Stability Analysis of MNCM Class of Algorithms
And Two More Problems!

EE384Y Project Final Report
By Nima Asgharbeygi
June 6, 2003

Table of Contents

1 Background and Motivation

In a recent work presented in INFOCOM 2003, Tabatabaee and Tassiulas [1] introduced a new class of scheduling algorithms for IQ switches with no speedup, called maximum node containing matching (MNCM). MNCM class is defined as below:

Definition: A maximal size matching algorithm *m* belongs to MNCM class if and only if *m* contains all nodes with maximum weight.

Weight of a node is defined to be the summation of its corresponding VOQ occupancies. We number ports of the switch from 1 to 2N where index sets {1,…,N} and {N+1,…,2N} corresponds to input and output ports respectively. Then the port weights are,

$$B_k(n) = \sum_{(i,j):(i,j)\to k} Z_{ij}(n) \tag{1}$$

Where $k \in \{1,...,2N\}$ and the notation $(i,j) \to k$ means link (i,j) is associated with port *k.* They also extended the fluid model for the links of an input buffered switch given in [3] to derive the port based fluid model. Then they used function $f(B(t))$ defined from $R^{2N} \to R$ as,

$$f(\underline{B}(t)) = \max\{B_1(t),...,B_{2N}(t)\} \tag{2}$$

to be the Lyapunov function, proved that the port based fluid model is weakly stable and concluded that MNCM class of algorithms is efficient under general traffic (satisfying law of large numbers).

MNCM is actually very interesting in the sense that it includes LPF algorithm (because it finds a min-max matching) with complexity $O(N^3)$, as well as simpler algorithms like MFM with complexity $O(N^{2.5})$.

But a recent counter example invalidated efficiency of MNCM under general traffic!

2 Problem Statement

The first problem is to examine the performance of MNCM algorithms with a discrete model, i.e. assuming IID Bernoulli arrivals.

I would also like to realize the issues with the fluid model proof which result in an invalid conclusion.

After this, I'll try to find another generalization on LPF algorithm; a class of efficient algorithms containing LPF. This might help deriving some statements about delay performance and behavior of iterative algorithms such as iLPF and maximum sorted matching (MSM).

3 Stability Results for MNCM

3.1 General Traffic

A counter-example produced by Da Chuang showed that MNCM class of algorithms is not as efficient as claimed in [1]. This is a periodic deterministic arrival pattern which makes the queue occupancies grow indefinitely. One period of the arrival pattern is shown in figure 1.

Matchings identified by underlies are obviously in MNCM. The same scheduling policy will apply to later periods of traffic pattern because all the node weights are equal at the end of each period. Therefore at the end of time slot 6*n,* the occupancy of queues will be,

$$\begin{pmatrix} n & n & 0 \\ n & 0 & n \\ 0 & n & n \end{pmatrix}$$

Time Slot		1	2	3	4	5	6
Arrival		1 0 0	0 1 0	1 0 0	0 1 0	0 1 0	1 0 0
Pattern		1 0 0	0 0 1	1 0 0	1 0 0	0 0 1	0 0 1
		0 1 0	0 1 0	0 0 1	0 0 1	0 1 0	0 0 1
Queues		1 0 0	1 $\underline{1}$ 0	$\underline{2}$ 0 0	1 1 0	1 $\underline{2}$ 0	$\underline{2}$ 1 0
Before		$\underline{1}$ 0 0	0 0 $\underline{1}$	1 0 0	$\underline{2}$ 0 0	1 0 $\underline{1}$	1 0 1
Departure		0 $\underline{1}$ 0	0 1 0	0 1 $\underline{1}$	0 $\underline{1}$ 1	0 1 1	0 1 $\underline{2}$
Queues	0 0 0	1 0 0	1 0 0	1 0 0	1 1 0	1 1 0	1 1 0
After	0 0 0	0 0 0	0 0 0	1 0 0	1 0 0	1 0 0	1 0 1
Departure	0 0 0	0 0 0	0 1 0	0 1 0	0 0 1	0 1 1	0 1 1

Figure 1 Counter-example details.

