Recursive Formulation

The recursive algorithm aims to maximize the total weight of selected exchanges in a kidney exchange graph. The formulation is as follows:

$$\begin{aligned}
\text{Maximize} \quad & \sum_{e \in E} w_e y_e \\
\text{Subject to:} \quad & \sum_{e \in \text{out}(v)} y_e \leq \sum_{e \in \text{in}(v)} y_e, \quad \forall v \in P \\
& \sum_{e \in \text{out}(v)} y_e \leq 1, \quad \forall v \in NDD \\
& \sum_{e \in \text{in}(v)} y_e \leq 1, \quad \forall v \in P \\
& y_e \in \{0, 1\}, \quad \forall e \in E
\end{aligned}$$

Explanation of Variables

- E: Set of directed edges / arcs representing potential kidney exchanges.
- w_e: Weight (or utility when we extend this) associated with edge e, representing the benefit of performing that exchange. This can be thought of as the benefit to the pair receiving the kidney.
- y_e: Binary decision variable, where $y_e = 1$ if edge e is selected (i.e., the exchange is conducted), and 0 otherwise.
- P: Set of patient-donor pairs.
- NDD: Set of Non-Directed Donors (altruistic donors).
- $\text{in}(v)$: Set of edges entering node v.
- $\text{out}(v)$: Set of edges leaving node v.

Constraints Explanation

1. **Flow Conservation for Pairs:** Ensures that if a patient-donor pair donates a kidney (outgoing), they must also receive one (incoming).
2. **Outgoing Limit for NDDs:** Each altruistic donor can start at most 1 chain, so outgoing flow is limited to 1 (they can only donate 1 kidney).
3. **Incoming Limit for Pairs:** Each patient can receive at most 1 kidney.

Recursive Algorithm Process

The algorithm follows these steps:

1. Solve the integer program using the current constraints.
2. If the solution contains a cycle longer than the allowed length $|C|$, add the following constraint to eliminate it:

$$\sum_{e \in C} y_e \leq |C| - 1$$

3. Re-solve the modified problem.
4. Repeat the process until no long cycles are present in the solution.

This iterative process is where the **recursive** nature comes in: the algorithm recursively adjusts the feasible region by adding constraints to eliminate undesired cycles. Note: I have still not figured out exactly how these are generated...