Extreme Optimization 2
Enabling Kidney Exchange

Applied Mathematics 121 — Fall 2019

Due Friday, November 15, 2019

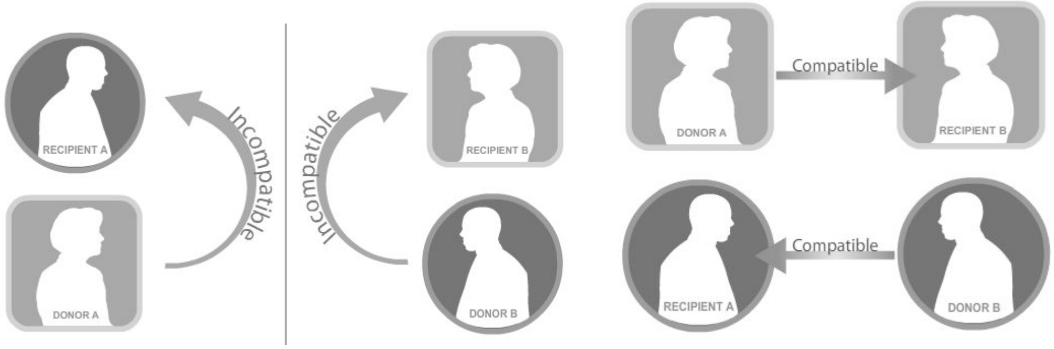
Announcements

- For this project, You are asked to form your own team of three.
- Some ground rules. You are to work with members of your team but not across teams. You are allowed to use any course material that has been provided to you (textbook, lecture notes, section notes, etc), and any other textbooks and general purpose discussions on linear programming and integer programming. But **you may not use the Internet in an attempt to look up kidney-exchange SPECIFIC solutions to this assignment.** It is ok to look online or in textbooks for ways to scale-up IPs in response to parts of the question that ask for how you might use 'non-standard techniques', but you should state any sources. If you have any questions or require any clarifications on said rules, please contact a member of the course staff.
- Empirical analysis is a major component of this assignment. Do *not* underestimate it.
- You can use (at most) one late day on EO 2 if all your team members have unused late days.
- Each team need only turn in one electronic copy of anything asked for. Please ensure that your writeup is coherent and complete (and correct!). Be sure to record the names of all team members. **State what each team member contributed to the project.**
- Work as a team. Have fun!

Goals

This assignment gives you an opportunity to tackle a real-world problem using the techniques you have learned in the course. It will also give you a chance to exercise your creativity and insightfulness.

Contents



(a) Two donor-patient pairs with incompatible matches

(b) Via a swap, both patients receive a compatible transplant from the other patient's willing donor

Figure 1.1: Pair exchange

1 Introduction

1.1 An overview

The kidneys are vital organs that filter out waste from blood. Kidney failure[1] leads to the accumulation of this waste in blood, and if untreated will result in death in a few months. One treatment option is dialysis, a process by which a patient's blood is filtered by an external machine at a hospital. However, patients undergoing dialysis have a low quality of life, and many end up withdrawing from treatment. Furthermore, only 12% of dialysis patients survive more than 10 years.[2]

The preferred treatment for kidney failure is a kidney transplant. Unfortunately, the demand for kidneys far outstrips the supply, and the number of patients on the waiting list for a cadaver kidney has increased steadily from 24,765 in 1993 to over 110,000 today. The median waiting time is between two to five years depending on blood type. In 2005, 4,052 people in the U.S. died while waiting for a life-saving kidney transplant.[3]

In addition to cadaver kidneys, there has been a steady increase in the number of live kidney transplants. Since a normal person has two kidneys and can survive with just one, a healthy willing donor with a kidney that is compatible for the patient can offer a transplant (where a match is based on appropriate blood types, absence of positive match antibodies, and other medical grounds). However, situations often arise in which the kidney of a willing donor for a particular patient is incompatible with the patient, in which case the patient joins (or remains) on the waitlist for a cadaver kidney.

The challenges of finding a willing donor with a compatible kidney and the long waitlist for cadaver kidneys are creating a serious problem.

