Overview of Lectures 11 and 12: Organ Exchange Optimization

These lectures are focused on finding solutions to the clearing problem in the static case. The first lecture looks at recent attempts to formulate the clearing problem in more compact ways. The second lecture deals primarily with the uncertainty associated with the true existence of edges in the compatibility graphs, while also considering the impacts on fairness to sensitized patients of the proposed improvements to deal with this uncertainty.

Details

0.1 Batch Clearing of Organ Exchanges - John Dickerson

First... We finish up some notes from the end of last week. We briefly talk about how there may soon be a liver exchange. The possibility of combining the kidney and liver exchanges is also mentioned. Though there are some ethical issues with combining the exchanges, e.g., donating a liver is more dangerous than donating a kidney. Because of this, there are not really any altruistic donors in a liver exchange. One idea to deal with this is to find long cycles in the liver pool, cycles that couldn’t possibly be used, and start a chain for it by coming from kidney exchange. We are just using the end of a kidney chain and asking if they would be willing to give a liver to continue it (again, ethical issues). We also mention how complicated a lung donation exchange would be, because one patient needs two separate donors.

Now on to today’s notes:

The clearing problem is to find the best disjoint set of cycles of length at most L, and chains (maybe with a cap K). This lecture only considered static matching, and the next lecture will look at dynamic matching.

How fast can we find this best set of cycles? First, consider the special case where $L = 2$. With only 2-cycles, we can translate our graph to an undirected graph by turning all 2-cycles into undirected edges and cutting all other edges. There are then polynomial time algorithms for maximum matching on undirected graphs.

Next, consider the special case where $L = \infty$. Here, we translate our original graph into a bipartite graph, splitting the donors from the patients. We put edges with 0 weight between a donor and patient if the donor cannot give to the patient. If the donor can give to the patient, then we add an edge with weight $w_e$. Our matching can then be formulated as a maximum weight perfect matching on a bipartite graph, which also has polynomial time algorithms.

Finally, we consider the general case. This is NP-hard via reduction from 3D-matching, and we give an overview of how that works. The 3D-matching problem can be formulated as... Given disjoint sets $X, Y, Z$ of size $q$ and a set of triples $T \subseteq X \times Y \times Z$, is there a disjoint subset $M \subseteq T$ of size $q$?
In order to reduce to 3D matching, we first construct a gadget for each \( t_i = \{x_a, y_b, z_c\} \) in \( T \), where gadgets intersect only on vertices in \( X \cup Y \cup Z \). The gadgets look like this:

The construction only has 3-cycles and L-cycles, where short cycles (i.e., 3-cycles) are disjoint from the rest of the graph by construction. Thus, given a perfect cover (by assumption), gadgets either contribute according to \( t_i \) in \( M \) or \( t_i \) not in \( M \). Thus there is a perfect matching in the original 3D matching instance.

We will mention two approaches to solving this problem. The first is via edge formulation and the second is via cycle formulation. The basic approach to edge formulation [Abraham et al. 2007] involves assigning a binary variable \( x_{ij} \) to the direct edge from each \( i \) to \( j \). Then

\[
\text{Maximize} \quad u(M) = \sum w_{ij} x_{ij}
\]

\[
\text{Subject to} \quad \begin{align*}
\sum_j x_{ij} &= \sum_j x_{yi} & \text{for each vertex } i \\
\sum_i x_{ij} &\leq 1 & \text{for each vertex } i \\
\sum_{1 \leq k \leq L} x_{i(k)(i+1)} &\leq L-1 & \text{for paths } i(1) \ldots i(L+1)
\end{align*}
\]

(no path of length L that doesn't end where it started – cycle cap)

Currently, the best edge formulation technique comes from [Anderson et al. PNAS-2015]. The approach builds off of the prize-collecting traveling salesperson problem; in the TSP variant, the salesperson is allowed to skip a city by paying a penalty. The patient-donor pairs can be thought of as the cities to visit, and the penalty being incurred when pairs are left unmatched.
They maintain decision variables for all cycles of length at most $L$, but build chains in the final solution from decision variables associated with individual edges. Then, an exponential number of constraints could be required to prevent the solver from including chains of length greater than $K$; these are generated incrementally until optimality is proved. To get a tighter IP formulation, one can leverage cut generation from the PC-TSP literature.

