Extreme Optimization 2

Ilija Wan-Simm, Brooke Stanford, Trey Whitehead, Lucy Leel

December 8, 2023

1 Interpreting the Swaps-Only Solution

1.1 AMPL Formulation

See attached submission zip for .mod, .dat, and .run files!

1.2 Analysis

Number of Pairs	Time (seconds)	# of Transplants	% of Pairs	MIP Iterations	Branch & Bound Nodes
10	0.124133	2	0.2	0	0
50	0.132829	28	0.56	24	0
100	0.150748	56	0.56	27	0
250	0.354171	160	0.64	294	0
500	1.01783	316	0.632	621	0
1000	3.21717	612	0.612	1311	0
2000	11.1088	1192	0.596	3037	0
5000	302.67	2976	0.5952	8030	0

Figure 1: Table of results for timings through number of pairs

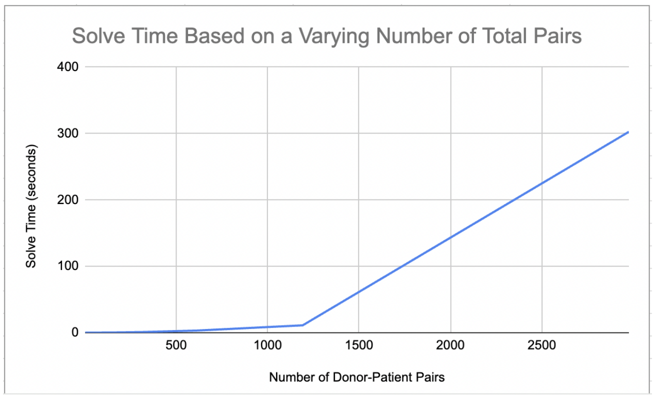


Figure 2: Graph showing relationship between solve time and number of donor-patient pairs

As we can see from the graph above, when we increase the number of donor-patient pairs, the solve time also increases. For example, the running time increases from 0.124 seconds for 10 pairs to 302.67 seconds for 5000 pairs. However it seems that this isn't a linear relationship, and the solve time does not increase proportionally as the number of pairs increases: the solve time increases considerably with high pair numbers, suggesting the program struggles with iterating through so many constraints as quickly as possible.

The "# of Transplant" column indicates the number of patient-donor pairs that resulted in successful matches. This number generally increases with the number of pairs, suggesting that the algorithm is able to find more matches as the data set size grows. However, it's worth noting that the percentage of pairs matched (% of Pairs) seems to decrease slightly as the data set size increases.

There seems to be a trade-off between running time and solution quality, as indicated by the decreasing percentage of pairs matched. It's possible that for larger instances, the algorithm is spending more time searching for an optimal solution but may not necessarily improve the percentage of matched pairs significantly.

2 Edge-Based Formulation

2.1 The Linear Program

Prior to initializing our linear program, let's identify our relevant sets, parameters, and variables.

Sets:

- E: set of compatible edges.

Decision Variables:

- $X_{ij} \in E$: A binary variable.$X_{ij} = 1$ if pair i offers a kidney to patient in pair j. $X_{ij} = 0$ otherwise.

Parameters:

- P : number of pairs in kidney exchange program.

Below is our linear program:

Maximize

$$\sum_{i,j \in E} x_{ij}$$

Such that

$$\begin{aligned}
&(1)\sum_{j \in E} x_{ij} \le 1 \quad \forall i \in E,\\
&(2)\sum_{i \in E} x_{ij} \le 1 \quad \forall j \in E,\\
&(3)x_{ji} = x_{ij} \quad \forall i, j \in E,\\
&(4)x_{kd} + x_{dn} + x_{ny} \le 2, k \ne d \ne y \in E\\
&(5)x_{ij} \in [0, 1] \quad \forall i, j \in E.
\end{aligned}$$

Constraint Justifications:

For constraints (1) and (2), a donor can only give a kidney to one patient and a patient can only receive from a donor. We chose constraint (3) as it ensures a pairwise exchange. Finally, constraint (4) ensures that there are no cycles of more than 3.

2.2 Do we Agree?

The group agrees!