3.2 IID Bernoulli

I tried Lyapunov approach to get some results for discrete case. But while trying to upper-bound the increase in Lyapunov function $(f(\underline{B}(n+1)) - f(\underline{B}(n)))$, I realized the inherent weakness in the definition of MNCM. So I moved toward simulations and implemented MFM algorithm using MATLAB. The code simply performs the following:

1. Find a maximum size matching M_1 on the graph induced by all input nodes with max weight (this will contain all max-weight input nodes).
2. Find a maximum size matching M_2 on the graph induced by all output nodes with max weight (this will contain all max-weight output nodes).
3. Combine M_1 and M_2 to find a matching M containing all max-weight nodes.
4. Add remaining possible matches to M to make it maximal.

My simulations show that MFM doesn't achieve 100% throughput even for IID Bernoulli arrival. Figure 2 is a plot of delay performance for a 4×4 switch with uniform IID Bernoulli arrival operating under MFM scheduling algorithm. The length of simulation

was 100,000 time slots. The algorithm is certainly unstable for an arrival rate of $\rho = 0.8$ (average backlog increases linearly over time).

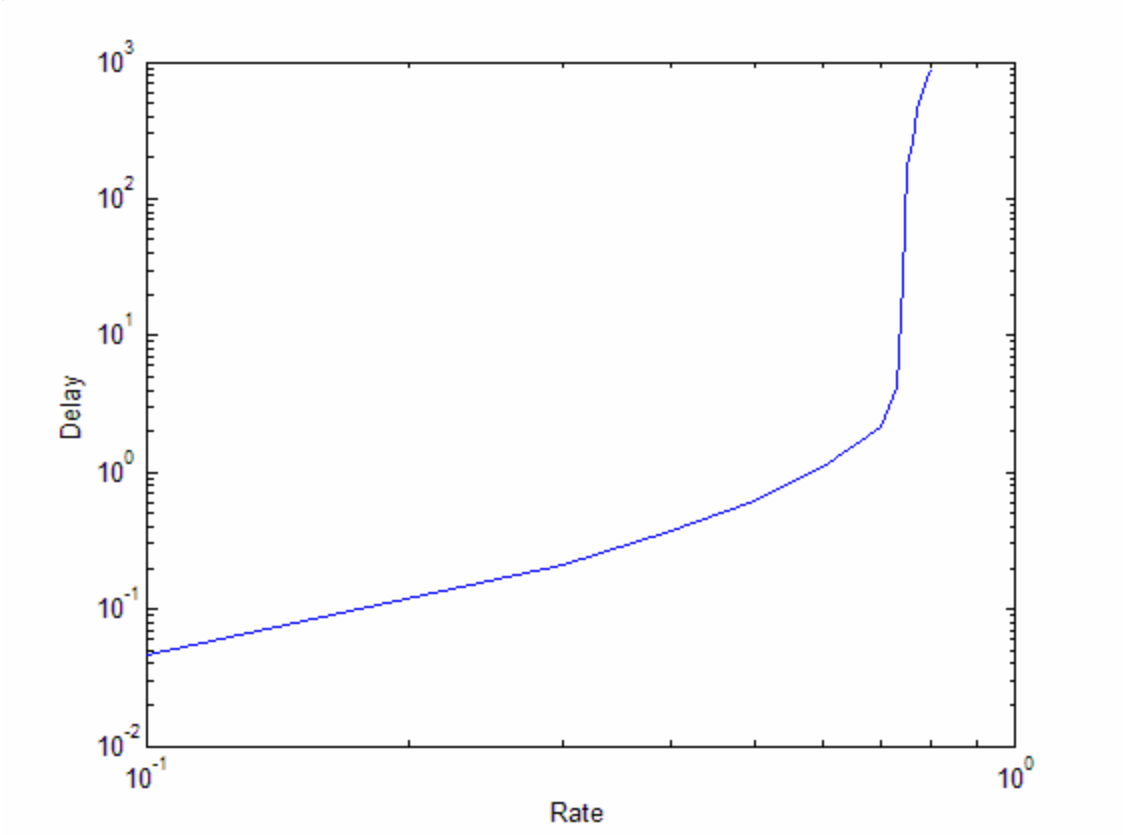

Figure 2 Average Delay vs. Throughput for MFM algorithm.