The basic cycle formulation assigns a binary variable $x_c$ to each feasible cycle or chain $c$. Then

$$\textbf{Maximize} \quad u(M) = \sum w_c x_c$$

$$\textbf{Subject to} \quad \sum_{c : i \in c} x_c \leq 1 \text{ for each vertex } i$$

Since we have a variable for every feasible cycle or chain, the number of variables is $O(\max |P|^L, |A||P|^{K-1})$, too large to write down in reasonable scenarios. Thus, we need a method for searching only a small portion of the space.

The method used by [Abraham et al. EC-07] is branch-and-price [Barnhart et al. 1998]. In general, branching involves selecting the fractional columns and fixing their values to 1 and 0 respectively. Then you fathom the search node only if it is no better than the incumbent. The LP relaxation can then be solved using column generation.

In the case of solving our clearing problem, we begin with a restricted LP $P'$, which contains only a small subset of the variables (i.e., cycles). We then solve $P'$ and add more variables until $\text{OPT}(P') = \text{OPT}(P)$. Choosing variables to add is done by solving the pricing problem. Abraham et al. solved using a depth-first search, as the general pricing of cycles and chains is NP-Hard, via reduction from the Hamiltonian path problem.

Student question: How do we know which cycles to start with? Answer: This is heuristic, can use ML to figure out promising cycles to start with. One can also have measures of how good the tree is going, and you may start over if it is going poorly. But the problem is NP-hard, so the worst case is bad.

In considering whether to use the edge or cycle formulation, it is a matter of a tradeoff between number of constraints and number of variables. The edge formulation has $O(|E|^L)$ constraints and $O(|V|^2)$ variables, while the cycle formulation has $O(|V|)$ constraints and $O(|V|^L)$ variables.

The newest work in this area finds more compact formulations that do not have exponential variables or constraints. The extended edge formulation (EEF) proposed by [Constantino et al. EJOR-14] involves creating $F$ copies of the compatibility graph, where $F$ is the upper bound on the number of cycles in a final matching. Then in each copy of the graph, you search for a single cycle or chain, keeping cycles/chains disjoint across graphs. We can then write a more compact LP:
The position-indexed edge formulation (PIEF), proposed by [Dickerson, Manlove, Plaut, Sandholm, Trimble EC-16] builds on the EEF. In the IPEF, only \(O(|V|)\) copies of the graph are created, with 1 binary variable per edge per copy. At most once cycle per graph copy is allowed, and the positions of the edges in cycles are tracked for LP tightness.

However, chains are important... which brings us to the position-indexed chain-edge formulation (PICEF) [Dickerson et al. EC-16]. The idea for PICEF was to enumerate all cycles but not all chains, since the cycle cap will be small and the chain cap will be large in practice. To accomplish this, they track not just if an edge is used in a chain, but where in a chain an edge is used. This ends up requiring \(O(K|V|) = O(|V|^2)\) constraints. Then

\[
\begin{align*}
\text{Maximize} & \quad u(M) = \sum_{ij \in E} \sum_{k \in K(ij)} w_{ik} y_{ijk} + \sum_{c \in C} w_c z_c \\
\text{Subject to} & \quad \sum_{ij \in E} \sum_{k \in K(ij)} y_{ijk} + \sum_{c \in C} z_c \leq 1 \quad \text{for every } i \text{ in Pairs} \\
& \quad \sum_{ij \in E} y_{ij} \leq 1 \quad \text{for every } i \text{ in Altruists} \\
& \quad \sum_{ij \in E} y_{ijk} + \sum_{k \in K(ij)} y_{ijk} \leq 1 \quad \text{for every } i \text{ in Pairs} \\
& \quad \sum_{ij \in E} y_{ijk+1} + \sum_{k \in K(ij)} y_{ijk} \leq 0 \quad \text{for every } i \text{ in Pairs} \\
& \quad \text{and } k \text{ in } \{1, \ldots, K-1\}.
\end{align*}
\]

Unfortunately, this does not completely rid us of the issue of there being too many variables in the cases of particularly dense graphs or if longer cycle caps are allowed. This can be solved via branch and price, and it works especially well since one only needs to price cycles, not chains! In fact, PICEF is the first branch-and-price-based model with provably correct polynomial-time pricing.
These models were tested on real and simulated match runs from the US UNOS exchange (with 143+ transplant center) and UK NLDKSS (with 20 transplant center). The results on the original data form the UNOS and NLDKSS are shown below, followed by the results from simulated data where the number of altruists is increased.