2.3 Counting the Variables and Constraints

- **Variables:** For every 1 potential swap there are $2\binom{p}{2}$ binary x_{ij} variables. We have any combination of two pairs out of a total number of pairs p, hence $\binom{p}{2}$. We multiply by 2 because both x_{ij} and x_{ji} exist as variables.
- **Constraints:** There are 3 constraints for every variable x_{ij} in the formulation, and 1 constraint for every permutation of 4 pairs to ensure no cycles of more than 3. Thus the constraint is $\frac{3p!}{(p-2)} + \frac{p!}{(p-4)}$.

2.4 Formulation Problems

The major potential problem is the factorial increase in constraints and variables as the total number of pairs participating in the exchange system increases. This will lend to much slower run and solve times for this formulation which means there is some inefficiency in an edge-based formulation.

2.5 Graph Visulization

Below are the graphs representing cycles in the exchange and another of all of the matches in the input compatibility graph:

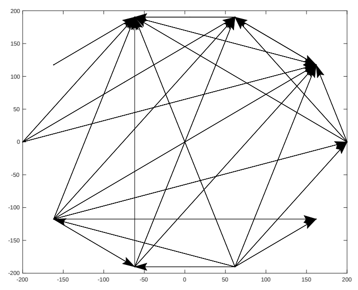


Figure 3: N=10: cycles in the exchange

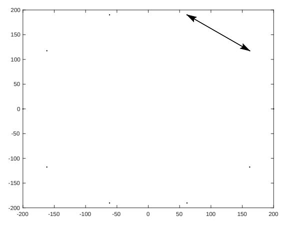


Figure 4: N=10: input compatibility graph

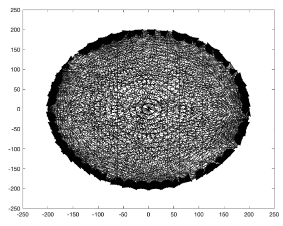


Figure 5: N=50: cycles in the exchange

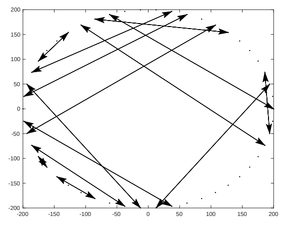


Figure 6: N=50: input compatibility graph

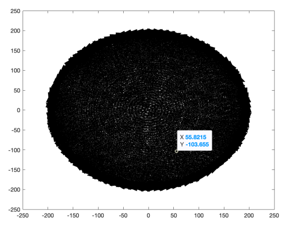


Figure 7: N=100: cycles in the exchange

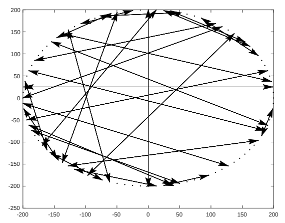


Figure 8: N=100: input compatibility graph

3 Cycle-Based Formulation

3.1 The Linear Program

Prior to initializing our linear program, let's identify our relevant sets, parameters, and variables.

Sets:

- PAIRS: Set of all possible donors and patients to be included within cycles.
- TWOS: Set of all possible 2-person cycles.
- THREES: Set of all possible 3-person cycles.

Decision Variables:

- X_{ij}: A binary decision variable corresponding to 2-person cycles. If a cycle can be validated, X takes the value of 1. Else, it is null.
- Y_{ijk}: A binary decision variable corresponding to 3-person cycles. If a cycle can be validated, Y takes the value of 1. Else, it is null.

Parameters:

- num: Corresponds to the number of donors and patients involved in the kidney transplant program; in other words, it determines the size of the PAIRS set. This directly implies that each element in the set PAIRS represents a unique individual in the program.
- match: Represents a binary two-dimensional array indexed by the elements of the PAIRS set. Its entries describe the compatibility between a potential donor, i, and patient, j.

Now, we arrive at the following linear program, in which we will seek to maximize the number of transfer cycles that will occur (and patients that get a kidney). We will elaborate upon each constraint after the following initialization of the LP:

