation with the theoretical results from the earlier part.
