IEOR 151 – Lecture 8
Solving Matching Games

1 Kidney Exchanges

Recall the model for a kidney exchange[1]: There are k groups of donor-recipients (DR's), and the market is described by a directed graph $G = (V, E)$ with edge weights. There is a vertex $v_i \in V$ for each DR and an edge $e_{i,j} = (v_i, v_j) \in E$ from v_i to v_j if the i-th DR would accept a kidney from the j-th DR. The weight $w_{i,j}$ is the utility obtained to v_i of obtaining v_j's kidney. The clearing problem is the find the maximum weight union of disjoint cycles under the constraint that all cycles can have length less than or equal to a small constant L. It turns out that this problem is NP-hard.

1.1 Edge Formulation Solution

The first approach to solving this problem is to consider an optimization over edges. We start by defining the variable $s_{i,j}$ to be a variable that indicates whether edge $e_{i,j}$ is in some cycle. If $e_{i,j}$ is in some cycle, then $s_{i,j} = 1$; otherwise, $s_{i,j} = 0$ if edge $e_{i,j}$ is not in a cycle. With this definition, we can formulate the solution as the maximizer to the following integer linear program (ILP):

$$
\begin{array}{lr}
\max \sum_{s_{i,j}, \forall e_{i,j} \in E} w_{i,j} s_{i,j} & \text{Sum Weight of Edges in cycles} \\
\text{s.t. } s_{i,j} \in \{0,1\}, \forall e_{i,j} \in E & \text{Variables are Binary} \\
\quad \sum_{e_{i,k} \in E} s_{i,k} - \sum_{e_{k,i} \in E} s_{k,i} = 0, \forall v_i \in V & \text{Conservation Constraint} \\
 & \text{(Outgoing minus Incoming Edges)} \\
\quad \sum_{e_{i,k} \in E} s_{i,k} \leq 1, \forall v_i \in V & \text{Capacity Constraint} \\
 & \text{(Only one outgoing edge in a cycle)} \\
\quad s_{i_1,i_2} + s_{i_2,i_3} + \ldots + s_{i_{L-1},i_L} \leq L - 1, & \text{Path Length Constraint} \\
\qquad \forall L\text{-length paths} &
\end{array}
$$

1.2 Cycle Formulation Solution

There is another approach to solving this problem. Let $C(L)$ be the set of all cycles of length L or less. We start by defining the variable t_c to be a variable that indicates whether cycle c is in the

[1] D. Abraham, A. Blum, and T. Sandholm, "Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges," Proceedings of the 8th ACM Conference on Electronic Commerce, pp. 295–304.

solution. If t_c is in the solution, then $t_c = 1$; otherwise, $t_c = 0$. With this definition, we can formulate the solution as the maximizer to the following ILP:

$$\begin{aligned} &\max \textstyle\sum_{c \in C(L)} w_c t_c && \text{Sum Weight of Cycles} \\ &\text{s.t. } \textstyle\sum_{c: v_i \in c} t_c \leq 1, \forall v_i \in V && \text{Each vertex in at most one cycle} \\ &\quad t_c \in \{0,1\}, \forall c \in C(L) && \text{Variables are Binary} \end{aligned}$$

1.3 Comparison of Formulations

There are some interesting facts about these two formulations. The edge formulation can be solved in polynomial time when there are no constraints on the maximum cycle size L, and the cycle formulation can be solved in polynomial time when the cycle size is at most 2. Also, the LP relaxation of the cycle formulation *weakly dominates* the LP relaxation of the edge formulation, meaning that the LP relaxation of the cycle formulation gives results that are in the worst case as good as those provided by the LP relaxation of the edge formulation. This is important because solving the ILP's for a kidney exchange at the nationwide scale is difficult because of memory and computational limitations, and so efficient algorithms make use of the LP relaxations. Lastly, for a graph with m edges, the edge formulation requires $O(m^3)$ constraints and the cycle formulation requires $O(m^2)$ constraints.

2 National Resident Matching Program

Recall our model for the NRMP. There are m residency programs and n applicants. We will denote the i-th applicant as v_i and the j-th program as p_j. Let s_i be the number of open positions in p_i, let a_i be the number of programs listed by the i-th applicant, and let b_i be the number of applicants listed by the i-th program. We will denote the preferences as a list from most to least preferred. For instance, the preferences of the v_i are given by $(p_{v_i[1]}, p_{v_i[2]}, \ldots, p_{v_i[a_i]})$ and the preferences of the p_i are given by $(v_{p_i[1]}, v_{p_i[2]}, \ldots, v_{p_i[b_i]})$. An important feature of this model is that preferences are unique in the sense that no program or applicant can be preferred at the same level. The problem is to match applicants to positions, so that no applicant matches to more than one position and so that preferences are maximized in an appropriate sense.

Let $I = (1, 2, \ldots, n)$ be a list, then the matching algorithm is given by the following:

1. While I is nonempty, remove the first element of I
    (a) For $j = 1 \ldots a_i$, attempt to place v_i into $p_{v_i[j]}$
        - If $p_{v_i[j]}$ has an empty spot and $p_{v_i[j]}$ has v_i in its list of preferences, then denote $v_i \leftrightarrow p_{v_i[j]}$ a tentative match.
        - If $p_{v_i[j]}$ does not have an empty spot, but there is a "least-preferred" tentative match $v_k \leftrightarrow p_{v_i[j]}$ such that v_i is preferred over v_k by $p_{v_i[j]}$; then remove the tentative match $v_k \leftrightarrow p_{v_i[j]}$, add the tentative match $v_i \leftrightarrow p_{v_i[j]}$, and append k to I.

2. Once I is empty, finalize the matches.

An example taken from literature provided by the NRMP can help clarify the algorithm. Suppose the applicants' preferences are given by:

Anderson	**Chen**	**Ford**	**Davis**	**Eastman**
1. City	1. City	1. City	1. Mercy	1. City
	2. Mercy	2. General	2. City	2. Mercy
		3. Mercy	3. General	3. General

Suppose that each program only has two open positions, and that the preferences of the programs are given by

Mercy	**City**	**General**
1. Chen	1. Eastman	1. Eastman
2. Ford	2. Anderson	2. Anderson
	3. Chen	3. Ford
	4. Davis	4. Davis
	5. Ford	

Then the final match results are given by

Mercy	**City**	**General**
1. Chen	1. Eastman	3. Ford
	2. Anderson	4. Davis