**UNOS: 286 match runs**

**NLDKSS: 17 match runs**

**0.2 Managing Short-term Uncertainty in Exchanges (w/ Some Fairness)**

- John Dickerson

First... We finish up some notes from the end of last lecture: Is life always so NP-hard?

The clearing problem can be done in polynomial time if we assume that graphs are constructed from a constant number of if statements (which is true in real life). The issue here is that the constant
on the polynomial time algorithm is not at all reasonable in practice. The current (more) compact graph representation from [Dickerson et al. arxiv:1605.07728] is described below.

A graph $G = (V, E)$. For each patient-donor pair $v_i$, there are attribute vectors $d_i$ and $p_i$ such that the $q$th element of $d_i$ (and respectively, $p_i$) takes on one of a fixed number of types. Call the set of possible types $\Theta$. For example, the set of types could be blood types, so $\Theta = \{O, A, B, AB\}$. Then, given a compatibility function $f: \Theta \times \Theta \rightarrow \{0, 1\}$ that uniquely determines if an edge between $d_i$ and $p_j$ exists, we can create any compatibility graph (for large enough vectors in $D$ and $P$). Altruists are patient-donor pairs where the patient is compatible with all donors, so chains become cycles.

Now, given a constant cycle cap $L$ and constant $|\Theta|$, the clearing problem can be accomplished in polynomial time.

Another nice property of this representation is that flipping attributes is easy. We may want to pay some cost to flip an attribute if, for example, we consider the ability to immunosuppress some traits of a patient in order to increase transplant opportunity, at the expense of the patients short-term health. Formally, $\forall \theta, \theta' \in \Theta$, $c: \Theta \times \Theta \rightarrow R$ is the cost of flipping $\theta$ to $\theta'$. Under the same conditions, the Flip-and-Cover problem can also be done in polynomial time.

Now we want an upper bound on the number of bits needed to represent the graph; we have two theorems. For any $n > 2R$, there exists a graph on $n$ vertices that is not $(k, 0)$-representable for all $k < n$. And the $(k, t)$-representation problem is NP-complete, which can be proven via reduction from 3SAT.

Now on to the day’s notes! This lecture focuses on short-term uncertainty. This is motivated by the fact that only about 10 percent of matches actually result in transplantations.

One way to deal with this is to encode probability of transplantation (instead of only feasibility). So on a compatibility graph, each edge $e$ has its associated weight $w_e$, but now also a success probability $q_e$. Thus the discounted utility of a cycle $c$ would be $u(c) = \sum w_e \cdot \prod q_e$. Our new discounted clearing problem is to find the matching $M^*$ with the highest discounted utility, where $u(M) = \sum u(c)$.

There is an ”in the large” result: For all $q \in (0, 1)$ and $\alpha, \beta > 0$, given a large $G(n, \alpha n, \beta/n)$, w.h.p. there exists some matching $M$ s.t. for every maximum cardinality matching $M$, $u_q(M) \geq u_q(M) + \Omega(n)$. Unfortunately, the unrestricted discounted maximum cycle cover problem is NP-hard, as it can also be reduced from 3D matching. However, the restricted discounted maximum cycle cover problem is solvable in polynomial time for $L = 2$.

At the current size of exchanges, the current solution is feasible. And after running some simulation experiments, we know that in theory and practice, this method can help the global bottom line by considering post-match failure. The pertinent question is then, are we hurting certain individuals in the process?

Let’s briefly consider some fairness rules we may want to employ in practice for kidney exchanges. One possibility is lexicographic fairness, which finds the best match that includes at least $\alpha$ fraction of highly-sensitized patients. Another possibility is weighted fairness, where we value matching a highly-sensitized patient at $(1 + \beta)$ and that of a lowly-sensitized patient at $\beta > 0$. The weighted fairness has an advantage over lexicographic fairness in that re-weighting is a preprocess, while lexicographic constraints requires an IP to solve.
Another idea for improving the efficiency of these matching is to do some pre-match edge testing. This can be very costly, but feasible if we only query a small fraction of the possible edges. The idea would be to find a maximal matching, then query only those edges that appear in that matching to test if they do exist. We then run the matching again with these edges taken out to find a new maximal matching. This greatly reduces the risk of edge failure, while keeping the number of edge existence queries low.

**Even 1 or 2 extra tests would result in a huge lift**