Maximize

$$\sum_{i,j \in \text{TWOS}} X_{ij} + \sum_{i,j,k \in \text{THREES}} Y_{ijk}$$

Subject to

$$
\begin{aligned}
&(1) \quad X_{ij} = X_{ji} \quad \forall i,j \in \text{TWOS},\\
&(2) \quad Y_{ijk} = Y_{kij} \quad \forall i,j,k \in \text{THREES},\\
&(3) \quad Y_{ijk} = Y_{jki} \quad \forall i,j,k \in \text{THREES},\\
&(4) \quad \sum_{j \in \text{PAIRS}:(i,j)\in\text{TWOS}} X_{ij} + \sum_{j\in\text{PAIRS},k\in\text{PAIRS}:(i,j,k)\in\text{THREES}} Y_{ijk} \le 1 \quad \forall i \in \text{PAIRS},\\
&(5) X_{ij} \in [0,1],\\
&(6) Y_{ijk} \in [0,1].
\end{aligned}
$$

Constraint Justifications:

- **Constraint 1: Counting 2-Person Exchanges** This constraint solidifies symmetry between donor pairs. In other words, X_{ij} and X_{ji} are collectively counted as one decision variable representing an exchange.

- **Constraint 2: Counting 3-Person Exchanges Part One** This constraint is equivalent to constraint 1, although it exclusively functions for 3-person systems. Given that a 3-person cycle can be offset not once, but twice, we need Constraint 3 to represent the possibility of the additional offset.

- **Constraint 3: Counting 3-Person Exchanges Part Two** As delineated above, this constraint completes the symmetry requirement for our 3-dimensional decision variable. It ensures that $Y_{ijk} = Y_{kij} = Y_{jki}$ all represent the same cycle.

- **Constraint 4: Limit on Cycle Participation** This constraint dictates that one person, regardless of whether they are involved in a 2- or 3- person cycle, can only participate in one exchange. After all, there are not infinite livers!

- **Constraint 5: X is a Binary Decision Variable** As written in the decision variables section, X_{ij} is a binary decision variable. Either a cycle is valid, which yields a 1, or it is not, which results in a 0.

- **Constraint 6: Y is a Binary Decision Variable** The same condition in Constraint 5 is valid for Y in Constraint 6, albeit in three dimensions.

Other Notes:

Initially, one might question why we are not multiplying Y to scale for the number of expected patients. This problem is mitigated in the execution of our AMPL code in Exercise 5, as we see that transfer is counted correctly per its associated constraints.

3.2 Do we Agree?

The group agrees!

3.3 Counting the Variables and Constraints

- **Variables:** For this linear program, there are two integer cycle lengths: 2 and 3. Similarly to exercise 2, we will have $2 \cdot \binom{p}{2}$ 2-person cycles. The new addition comes from 3-person cycles, which adopts a similar form: $3 \cdot \binom{p}{3}$. Hence, our total number of variables will be equivalent to $N_{cycles} = 2 \cdot \binom{p}{2} + 3 \cdot \binom{p}{3}$.

- **Constraints:** We have $p + \binom{p}{2} + 2 \cdot \binom{n}{3}$ total constraints. Let's break this down. First, p is captured in the limit placed upon the number of cycles each person could theoretically participate in (meaning 1); we derive it from Constraint 4. $\binom{p}{2}$ comes from Constraint 1, and we get the $2 \cdot \binom{n}{3}$ term from Constraints 2 and 3 collectively.

3.4 Formulation Problems

There are several problems that this LP could generate. We have outlined a couple of potential issues in additional detail below:

- **Complex Computations:** Solving large-scale binary optimization problems can become computationally expensive, given a large number of patients and large number of cycles. For example, as the size of our patient population grows, our size of 3-person cycles will grow non-linearly, as more potential exchanges will become feasible. This could wreak havoc on our ability to store our results and compute them in a timely-fashion.
- **Isolated Patients:** Although the edge-based formulation could generate the same problem, it is worthwhile to acknowledge that in this system, there might still be unmatched patients. This is not necessarily a consequence of the formulation, but rather our compatibility matrices themselves. This could be mitigated by a larger population pool, but given the time-consuming nature of increased computations, we could still run into trouble, all the same.

4 LP Relaxation

For proof of concept, we will now construct a simple example to demonstrate that there is a fractional solution in the LP relaxation of the edge formulation that is precluded in the LP relaxation of the cycle formulation.