Inefficiency is clearly observable by means of an analytical counter-example. Consider the following non-uniform arrival rate matrix and corresponding backlog matrix:

$$\Lambda = \begin{pmatrix} 0 & 0 & .5(1-\alpha-\beta) \\ 0 & 0 & .5(1-\alpha-\beta) \\ .5(1-\alpha-\beta) & .5(1-\alpha-\beta) & \beta \end{pmatrix}, Q = \begin{pmatrix} 0 & 0 & q_{13}(n) \\ 0 & 0 & q_{23}(n) \\ q_{31}(n) & q_{32}(n) & q_{33}(n) \end{pmatrix} \quad (3)$$

where $0 \le \alpha, \beta < 1$. Now consider this scheduling algorithm:

Algorithm: Serve q_{33} only if $q_{31}(n) = q_{32}(n) = q_{13}(n) = q_{23}(n) = 0$; otherwise serve some other non-empty VOQ's to maximize weight of the matching.

It's easy to see that the generated matching is always in MNCM with given arrival matrix. That's because when $q_{33}(n) \neq 0$, the max-weight node is input 3, output 3, or both. Now note that Rate of service to $q_{33} = \Pr\{q_{31}(n) = q_{32}(n) = q_{13}(n) = q_{23}(n) = 0\}$

$$\le \Pr\{\text{no arrival to these VOQ's at time } n\} = \left[1 - .5(1-\alpha-\beta)\right]^2 . \left[1 - (1-\alpha-\beta)\right]$$

$$= \frac{1}{4}(1+\alpha+\beta)^2 . (\alpha+\beta) \quad (4)$$

The last quantity in equation (4) can be less than β, e.g. with $\alpha = 0.1, \beta = 0.3$, and hence q_{33} will become unstable.

3.3 Uniform

A similar counter-example for uniform arrival pattern can be stated as below:

$$\Lambda = \begin{pmatrix} \beta & \beta \\ \beta & \beta \end{pmatrix}, Q = \begin{pmatrix} q_{11}(n) & q_{12}(n) \\ q_{21}(n) & q_{22}(n) \end{pmatrix} \tag{5}$$

Algorithm: Don't serve q_{11} and q_{22} if $q_{12}(n) \neq 0$ or $q_{21}(n) \neq 0$.

$$\begin{aligned} \text{Rate of service to } q_{11} \text{ and } q_{22} &= \Pr\{q_{12}(n) = q_{21}(n) = 0\} \\ &\leq \Pr\{\text{no arrival to } q_{12} \text{ and } q_{21} \text{ at time } n\} = (1-\beta)^2 \end{aligned} \tag{6}$$

And this can be less than arrival rate to q_{11} and q_{22} (2β), say for $\beta = 0.4$.

3.4 Issues with Fluid Model

Now the question arises here is why the fluid model proof which seems to be legitimate gives a wrong result. The issue lies in the argument they gave to take the derivative of the Lyapunov function. In [1] they discus that:

"Due to continuity properties of B(t), for every $t_0 \geq 0$ *there exists some* $\delta \geq 0$ *such that for all* $t \in [t_0, t_0 + \delta[$ *there is always one common index* $q(t_0, t_0 + \delta)$ *such that* $f(B(t)) = B_{q(t_0, t_0+\delta)}(t)$*."*

That means during an interval of $\delta \geq 0$, the maximum weight node keeps being maximum. But this is wrong! An interval of length $\delta \geq 0$ in continuous time, corresponds to an interval of arbitrarily large length ($r\delta$ as $r \to \infty$) in discrete time domain. This is not guaranteed by MNCM. Actually the index of maximum weight node can have arbitrarily large rate of change over discrete time. This can be easily seen using a periodic arrival pattern example.
Overall, the fluid model proved to be weakly stable, doesn't represent the original discrete system, and hence the proof is invalid.

