IEOR 8100: Scheduling Algorithms
Lecture 4

Instructor: Clifford Stein
Scribe: Sandip Sinha

September 16, 2016

In this lecture, we complete the analysis of the correctness of the polynomial-time algorithm for $R|pmtn|C_{\max}$ by showing the existence of a matching among jobs and machines which includes all the "tight nodes", prove that $R||C_{\max}$ is **NP**-complete, and give a 2-approximation algorithm for $R||C_{\max}$ using LP relaxation and rounding.

1 R|pmtn|$C_{\max}$ (continued from previous lecture)

We complete the analysis of the optimal algorithm for $R|pmtn|C_{\max}$. Recall the following theorem and definition:

Theorem 1. *Given a matrix $T = (T_{ij})_{i\in[m],j\in[n]}$ with non-negative entries, it is always possible to find a feasible schedule such that*

$$C_{\max} = \max\left\{\max_i\left(\sum_j T_{ij}\right), \max_j\left(\sum_i T_{ij}\right)\right\}$$

Definition 1. We say a row i is tight if $\sum_j T_{ij} = C_{\max}$. Similarly, we say a column j is tight if $\sum_i T_{ij} = C_{\max}$.

We construct a bipartite graph having nodes for jobs and machines. For a given job j and machine i, there is an edge $e = (j, i)$ iff $T_{ij} > 0$. We say a node is tight if the corresponding row or column in the matrix t is tight. We want to compute the matching that matches all the tight nodes. The remaining part of the analysis is to show that in such a graph, there always exists a matching which matches all the tight nodes. This follows from the Birkhoff-von Neumann theorem.

Definition 2. A doubly stochastic matrix is a square non-negative real matrix with all entries being real and non-negative, and all row sums and column sums equal to 1.

Definition 3. A permutation matrix is a square $0-1$ matrix with exactly one 1 in each row and column.

Obviously, a permutation matrix is a doubly stochastic matrix. It is easy to verify a convex combination of doubly stochastic matrices (and in particular, permutation matrices) is also a doubly stochastic matrix.

Theorem 2 (Birkhoff-Von Neumann)**.** *Given any doubly stochastic matrix Q, there exist k permutation matrices $P_1, \cdots, P_k$ and non-negative real numbers $\theta_1, \cdots, \theta_k$ for some $k \in \mathbb{N}$ such that*

$$Q = \sum_{i=1}^{k} \theta_i P_i$$

Clearly, in the above decomposition, $\sum_{i=1}^{k} \theta_i = 1$.

We provide an example of a doubly stochastic matrix along with its decomposition into permutation matrices.

$$\begin{pmatrix} 0.3 & 0.4 & 0.3 \\ 0.5 & 0 & 0.5 \\ 0.2 & 0.6 & 0.2 \end{pmatrix} = 0.3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} + 0.2 \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 0.3 \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 0.2 \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

It is **NP**-hard to determine the minimum k in the above decomposition (more precisely, the problem of deciding, for a given Q and k, whether there exists a decomposition with at most k permutation matrices is **NP**-hard).

The following discussion is interesting but not important for the purpose of proving the main theorem. Fix $n \in \mathbb{N}$, and consider the set of $n \times n$ doubly stochastic matrices as a subset of $\mathbb{R}^{n^2}$. This set is a convex polytope, known as the *Birkhoff polytope* $\mathbf{B_n}$ [1]. It is easy to verify that a permutation matrix must be a vertex of this polytope. Suppose, for the sake of contradiction, that P is a permutation matrix which is not a vertex. Then P can be written as a convex combination of other doubly stochastic matrices:

$$P = \sum_{i=1}^{k} \theta_i Q_i$$

with $\theta_i > 0$ for all i, $\sum_{i=1}^{k} \theta_i = 1$, Q_i doubly stochastic matrix and $Q_i \neq P$ for all i. As $Q_1 \neq P$, there must exist indices $r, c \in [n]$ such that $P_{rc} = 0$ and $Q_{1_{rc}} > 0$. Thus we have

$$\sum_{i=1}^{k} \theta_i Q_{i_{rc}} \geq \theta_1 Q_{1_{rc}} > 0 = P_{rc}$$

which is a contradiction. Thus, permutation matrices are always vertices of this polytope. The Birkhoff-von Neumann theorem can be interpreted as saying that the vertices of the polytope are precisely the permutation matrices and no other doubly stochastic matrices.

