Perfect Matching According to Weights - with Vectorization

Yair Mazal

April 21, 2020

Problem Statement

Consider the problem of matching all vertices of a bipartite graph $G = \{F \cup M, \epsilon\}$, where each vertex is matched to no more than a single vertex. Each edge representing a single possible match, has a certain weight. The cardinality of the two sets is equal: $|M| = |F|$. We wish to:

- Formulate the integer programming optimization problem.
- Relax it to an LP problem
- Solve it for the case when every $m \in M$ has only $k \leq |M|$ edges connected to it for various values of k.
- We determine a solution as valid, iff all vertices are matched.

Solution

LP Formulation

Integer Programming Problem

Let us start by thinking on the decision variables and weights as matrices: $w_{ij} = [\mathbf{W}]_{ij}$, $x_{ij} = [\mathbf{X}]_{ij}$. Since we assumed equal cardinality of both vertices sets both matrices are square. The cost functions can then be expressed as:

$$f(\mathbf{x}) = \sum_{(i,j)\in\epsilon} w_{ij}x_{ij} = \sum_{(i,j)\in\epsilon} W_{ij}X_{ij} = \sum_{(i,j)\in\epsilon} W_{ji}^{\intercal}X_{ij} = \sum_{(i,j)\in\epsilon} \underbrace{\mathbf{e}_j^{\intercal}W^{\intercal}\mathbf{e}_i}_{=W_{ji}^{\intercal}} \underbrace{\mathbf{e}_i^{\intercal}X\mathbf{e}_j}_{=X_{ij}}$$

$$= \sum_j \mathbf{e}_j^{\intercal}W^{\intercal} \underbrace{\left(\sum_i \mathbf{e}_i\mathbf{e}_i^{\intercal}\right)}_{\mathbb{I}} X\mathbf{e}_j = \sum_j \mathbf{e}_j^{\intercal}W^{\intercal}X\mathbf{e}_j = \mathrm{tr}\left(W^{\intercal}X\right), \quad (1)$$

where $\mathbf{e}_k$ are unit vector in the k-th direction, $k = 1, \ldots, |M|$. Thus we're able to have a linear cost function (the cost function is a linear operator of the matrix X). The constraints on the problem are such that $x_{ij} \in \{0, 1\}$, and that the sum in each column or row in X is equal to unity. Thus the constraints are:

$$\begin{cases} X \cdot \mathbf{1} = \mathbf{1} \\ \mathbf{1}^\intercal \cdot X = \mathbf{1}^\intercal \\ X_{ij} \in \{0, 1\} \end{cases} . \tag{2}$$

Above and later on $\mathbf{1}$ stands for a column vector of all ones, and $\mathbb{I}$ is the identity matrix.

Relaxation of the Matching Problem

The relaxed problem reads:

$$\begin{aligned} \max_{\{x_{ij}\}_{(i,j)\in\epsilon}} f(\mathbf{x}) = \max_{\{x_{ij}\}_{(i,j)\in\epsilon}} \operatorname{tr}(W^\intercal X) \\ s.t. \quad \begin{cases} X \cdot \mathbf{1} = \mathbf{1} \\ \mathbf{1}^\intercal \cdot X = \mathbf{1}^\intercal \\ 0 \le X_{ij} \le 1 \end{cases} . \end{aligned} \tag{3}$$

Note the difference in the last constraint, following which the relaxed problem is an LP problem. The latter can be converted to standard notation instead of the matrix form using vectorization. This will yield:

$$\begin{aligned} \max_{\{x_{ij}\}_{(i,j)\in\epsilon}} vec(W)^\intercal \, vec(X) \\ s.t. \quad \begin{cases} (\mathbf{1}^\intercal \otimes \mathbb{I}) \cdot vec(X) = \mathbf{1} \\ (\mathbb{I} \otimes \mathbf{1}^\intercal) \cdot vec(X) = \mathbf{1} \\ 0 \le X_{ij} \le 1 \end{cases} , \end{aligned} \tag{4}$$

where $vec(X)$ are simply the columns of X, stacked on top of each other. To solve this with *linprog* the only thing remaining to do is to stack the two equality constraints on top of each other.