4 Fluid Analysis of LPF

After finishing the first problem, I got interested in finding a fluid model proof for stability of LPF under general traffic.

4.1 Approach

The discontinuity in definition of link weights in LPF algorithm makes it difficult to derive a direct fluid proof and to justify the equivalency of fluid and discrete systems. So in this way, some innovative ideas might be helpful. One idea is to find a series of efficient algorithms whose limiting point is LPF algorithm. Then it may be easier to prove stability and equivalency at each point of the series of algorithms.
I define a more general algorithm LPF_f as below:

LPF_f Algorithm: Apply MWM algorithm on the following link weights to find the scheduling matching at time n:

$$W_{ij}^{D}(n) = f(Q_{ij}(n)).\left[\sum_{k} Q_{ik}(n) + \sum_{k} Q_{kj}(n)\right] \tag{7}$$

Where $f(Q)$ is an increasing function and $f(Q_{ij}(n)) = 0$ if $Q_{ij}(n) = 0.$

Note that standard LPF is included in this class with $f(Q_{ij}) = 1_{\{Q_{ij}>0\}}$. Having defined a series of functions converging to $1_{\{Q_{ij}>0\}}$, we will have a series of algorithms converging to LPF.

4.2 Stability of Fluid Policy

I'll define the fluid model link weights to be:

$$W_{ij}^{F}(t) = g(Z_{ij}(t)).\left[\sum_{k} Z_{ik}(t) + \sum_{k} Z_{kj}(t)\right] \tag{8}$$

Here I'm using function $g(\cdot)$ instead of $f(\cdot)$. This is to provide enough flexibility in order to make the fluid model and discrete model equivalent by choosing an appropriate function $g(\cdot)$ (discussed later). Now we have a theorem.

Theorem 1: The fluid model based on weights $W_{ij}^{F}(t)$ is weakly stable under MWM policy if $g(z) \le A$ and $z.g'(z) \le B, \forall z \ge 0$ for some constants $A, B \ge 0.$

Proof: Let's define our Lyapunov function to be $L(t) = \ < Z(t), W^{F}(t) >$. Taking the derivative we have:

$$\dot{L}(t) = \ < \dot{Z}(t), W^{F}(t) > + < Z(t), \dot{W}^{F}(t) >, \tag{9}$$

$$< Z(t), \dot{W}^{F}(t) >= \sum_{i,j} Z_{ij} g(Z_{ij}) \left[\sum_{k} Z'_{ik}(t) + \sum_{k} Z'_{kj}(t) \right] + \sum_{i,j} Z_{ij} Z'_{ij} g'(Z_{ij}) \left[\sum_{k} Z_{ik}(t) + \sum_{k} Z_{kj}(t) \right]$$

$$\leq A \sum_{i,j} Z_{ij} \left[\sum_{k} Z'_{ik}(t) + \sum_{k} Z'_{kj}(t) \right] + B \sum_{i,j} Z'_{ij} \left[\sum_{k} Z_{ik}(t) + \sum_{k} Z_{kj}(t) \right] \quad (10)$$

$$= (A+B) < \dot{Z}(t), W^{F}(t) >$$

But the algorithm applies MWM based on weights $W^{F}(t)$, so we have:

$$< \dot{Z}(t), W^{F}(t) >\leq 0 \quad (11)$$

$$(15),(16),(17) \Rightarrow \dot{L}(t) \leq (1 + A + B) < \dot{Z}(t), W^{F}(t) >\leq 0 \quad (12)$$

So the fluid model is weakly stable.■

4.3 Equivalency of Fluid and Discrete Models

Now we should choose a function $g(\cdot)$ such that the fluid policy and the discrete policy always make the same decision, for any set of queue occupancy Q_{ij}'s and their corresponding Z_{ij}'s. That means the same matching π^{*} must be the maximum weight matching for both models:

$$\sum_{(i,j)\in\pi^{*}} W_{ij}^{D} \geq \sum_{(i,j)\in\pi} W_{ij}^{D} \Leftrightarrow \sum_{(i,j)\in\pi^{*}} W_{ij}^{F} \geq \sum_{(i,j)\in\pi} W_{ij}^{F} \quad (13)$$