To do so, we will need to create our own theoretical data set. For our example, we came up with the following compatibility matrix:

$$\text{Theoretical System} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Upon running this compatibility matrix with two new files representing the LP Relaxations of Task 2 and Task 3, respectively, we generated the following results (See Figures 9 and 10):

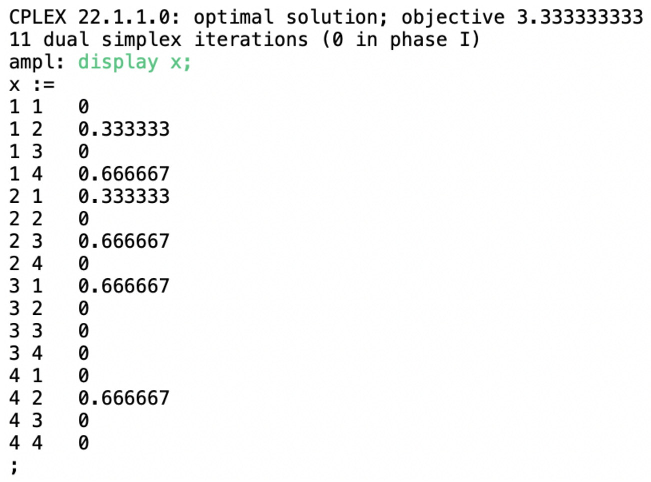
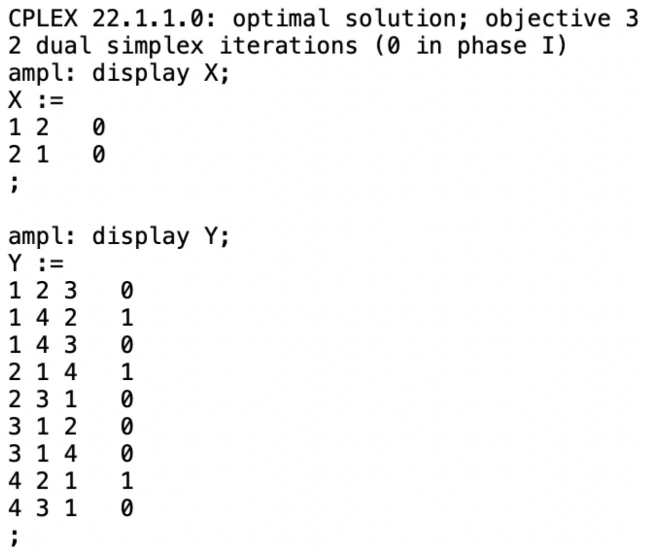

```
CPLEX 22.1.1.0: optimal solution; objective 3.333333333
11 dual simplex iterations (0 in phase I)
ampl: display x;
x :=
1 1   0
1 2   0.333333
1 3   0
1 4   0.666667
2 1   0.333333
2 2   0
2 3   0.666667
2 4   0
3 1   0.666667
3 2   0
3 3   0
3 4   0
4 1   0
4 2   0.666667
4 3   0
4 4   0
;
```

Figure 9: LP Relaxation Results for Edge-Based Formulation

From these outcomes, we can see that, using this data, the edge-based formulation produces results that include fractional solutions, while the cycle-based formulation only consists of whole number solutions. Therefore, we can conclude that the cycle-based model is more efficient than its edge-based counterpart.



```
CPLEX 22.1.1.0: optimal solution; objective 3
2 dual simplex iterations (0 in phase I)
ampl: display X;
X :=
1 2   0
2 1   0
;

ampl: display Y;
Y :=
1 2 3   0
1 4 2   1
1 4 3   0
2 1 4   1
2 3 1   0
3 1 2   0
3 1 4   0
4 2 1   1
4 3 1   0
;
```

Figure 10: LP Relaxation Results for Cycle-Based Formulation

5 AMPL Implementations

We have finally built up enough intuition to implement and share our AMPL formulations.

We will begin with the edge-based system before progressing to the more efficient cycle-based program. Then, to wrap things up, we will run our formulations on data sets of varying sizes and derive a set of conclusions.

5.1 Edge-Based Formulation

See attached submission zip for .mod, .dat, and .run files!

Also see the graphs and tables in 5.3!

5.2 Cycle-Based Formulation

See attached submission zip for .mod, .dat, and .run files!

Also see the graphs and tables in 5.3!

5.3 Interpreting Results!

Now that we have run our two formulations on data sets of varying sizes we are capable of answering the following questions:

Q1: How does the running time increase as a function of the number of patient-donor pairs? How do the running times compare among the pair exchange, the edge-based formulation,and the cycle-based formulation?