There is an obvious bijection from $n \times n$ permutation matrices P to perfect matchings on a bipartite graph $G = (U, V, E)$ with $|U| = |V| = n$. For simplicity of notation, let $U = V = [n]$. We have a vertex in U for each row in P and a vertex in V for each column in P. There is an edge $e = (i, j) \in E$ if and only if $P_{ij} = 1$.

Claim 1. In any doubly stochastic matrix, there exists a perfect matching on the non-zero elements.

We will use Hall's theorem (which we state without proof) to prove the claim. Let $G = (V, E)$ be an undirected graph. For a vertex v, we define the neighbors of v by

$$N(v) := \{u \in V | \exists \text{ edge } e = (u, v) \in E\}.$$

For a subset $S \subset V$, we define the neighbors of S by $N(S) := \bigcup_{v \in S} N(v)$.

Theorem 3 (Hall)**.** *A bipartite graph $G = (U, V, E)$ has a perfect matching if and only if for all subsets $S \subset U$, $|N(S)| \geq |S|$.*

Note that the theorem considers all subsets of either one of the partitions.

Proof of Claim 1. Suppose Q is a doubly stochastic matrix but there exists no perfect matching on the corresponding bipartite graph $G = (U, V, E)$. Then, by Hall's theorem, there exists $S \subset U$ such that $|S| > |N(S)|$ This leads to a contradiction:

$$\sum_{i \in S, j \in N(S)} Q_{ij} = |S| > |N(S)| = \sum_{i \in N(N(S)), j \in N(S)} Q_{ij} \geq \sum_{i \in S, j \in N(S)} Q_{ij}$$

□

This claim immediately leads to proofs of 1 and 2. In fact, they are essentially different formulations of the same result, as will be evident from the proofs below.

Proof of Birkhoff Von-Neumann Theorem 2. If Q is a permutation matrix, setting $k = 1$ and $P_1 = 1$, we are done. Otherwise, the claim guarantees the existence of a perfect matching M on the non-zero elements of Q. We abuse notation and use M to also refer to the permutation matrix corresponding to the matching. Let

$$\delta = \min\{Q_{ij} | e = (i, j) \in M\}.$$

Note that $0 < \delta < 1$. We have $Q = \delta M + (1 - \delta)Q'$, where M is a perfect matching and Q' is a doubly stochastic matrix with strictly fewer non-zero entries than Q'. We apply the claim to Q'. Iterating this process at most n^2 times, we get the required decomposition. □

Proof of Theorem 1. The matrix T is not necessarily square, nor does it have all row and column sums equal. Let $T_{\max}$ be the maximum of the row sums and the column sums. We will transform T into a non-negative square matrix $\tilde{T}$ having all row and column sums equal to $T_{\max}$.

$$\tilde{T} := \begin{pmatrix} T & D_r \\ D_c & T^T \end{pmatrix}$$

where D_r is a m-dimensional diagonal matrix with $D_r(i, i) = T_{\max} - \sum_j T_{ij}$ and D_c is a n-dimensional diagonal matrix with $D_c(j, j) = T_{\max} - \sum_i T_{ij}$. We demonstrate the transformation below with the same instance of T as that used in the previous lecture.

$$\begin{pmatrix} 3 & 4 & 0 & 4 \\ 4 & 0 & 6 & 0 \\ 4 & 0 & 0 & 6 \end{pmatrix} \mapsto \begin{pmatrix} 3 & 4 & 0 & 4 & 0 & 0 & 0 \\ 4 & 0 & 6 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 6 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 3 & 4 & 4 \\ 0 & 7 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 1 & 4 & 0 & 6 \end{pmatrix}$$