Monte-Carlo Simulation

As required the simulation iterates over k which stands for the number of allowed matches per male. For each value of k, I randomly choose potential matches, with random weights. This is repeated 100 times.

To disallow the matching of forbidden matches, before invoking the *linprog* function, I first limit the upper and lower bounds such that $0 \le X_{ij} \le 0 \to X_{ij} = 0$, for combinations i, j of forbidden matches. The code is provided at the end of the question.

Fig. 1 shows the matching success rate vs. k, i.e. the number of times a perfect matching was normalized by the 100 attempts As can be seen for every $k \ge 9$ the success

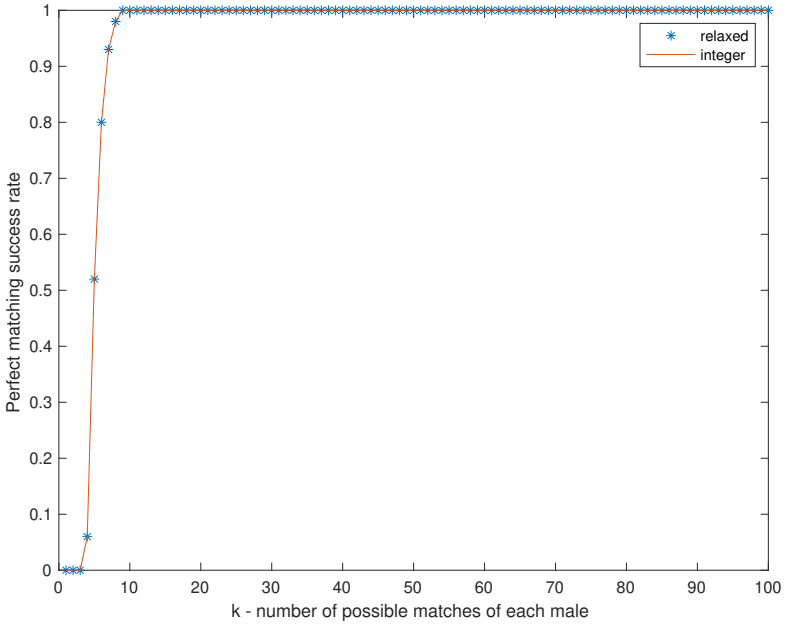

Figure 1: Matching success rate

rate is 1. Moreover, as discussed in class (but not proven), for this specific problem , it seems the solutions to the relaxed problem, are also solutions to the original integer programming problem. The problem to find solutions for small k, arise from the fact that if it is too small, it is possible that some females will not have any eligible matches, i.e. that some vertices will be be completely disjoint, or for instance two female vertices, which are both connected only to a single male vertex. In such case a feasible solution cannot be found, as the constraints require all female vertices be matched to some male vertex. The *Hall Marriage Theorem* provides a necessary and sufficient condition for the matching to occur, and it is easily seen the the chances for satisfying the condition for *X-saturating matching* increases with k.

Solution for Specified Case

Using the code I wrote, this is simple matter of choosing a specific matrix W. Code and its output is provided at the end of the question. The optimal solution reads:

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

where $X_{ij} = 1$ indicates matching the i-th male, with the j-th female. Counting was done from the top of the graph. A more visual solution is provided in Fig. 2. The

optimal value of the cost function (maximized happiness) is: $p^* = 59$. Due to two female vertices, and and one male vertex being connected to only a single potential match, the set of feasible solutions is very limited, and actually even by visual inspection one can see that there are only two feasible solutions. The only alternative to the selected one is to switch the matchings of vertices d and e.

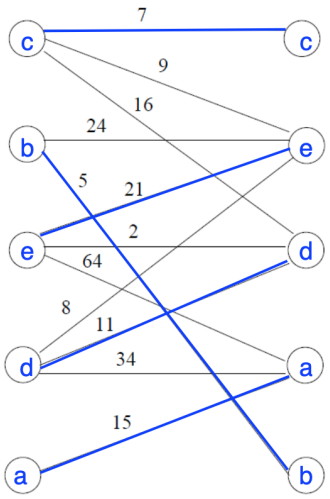

Figure 2: Optimal solution