Answer:

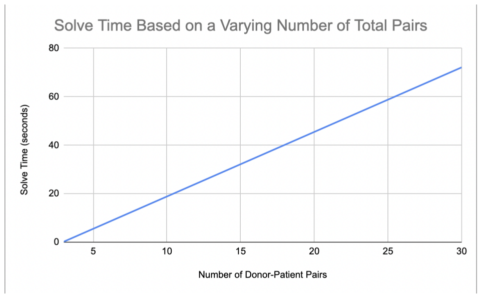


Figure 11: Graph representing Edge-Based Solution

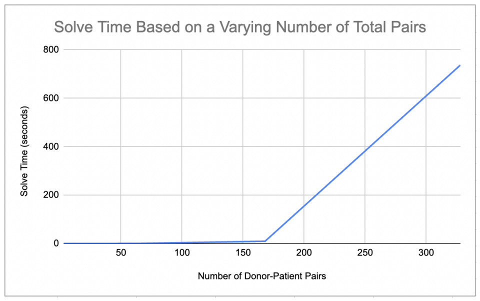
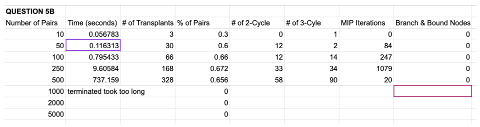


Figure 12: Graph representing Cycle-Based Solution

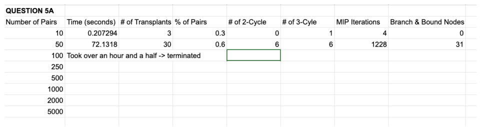
Q2: How many users are getting matched? As a percentage of the number of patient-donor pairs? How many 3-cycles versus 2-cycles are there in solutions? How do these results compare to the results from the pair exchange?

Answer:



QUESTION 5B							
Number of Pairs	Time (seconds)	# of Transplants	% of Pairs	# of 2-Cycle	# of 3-Cyle	MIP Iterations	Branch & Bound Nodes
10	0.056783	3	0.3	0	1	0	0
50	0.116313	30	0.6	12	2	84	0
100	0.795433	66	0.66	12	14	247	0
250	9.60584	168	0.672	33	34	1079	0
500	737.159	328	0.656	58	90	20	0
1000	terminated took too long		0				
2000			0				
5000			0				

Figure 13: Table with Edge-Based Solution Mechanics



QUESTION 5A							
Number of Pairs	Time (seconds)	# of Transplants	% of Pairs	# of 2-Cycle	# of 3-Cyle	MIP Iterations	Branch & Bound Nodes
10	0.207294	3	0.3	0	1	4	0
50	72.1318	30	0.6	6	6	1228	31
100	Took over an hour and a half -> terminated						
250							
500							
1000							
2000							
5000							

Figure 14: Table with Cycle-Based Solution Mechanics

The pair exchange had less MIP iterations and a shorter run-time, as evidenced by the graphs. However, the new formulations had more pairs due to the permitted recycles.

Q3: How is CPLEX solving these formulations? Turn on logging to see the IP's progress over time. Is it spending a long time finding incumbent solutions or proving optimality? Offer a comparison between the formulations.

Answer: CPLEX does not spend most of its time finding incumbent solutions, but instead proving optimality. For edge-based, CPLEX largely used the branch and bound procedure, but for cycle-based, it only used MIP. What is interesting is that, per our graphs above, the cycle-based formulation proved far more efficient.

Q4: Are there other difficulties you encountered? All in all, which formulation scales better?

Answer: Computationally, we ran into issues with larger data sets. In fact, some of the sets took so long to run that we couldn't calculate their run time within the span of our project. It is important to note, however, that the cycle-based method scaled better, per the reasons outlined in the answers above. The cycle-based system handled more data at a faster rate!

6 Thank You!

This has been a fantastic course, and as a team, we are extremely grateful for the efforts of the teaching staff this term. Our final project felt very rewarding to finish, as it represents the culmination of our efforts over the semester. While the lessons of APMTH 121 are mathematical by design, we have embraced the idea of optimizing in life, striving to maximize the value of each day given its challenges, or rather, the external constraints placed upon us. The foremost of which, we consider to be time, and given that our time at Harvard is running short, we are thankful to have had the opportunity to make friends and learn at every step of the way in APMTH 121.