As $\tilde{T}$ is a scaled version of a stochastic matrix, we can effectively apply Claim 1 to $\tilde{T}$. If a row i is tight in T, then the corresponding diagonal entry $D_r(i, i)$ in $\tilde{T}$ is 0. Hence, the row i in $\tilde{T}$ contains non-zero entries only in the positions from T and T^T. Similarly, for any tight column j in T, all non-zero entries in the column are restricted to be in the positions from T and T^T. By Claim 1, the matrix $\tilde{T}$ has a matching on the non-zero elements, and hence the matching includes all the tight nodes in T. This completes the proof. □

2 NP-completeness of $\mathbf{R||C_{max}}$

We will actually show that it is **NP**-complete to decide whether a given instance of $R||C_{\max}$ has $C_{\max} \leq 3$. Clearly the problem is in **NP**, as it is easy to verify a given certificate. The hardness reduction is from the problem of deciding whether there exists a (perfect) $3D$-Matching, which is known to be **NP**-complete. We note that this reduction can be improved to show that it is **NP**-hard to decide if $C_{\max} \leq 2$ for $R||C_{\max}$, which leads to the following Hardness of Approximation result:

Fact 1. Let $1 \leq \rho < \frac{3}{2}$. There does not exist a ρ-approximation algorithm for $R||C_{\max}$ with all processing times $P_{ij} \in \mathbb{N}$, unless $\mathbf{P} = \mathbf{NP}$.

Claim 2. $3D$-Matching $\leq_P R||C_{\max}$

Proof. Let G be a tripartite graph with n nodes in each partition and m edges. Specifically, $G = (V, E)$ where $V = A \sqcup B \sqcup C$ with $|A| = |B| = |C| = n$ and $E \subset A \times B \times C$ with $|E| = m$. We fix an arbitrary ordering on the m edges and the $3n$ nodes. The construction is given in detail below:

- Each edge i, $1 \leq i \leq m$, corresponds to a machine M_i.
- Each node j, $1 \leq j \leq 3n$, corresponds to a job with processing time

$$P_{ij} = \begin{cases} 1 & \text{if node } j \text{ is incident to edge } i \\ 3 & \text{otherwise} \end{cases}$$

- There are $m - n$ "dummy" jobs j with processing time $P_{ij} = 3$ for all machines i.

We now show that there exists a $3D$-Matching in the graph if and only if $C_{\max} = 3$, i.e. there exists a feasible schedule with makespan 3. Clearly, it is impossible to design a schedule with less makespan since each dummy job itself takes 3 units of time.

3D-Matching $\Rightarrow$ $\mathbf{C_{max} = 3}$:

Suppose $F \subset E$ is a perfect 3D-matching. Then $|F| = n$. Each edge $e = (a, b, c) \in M$ corresponds to a machine M_e, and we schedule jobs a, b and c to M_e in any order. Each job takes 1 unit of time and hence each such machine requires 3 units of time. There are $m - n$ machines remaining, and we schedule each dummy job on one of these machines, which again requires 3 units of time. As F is a matching, no job is processed by two machines. Thus, this gives a feasible schedule of makespan 3.

$\mathbf{C_{max} = 3}$ $\Rightarrow$ 3D-Matching:

Suppose S is a schedule with makespan 3. Clearly, the $m - n$ dummy jobs, requiring 3 units of time on each machine, are scheduled on separate machines. This means that there are $m - (m - n) = n$ machines remaining. Since there are $3n$ jobs to be scheduled and only n machines, it must be the case that each machine gets allocated 3 jobs, each of which requires 1 unit of time to run on that machine. By construction, for each machine M_i, there is exactly 1 job j in each partition, such that $P_{ij} = 1$. Thus, the 3 jobs that

are allocated to a particular machine with processing time 1 each must be from different partitions, and hence correspond to an edge in the graph. Define

$$F := \{e = (a, b, c) \in E | \text{ jobs a, b and c are scheduled on the same machine}\}$$

As S is a valid non-preemptive schedule, no job gets scheduled on more than one machine. Further, by the discussion above, $|F| = n$. Hence, F is a perfect $3D$-Matching. □

3 2-approximation for $\mathbf{R||C_{max}}$

We show a 2-approximation algorithm for $R||C_{\max}$ using LP relaxation and rounding.