Or,

$$\sum_{(i,j)\in\pi^{*}} W_{ij}^{D} \geq \sum_{(i,j)\in\pi} W_{ij}^{D} \Leftrightarrow \sum_{(i,j)\in\pi^{*}} \lim_{r\to\infty} g(\frac{Q_{ij}}{r}).\left[\sum_{k} \frac{Q_{ik}}{r} + \sum_{k} \frac{Q_{kj}}{r} \right] \geq \sum_{(i,j)\in\pi} \lim_{r\to\infty} g(\frac{Q_{ij}}{r}).\left[\sum_{k} \frac{Q_{ik}}{r} + \sum_{k} \frac{Q_{kj}}{r} \right]$$

This will be satisfied if we have:

$$\lim_{r\to\infty} g(\frac{Q_{ij}}{r}) = f(Q_{ij}) \quad (14)$$

Now again the trick is to find a series of functions that satisfies (19) for every r, and also possibly for $r \to \infty$. If I choose:

$$g_{r}(z) \triangleq f(rz) \quad (15)$$

Then (19) based on $g_{r}(\cdot)$ holds for every value of r. But our fluid model corresponds to $r \to \infty$, so if (19) remains valid for $r \to \infty$ ($g_{r}(\cdot) \to g_{\infty}(\cdot)$) it means that the equivalency of (19) holds between our fluid model (based on $g_{\infty}(\cdot)$) and the discrete model. For (19) to remain valid as $r \to \infty$, uniform convergence of g_{r} to g_{∞} suffice.

Now as a good example, suppose:

$$f(Q) = 1 - e^{-aQ} \quad \text{for } Q \geq 0 \text{ and constant } a > 0 \quad (16)$$

$$\Rightarrow g_{r}(z) = 1 - e^{-arz} \quad \text{for } z \geq 0 \quad (17)$$

Thus equivalency (19) holds for this choice of f and g_r $(\forall r > 0)$. It's easy to show that convergence of $g_r \to g_\infty$ is uniform, therefore (19) also holds for f and g_∞ which represents our fluid model. It's interesting that g_∞ does not depend on value of a, i.e. the fluid model is the same for any choice of constant a. I believe it's also possible to argue (19) directly for f and g_∞ without any need for limit and convergence issues.
Now note that:

$$\lim_{z\to 0} z.g'_\infty(z) = \lim_{\substack{z\to 0 \\ r\to\infty}} z.g'_r(z) = \lim_{\substack{z\to 0 \\ r\to\infty}} arz.e^{-arz} \le e^{-1} \qquad (18)$$

By theorem 1, this fluid model is weakly stable.

4.4 Stability of LPF

Again considering previous example for $f(\cdot)$, note that this function converges to $1_{\{Q>0\}}$ as $a \to \infty$, and this is again a uniform convergence. By taking this limit, the weak stability of the fluid model keeps valid (because g_∞ would be the same and still (24) holds). But we should argue the equivalency. A good intuition on this is that when a becomes larger and larger, with fixed r, g_r gets closer to g_∞ under the same rate of convergence as when r becomes larger. Therefore as $a \to \infty$, the uniform convergence of g_r to g_∞ remains uniform, which results in (19) being valid for fluid model of LPF (which is essentially the same for any value of a).

5 iSLIP Random

Parallel to previous problem, I was also working on iSLIP random and I wished to find results on stability and convergence of this algorithm.

5.1 Introduction

iSLIP random [4] is an iterative scheduling algorithm operating on an IQ switch with VOQs and is defined as below:

iSLIP Random: In each iteration, each input port selects one of it's non empty VOQs uniformly at random and requests that output. After all requests, each input selects one of its incoming requests uniformly at random to grant. This specifies a sub matching to be removed from the graph and then run the next iteration.

