"It only takes a memory bandwidth of $3NR$?
We had a bit more speedup than that . . . ".

— Pradeep Sindhu[†]

Proofs for Chapter 3

C.1 Proof of Lemma 3.1

Lemma 3.1. *A request matrix can be ordered in no more than $2N-1$ alternating row and column permutations.*

Proof. We will perform the ordering in a iterative way. The first iteration consists of one ordering permutation of rows or columns, and the subsequent iterations consist of two permutations, one of rows and one of columns. We will prove the theorem by induction.

1. After the first permutation, either by row or by column, the entry at $(1,1)$ is non-zero, and this entry will not be moved again. We can define sub-matrices of S as follows:

$$\begin{aligned} A_n &= \{S_{ij} | 1 \leqslant i,j \leqslant n\}, \\ B_n &= \{S_{ij} | 1 \leqslant i \leqslant n, n \leqslant j \leqslant N\}, \\ C_n &= \{S_{ij} | n \leqslant i \leqslant N, 1 \leqslant j \leqslant n\}. \end{aligned} \tag{C.1}$$

2. If a sub-matrix of S is ordered and will not change in future permutations, we call it *optimal*. Suppose A_n is optimal after the n^{th} iteration. We want to prove

[†]Pradeep Sindhu, CTO, Juniper Networks.

that after another iteration, the sub-matrix A_{n+1} is optimal. Without loss of generality, suppose a row permutation was last performed, then in this iteration, we will do a column permutation followed by a row permutation. There are four cases:

(a) The entries of B_n and C_n are all zeros. Then $S_{n+1,n+1} > 0$ after just one permutation, so the sub-matrix A_{n+1} is optimal.

(b) The entries of B_n are all zeros, but those of C_n are not. After the column permutation, suppose $S_{m,n+1}(m > n)$ is the first positive entry in column $n+1$, then the first m rows of S are ordered and will remain so. Thus, column $n+1$ will remain the biggest column in B_n, and A_{n+1} is optimal.

(c) The entries of C_n are all zeros, but those of B_n are not. This case is similar to case (b).

(d) The sub-matrices B_n and C_n both have positive entries. The column permutation will not change row $n+1$ such that it becomes smaller than the rows below it. Similarly, the row permutation following will not change column $n+1$ such that it becomes smaller than the columns on its right. So A_{n+1} is optimal.

After at most N iterations, or a total of $2N-1$ permutations, the request matrix is ordered. ❒

C.2 Proof of Theorem 3.7

Theorem 3.7. *If a request matrix S is ordered, then any maximal matching algorithm that gives strict priority to entries with lower indices, such as the WFA [15], can find a conflict-free schedule.*

Proof. By contradiction. Suppose the scheduling algorithm cannot find a conflict-free time slot for request (m, n). This means

$$\sum_{j=1}^{n-1} S_{mj} + \sum_{i=1}^{m-1} S_{in} \geqslant 4. \tag{C.2}$$

Now consider the sub-matrix S', consisting of the first m rows and the first n columns of S. Let's look at the set of the first non-zero entries of each row, L_r, and the set of the first non-zero entries of each column, L_c. Without loss of generality, suppose S'_{11} is the only entry belonging to both sets. (If this is not true, and S'_{kl}, where $k \neq 1$ or $l \neq 1$, also belongs to both L_r and L_c, then we can remove the first $k-1$ rows and the first $l-1$ columns of S' of to obtain a new matrix. Repeat until L_r and L_c only have one common entry.) Then $|L_r \cup L_c| = m + n - 1$. At most two of the entries in the m^{th} row and those in the n^{th} column are in $L_r \cup L_c$, so the sum of all the entries satisfies

$$\sum_i \sum_j S_{ij} \geqslant (|L_r \cup L_c| - 2) + S_{mn} + \sum_{j=1}^{n-1} S_{mj} + \sum_{i=1}^{m-1} S_{in} \geqslant 4. \tag{C.3}$$

Hence we get,

$$\sum_i \sum_j S_{ij} \geqslant m + n + 2, \tag{C.4}$$

which conflicts with property 1 in Section 3.6. ❒