3.1 Feasibility LP

For $i \in [m], j \in [n]$, let X_{ij} be the indicator variable for job j running on machine M_i.

$$X_{ij} := \begin{cases} 1 & \text{if job } j \text{ runs on machine } i \\ 0 & \text{otherwise} \end{cases}$$

We fix a bound D and write a feasibility LP checking if $C_{\max} \leq D$. Ultimately, we will run a binary search over D and use the algorithm described below as a subroutine to determine $C_{\max}$.

We will work with the following running example, in which the processing times are specified by the table below and $D = 9$. $P_{ij} = \infty$ denotes that it is not possible to run job j on machine i (in practice, we would replace ∞ by a large constant).

P	J1	J2	J3	J4	J5	J6	J7
M1	9	3	1	∞	∞	∞	∞
M2	12	∞	∞	2	1	∞	∞
M3	5	∞	∞	∞	∞	2	3

Table 1: An example of processing times P_{ij}

Since the program will be an integer program which is hard to solve, we will relax the constraint on X_{ij}'s, allowing them to be non-negative real numbers instead of $\{0, 1\}$ variables. This essentially means that as per the LP, a job can be scheduled on multiple machines.

The constraints of the LP are as follows:

$$\sum_{j=1}^{n} X_{ij} P_{ij} \leq D \text{ for all } 1 \leq i \leq m,$$
$$\sum_{i=1}^{m} X_{ij} = 1 \text{ for all } 1 \leq j \leq n,$$
$$X_{ij} \geq 0, \text{ for all } 1 \leq i \leq n, 1 \leq j \leq m,$$
$$X_{ij} = 0 \text{ if } P_{ij} > D \text{ for all } 1 \leq i \leq n, 1 \leq j \leq m.$$

The final constraint is imposed to facilitate rounding. Note that for a job j and a machine M_i, if $P_{ij} > D$, it is not impossible to design a schedule with makespan $C_{\max} \leq D$ in which job j is scheduled on machine i. If this constraint is not enforced, the LP might put a small non-zero weight on X_{ij}. The rounding procedure might assign job j to machine i, which can make the rounded solution arbitrarily bad.

There are at most mn constraints of the last type, and they can be enforced during pre-processing or simply removing the variables from the LP. We observe that this constraint requires D to be a known constant, and this is why we write a feasibility LP instead of writing a minimization LP subject to the first 3 constraints.

3.2 Rounding

Let X be the LP solution. X can be represented as a $m \times n$ non-negative column-stochastic matrix. We want a solution that is in the support of this matrix, i.e., a schedule in which each job j is allocated to a machine i such that $X_{ij} > 0$.

X	J1	J2	J3	J4	J5	J6	J7
M1	1/3	1	1	0	0	0	0
M2	0	0	0	1	1	0	0
M3	2/3	0	0	0	0	1	1

Table 2: An example of the LP solution X for the instance specified by 1

For $1 \leq i \leq m$, define $K_i := \lceil \sum_j X_{ij} \rceil$. Construct a bipartite graph $G = (J, M, E)$ with 1 node in J for each job (called a job node), and k_i nodes in M for each machine i (called machine nodes). We denote the K_i copies of machine i by $i_k, 1 \leq k \leq K_i$. For machine i, sort the jobs in decreasing order of P_{ij}, and assign the jobs to the machine copies i_k greedily with 1 unit of X to each node. The values of X_{ijk} in this assignment are the edge weights on the bipartite graph, as shown in the figure.

Clearly, this is a fractional matching with the total weight on any job node being 1 and the total weight on any machine node being at most 1. We define the value of a matching as the maximum of the total weight of nodes on either side. The following theorem guarantees that we can find an integral matching of same value as the fractional matching.