Essentially I want to consider just the *first* iteration of the algorithm to see what the minimum average size of the generated matching is (compared with the size of a maximal matching). That is to find:

$$\alpha(N) = \min_{\substack{\text{all possible N}\times\text{N} \\ \text{graphs}}} \frac{E[\#\text{ of non-empty output bins}]}{\text{size of maximal matching}} \tag{19}$$

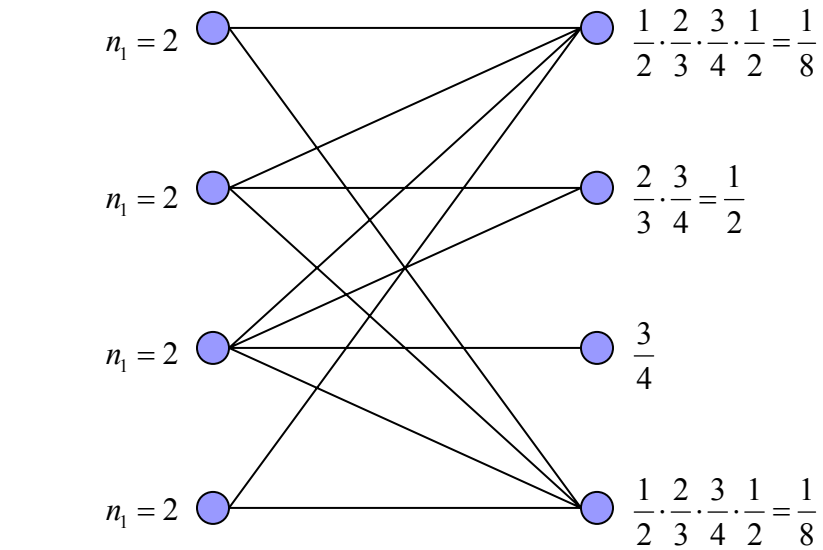

Figure 3 first iteration of iSLIP

We show a non empty VOQ_{ij} by an edge (i, j) in the $N \times N$ bipartite graph of the switch. Let's define:

$$n_i = \text{Number of non empty VOQs at input } i \tag{20}$$

$$O_j \triangleq \{1 - \frac{1}{n_i} \mid \text{VOQ}_{ij} \text{ is non empty}\} \qquad \text{(call them output sets)} \tag{21}$$

$$\pi(O_j) \triangleq \prod_{p \in O_j} p \tag{22}$$

Then obviously:

$$E[\#\text{ of empty output bins}] = \sum_{j=1}^{N} \pi(O_j) \tag{23}$$

What I want to do is to find maximum value of the quantity in (29), given the size of maximal matching. This maximizing is going to be done over all possible $N \times N$ bipartite graphs.

5.2 Approach

First I'm going to find the maximum value of $E[\#\text{ of empty output bins}]$ given a specific vector of input degrees, $(n_1, n_2, ..., n_N)$. I'll assume that the size of maximal matching in the graph is N. It'll become evident later why this assumption doesn't produce any loss of generality. To ensure the size of maximal matching, I'll initially connect every input i to output i $(i = 1, 2, ..., N)$.

Now I introduce a greedy algorithm that with given input degree vector $(n_1, n_2, ..., n_N)$, finds an $N \times N$ bipartite graph which has maximum value of $E[\#\text{ of empty output bins}]$.

Greedy Algorithm: Pick an available input i with smallest n_i and connect it to a possible output with smallest $\pi(O_j)$ (i.e. add $1 - 1/n_i$ to set O_j). Repeat until no available input remains.

Theorem 2: Given $(n_1, n_2, ..., n_N)$ and initially input i connected to output i (for all i), the greedy algorithm maximizes $E[\#\text{ of empty output bins}]$.

