

## Matrix Scaling

We discuss yet another algorithm for bipartite perfect matching from. The algorithm is not necessarily very fast, but it is shockingly simple:

**Algorithm** `scale(A)`       $A = \{a_{ij}\}$  is the  $n \times n$  incidence matrix of a bipartite graph  $G$ .

**Output:** Whether  $G$  has a perfect matching.

- 1: **for**  $100n^3 \log n$  steps **do**
- 2:    `NormalizeRows(A)`       $a_{ij}$  is set to  $a_{ij}/r_i$ , where  $r_i$  is the row-sum  $r_i := \sum_{j=1}^n a_{ij}$ .
- 3:    `NormalizeColumns(A)`       $a_{ij}$  is set to  $a_{ij}/c_j$ , where  $c_j$  is the column-sum  $c_j := \sum_{i=1}^n a_{ij}$ .
- 4:    **if**  $r_i = \sum_{i=1}^n a_{ij} \in [1 - 1/n, 1 + 1/n]$  for all  $i$  **then return yes**
- 5: **return no**

**Algorithm 1:** Matrix Scaling Algorithm for Bipartite Perfect Matching

In words, the algorithm alternates between normalizing the rows and columns, and outputs **yes** if it succeeds in approximately normalizing both the columns and rows *simultaneously*. To see the perfect matchings, note that if  $G$  has a perfect matching  $M$ , one can always succeed in normalizing both simultaneously by setting all entries that correspond to the edges of  $M$  to 1 (and the other to 0). But note the algorithm will not necessarily output such a matrix. For example, if  $G$  is regular it will already stabilize after one round. Here is an example partial sequence of normalizations:

1	1	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{3}{5}$	$\frac{2}{5}$	$\frac{2}{7}$	$\frac{21}{45}$	$\frac{14}{45}$	$\frac{10}{45}$	...
1	0	1	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{2}{5}$	0	$\frac{3}{7}$	$\frac{14}{29}$	0	$\frac{15}{29}$	
0	1	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{3}{5}$	$\frac{3}{7}$	0	$\frac{21}{36}$	$\frac{15}{36}$	

**Lemma 1.** *If `scale(A) = yes`, then  $G$  has a perfect matching.*

*Proof.* Let  $A = \{a_{ij}\}$  denote the matrix after the last iteration of the algorithm, and  $r_i$  and  $c_j$  denote the corresponding row/columns-sums. Thus,  $r_i \in [1 - 1/n, 1 + 1/n]$  and  $c_j = 1$  for every  $i, j$ . Suppose  $G$  has no perfect matching. Let  $X := \{x_1, \dots, x_n\}$  and  $Y := \{y_1, \dots, y_n\}$  be the two parts of  $G$  (so  $|X| = |Y| = n$ ). By König's theorem,  $G$  has a vertex cover  $C$  of size at most  $n - 1$ . Let  $X_C := \{i : x_i \in C\}$  and  $Y_C := \{i : y_i \in C\}$ . Since  $a_{ij} > 0$  only if  $\{x_i, y_j\} \in E(G)$ , we have that

$$\sum_{1 \leq i \leq n} c_i \leq \sum_{i \in X_C} \sum_{j \in N(i)} a_{ij} + \sum_{j \in Y_C} \sum_{i \in N(j)} a_{ij} \leq (1 + 1/n)|X_C| + |Y_C| \leq |C| + |C|/n < n,$$

which contradicts that  $c_i = 1$  for every  $i$ . □

It remains to prove the harder direction. That is, if  $G$  has a perfect matching, the condition on Line 4 will be met. We introduce a parameter of the matrix  $A$ , the so-called *permanent*, that measures progress of the algorithm towards the simultaneous normalization. Formally, we define

$$\text{per}(A) := \sum_{\pi \in S_n} \prod_{i=1}^n a_{i,\pi(i)},$$

where  $S_n$  denotes the set of all permutations of  $\{1, \dots, n\}$ . We now prove 3 properties of the progress of  $\text{per}(A)$  during the algorithm:

First, note that if  $G$  has a perfect matching and if  $A$  is the adjacency matrix of  $G$ , the corresponding permutation contributes 1 to  $\text{per}(A)$ . As normalization divides each entry by at most  $n$ , we obtain that if  $A$  is the matrix after the first row-normalization, then

$$\text{per}(A) \geq 1/n^n \tag{1.1}$$

Second, we show that  $\text{per}(A)$  increases significantly if  $r_i \notin (1 - 1/n, 1 + 1/n)$  for some  $i$ . If  $B$  is the result of `NormalizeRows`( $A$ ) and if  $r_i$  are the row sums of  $A$  (where  $A$  is column-normalized), then

$$\text{per}(B) \geq \frac{\text{per}(A)}{\prod_{i=1}^n r_i}, \tag{1.2}$$

as every product has actually one term per row. Letting  $d_i = 1 - r_i$ , and using that  $1 + t \leq \exp(t - t^2/2 + t^3/3)$  we have that

$$\prod_{i=1}^n r_i \leq \prod_{i=1}^n (1 + d_i) \leq \exp\left(\sum_{i=1}^n d_i - \frac{1}{2} \sum_{i=1}^n d_i^2 + \frac{1}{3} \sum_{i=1}^n d_i^3\right) \leq \exp\left(-\frac{1}{6} \sum_{i=1}^n d_i^2\right) \leq \exp(-1/6n^2), \tag{1.3}$$

where we use that  $d_i \leq 1$  in the penultimate inequality and that some  $|d_i| \geq 1/n$  for some  $i$ .

Third, if  $A$  is row (or, similarly, column) normalized, then

$$\text{per}(A) = \sum_{\pi \in S_n} \prod_{i=1}^n a_{i,\pi(i)} \leq \sum_{f: \{1, \dots, n\} \rightarrow \{1, \dots, n\}} \prod_{i=1}^n a_{i,f(i)} = \prod_{i=1}^n r_i \leq \sum_{i=1}^n r_i/n \leq 1, \tag{1.4}$$

where we use the AM-GM inequality in the penultimate inequality.

Combining (1.1)-(1.4) we obtain that the algorithm returns **yes** if a perfect matching exists: If not it would increase  $\text{per}(A)$  with a factor  $\exp(1/6n^2)$  in all  $100n^3 \log(n)$  iterations by (1.2) and (1.3) (amounting to a  $\exp(100n \log(n))$  multiplicative factor), but that contradicts that it starts with  $\text{per}(A) \geq 1/n^n$  (by (1.1)) and stops with  $\text{per}(A) \leq 1$  (by 1.4).

**Literature:** Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, Apr 2000.