Theorem 4. *Let $G = (U, V, E)$ be a bipartite graph. If there exists a fractional matching of value V^* in G, then there exists an integral matching of the same value V^* in G.*

This is obviously not true for non-bipartite graphs. The triangle K_3 with all edges having weight $1/2$ is itself a fractional matching of value $3/2$. However, no integral matching of value $3/2$ exists.

This result follows from the fact that the adjacency matrix of a bipartite graph is totally unimodular (the determinant of any square submatrix is $0, 1$ or -1). However, we give a simple proof sketch which avoids this result. Given a fractional matching, we construct an integral matching iteratively.

Proof. Let M_f be a fractional matching in the graph. If it is integral, we are done. Otherwise, choose a vertex $u_1 \in U$ which has an edge $e = (u_1, v_1) \in M_f$ with $0 < w_e < 1$. Since e is incident on v_1, there must be another edge of fractional weight $e' = (u_2, v_2) \in M_f$ with $u_2 \neq u_1$. Continuing in this manner, we can eventually find a cycle (of even length)

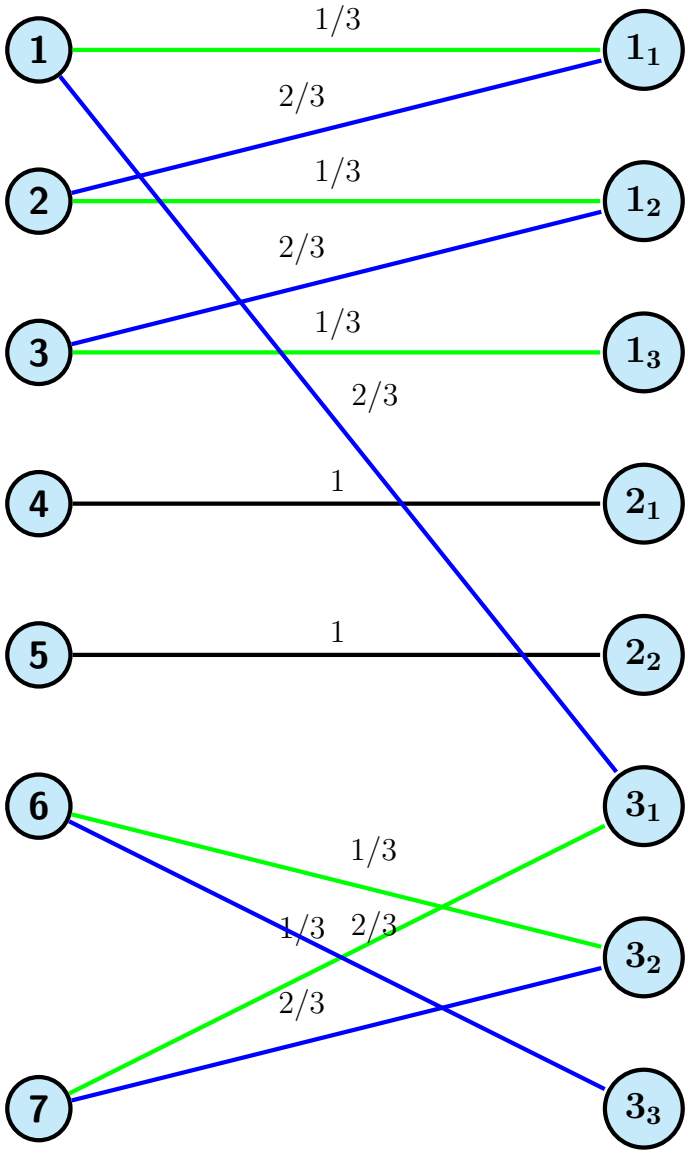

Figure 1: The fractional matching $\tilde{X}$ constructed from 2. The green, blue and black lines represent edges of weight 1/3, 2/3 and 1 respectively.