Proof of this theorem is based on the following lemma:

Lemma 1: If for fixed $(n_1, n_2, ..., n_N)$, a graph G with sets $O_1, O_2, ..., O_N$ maximizes $E[\#\text{ of empty output bins}] = \sum_{j=1}^{N} \pi(O_j)$, then for any j and k:

$$\text{if } \pi(S_j) > \pi(S_k) \Rightarrow \pi(S_j^c) \geq \pi(S_k^c), \begin{cases} \forall S_j \subseteq O_j - O_k \\ \forall S_k \subseteq O_k - O_j \end{cases} \tag{24}$$

Proof: Suppose in graph G for some j and k, there is S_i and S_j that $\pi(S_j) > \pi(S_k)$ but $\pi(S_j^c) < \pi(S_k^c)$. Now construct a new graph H by moving subset S_j^c from O_j to O_k, and moving subset S_k^c from O_k to O_j (that's always possible because we've eliminated common

parts of O_j and O_k from consideration in these subsets, thus no conflicts will happen). Now note that:

$$\pi(S_j) - \pi(S_k) > 0 \tag{25}$$

$$\pi(S_k^c) - \pi(S_j^c) > 0 \tag{26}$$

$$\Rightarrow \pi(S_j).\pi(S_k^c) + \pi(S_k).\pi(S_j^c) > \pi(S_j).\pi(S_j^c) + \pi(S_k).\pi(S_k^c) \tag{27}$$

The right quantity appears in H and the left quantity appears in G. Nothing else has changed. So now the sum of products is increased in H, which is in contradiction with the assumption that G maximizes E[# of empty output bins].■

Proof of Theorem 2: Let's call the graph with maximum E[# of empty output bins] to be G with output sets O_j^G, and let H be the graph generated by greedy algorithm with output sets O_j^H. I'll show that the output sets are identical in both graphs and hence H maximizes E[# of empty output bins] too.

Suppose this is not true and for the first time during the execution of greedy algorithm, it chooses a number $p_{ij} = 1 - 1/n_i$ and adds it to O_j^H, but p_{ij} is not included in O_j^G. So there is $q \in O_j^G$ which is not in O_j^H. Certainly $q > p_{ij}$ because greedy algorithm always picks smallest available numbers, and till here both algorithms have been the same. Also p_{ij} will appear somewhere in G, say in O_k^G. But $\pi(O_k^G) \geq \pi(O_j^G)$, because greedy algorithm always chooses the smallest subset to feed. By lemma 1, this is not possible in G. Therefore greedy algorithm always makes the correct decision.■

After this, we need to search for best $(n_1, n_2, ..., n_N)$ to maximize E[# of empty output bins]. This can be done using simulations based on greedy algorithm, and then proving the results analytically.

I guess the vector $(1, 2, ..., N)$ is the maximum point, or at least it's very close to maximum for large N. Yet I have no proof for that and it'll be added to the list of future work! But if this is true, then we can conclude that:

- $E[\#\text{ of empty output bins}] \leq \dfrac{N-1}{2} \Rightarrow \alpha(N) \geq \dfrac{N+1}{2N}$
- Hence iSLIP random with only one iteration would give 100% throughput with speedup 4. That's because any maximal matching algorithm gives 100% throughput with speedup 2.
- There's an upper bound on the average of maximum number of iterations needed for iSLIP random to converge:

$$E[\max\ \#\text{ of iterations needed}] \leq \log_2 N.$$

6 Summary

What I did in this project:

- Work on MNCM algorithm which finally resulted in some counter examples, and found out the issue with the fluid model proof.
- An innovative look into fluid model method to prove stability of LPF under general traffic. Although I think the proof needs to be more detailed and accurate.
- A new way toward analyzing stability and convergence of iterative scheduling algorithm. The problem is not finished yet.

Essentially, it was a unique experience for me to take the course. Failing the first problem made me motivated enough to look for new problems, to try risky things and to be more adventurous; what I've never experienced before; facing with challenging problems while constrained in time.
Thank you dear TA's and Professors for your great assistance.

7 References

[1] V. Tabatabaee, L. Tassiulas, "MNCM a new class of efficient scheduling algorithms for input-buffered switches with no speedup", *INFOCOM '03*

[2] A. Mekkittikul, N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches", *INFOCOM '98*

[3] J.G. Dai, B. Prabhakar, "The throughput of data switches with and without speedup", *INFOCOM '00*

[4] "iSLIP: A Scheduling Algorithm for Input-Queued Switches", Nick McKeown , *IEEE Transactions on Networking, Vol 7, No.2, April 1999.*