$C \subset M_f$ of fractional edges in the matching. Starting with $e_1 = (u_1, v_1)$, label the edges in the cycle as '+' and '-' alternatively. Let $\delta = \min(w_e, 1 - w_e)$ over all $e \in C$. Clearly, $0 < \delta \leq 1/2$. Without loss of generality (since we can invert the signs on all edges), assume δ is achieved for a '+' edge e^*. If $w_{e^*} < 1/2$, modify the edge weights $w'_e = w_e - \delta$ for all '+' edges and $w'_e = w_e + \delta$ for all '-' edges. Otherwise, set $w'_e = w_e + \delta$ for all '+' edges and $w'_e = w_e - \delta$ for all '-' edges. The weights remain unchanged for all $e \notin C$.

It is easy to observe that with the changed weights, we still have a valid fractional matching of the same value, due to the fact that all new edge weights satisfy $0 \leq w_e \leq 1$ and for each vertex, there is a bijection between '+' and '-' edges. By this procedure, the weight of some edge $e = (u, v)$ in the cycle gets modified to 1 or 0. If $w'_e = 1$, then u and v are matched by an integral edge. All other edges incident to u or v must have weight 0 and hence they can be removed without affecting the value of the matching. Otherwise, $w'_e = 0$, in which case this edge can be removed. In either case, we have managed to remove at least one edge with fractional weight, and hence obtained a fractional matching

with fewer edges. Iterating this process at most $|M_f| \leq m$ times, we can find an integral matching of the same value. □

Thus, the algorithm is simply to construct G using the LP solution X, compute an integral matching in G and schedule jobs to machines according to the matching. It remains to analyze the maximum load on a machine after this assignment.

3.3 Analysis

Let X be the LP solution, $\tilde{X}$ be the set of edge weights in the constructed graph, and X' be the integral matching in the graph. Fix a machine i. For a job j, X_{ij} is split across K_i copies of M_i in G. However, the processing time P_{ij} remains the same for all copies. Hence, we have

$$\sum_{k=1}^{K_i}\sum_{j=1}^{n}\tilde{X}_{ijk}P_{ij} = \sum_{j=1}^{n}\sum_{k=1}^{K_i}\tilde{X}_{ijk}P_{ij} = \sum_{j=1}^{n}X_{ij}P_{ij} \leq D$$

For $k \in [K_i]$, let P_{ik}^{max} and P_{ik}^{min} denote the maximum and minimum processing time of a job assigned to the k^{th} copy of i respectively. We note that the maximum load on a particular copy of a machine is at most the minimum load on the previous copy of the same machine, due to the greedy assignment of jobs. Further, $P_{i1}^{max} \leq D$ due to the final constraint in the LP. Finally, P_{ik}^{min}, being the minimum among the processing times of all jobs for the k^{th} copy of the machine, is at most $\sum_{j=1}^{n}\tilde{X}_{ijk}P_{ij}$ since $\sum_{j=1}^{n}X_{ijk} = 1$ for all $k < K_i$ (the minimum of a set of numbers is bounded above by any convex combination).

$$\begin{aligned}
Load(i) &\leq \sum_{k=1}^{K_i} P_{ik}^{max} \\
&= P_{i1}^{max} + \sum_{k=2}^{K_i} P_{ik}^{max} \\
&\leq P_{i1}^{max} + \sum_{k=1}^{K_i-1} P_{ik}^{min} \\
&\leq P_{i1}^{max} + \sum_{k=1}^{K_i-1}\sum_{j=1}^{n}\tilde{X}_{ijk}P_{ij} \\
&\leq D + D \\
&= 2D
\end{aligned}$$

Thus, if there exists a schedule of makespan D, then this algorithm returns a schedule of at most $2D$, which implies a 2-approximation for $R||C_{\max}$ since we can run a binary search over D to approximate $C_{\max}$.

References

[1] Wikipedia article on Doubly Stochastic Matrix, `https://en.wikipedia.org/wiki/Doubly_stochastic_matrix` (last accessed on 22 September, 2016)