In recent years, there have been advances in the use of *paired exchange*, in which a pair of patients with incompatible willing donors perform a swap that allows each patient to obtain a kidney from a compatible donor. Figure 1.1 presents an example.[4] In figure 1.1a, we see patient A with willing incompatible donor A and patient B with willing incompatible donor B. However, donor A's kidney happens to be compatible with patient B and donor B's kidney happens to be compatible with patient A. Through a paired exchange (figure 1.1b), both patients receive a kidney.

Such exchanges are in the interest of both patients, and improves the life for both transplant recipients.

[1] Common causes of kidney failure are diabetes and high blood pressure, but in many patients the cause is never found. See http://www.kidneypatientguide.org.uk/site/fail.php

[2] http://www.usrds.org

[3] Data from United Network for Organ Sharing, see http://www.unos.org

[4] Images from Alliance for Paired Donation. See animation at http://www.paireddonation.org/anim.htm

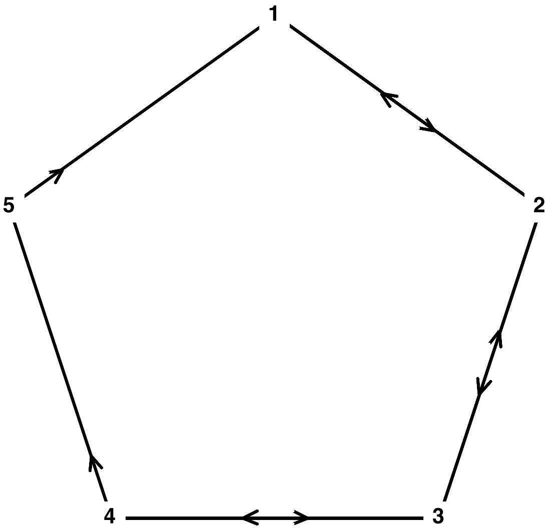


Figure 1.2: An example matching market

Furthermore, since no money is involved, such exchanges are legal. The main reason why such transplants have been rare is because they have been limited to the hospital and regional levels and there isn't a national donor-patient database from which to find matches. However, a unified national exchange can save more lives, and the trend is moving towards a centralized exchange market for kidneys.

1.2 The problem at hand

The United Network for Organ Sharing (UNOS) is working hard with researchers to enable a nation-wide kidney exchange. They have provided us with a data generator based on the extensive nationwide data they have maintained, so the generated patient data are drawn from a realistic instance distribution. The generator takes into account the distribution over blood types, panel reactive antibody measures, gender ratio, and spouse ratio. Given n patient-donor pairs, it outputs a binary matrix c such that $c_{ij} = 1$ if and only if the donor associated with pair i is compatible with the patient associated with pair j. We have provided you with data generated this way for problem instances with as few as 10 patient-donor pairs and as many as 5000 patient-donor pairs.

To think about how patients and donors relate, we can encode this compatibility matrix as a directed graph $G = (V, E)$. Each vertex of the graph represents a patient-donor pair. A directed edge from vertex i to j corresponds to $c_{ij} = 1$ (donor associated with pair i is compatible with patient associated with pair j). We can also attach weights to these edges (perhaps to denote the utility of patient associated with pair j receiving a organ transplant from donor associated with pair i), but let us just assume that all weights are 1. A (directed) cycle in the graph is a set of vertices $v_1, \ldots, v_k$ for which the edges $(v_1, v_2), \ldots, (v_{k-1}, v_k), (v_k, v_1)$ are in the graph. A cycle represents a possible swap, in which the patient associated with pair i receives a transplant from donor $i - 1$ in the cycle (or in the case of $i = 1$ from donor k). Notice that this definition ensures that the donor associated with pair i will donate a kidney to the patient associated with pair j if and only if the patient associated with pair i is to receive a kidney.

A *feasible exchange* is a set of disjoint cycles. To be a bit more concrete, consider the example in Figure 1.2. Here there are five agents. There are 4 cycles in this graph, and they are $c_1 = (v_1, v_2)$, $c_2 = (v_2, v_3)$, $c_3 = (v_3, v_4)$, and $c_4 = (v_1, v_2, v_3, v_4, v_5)$. A set containing any one cycle represents a feasible exchange, but so does the set containing cycles (c_1, c_3), which are disjoint. We say that (c_1, c_3) and (c_4) are maximal exchanges, in the sense that no more cycles can be added while still maintaining feasibility. Notice that under (c_4) all patients would receive a transplant (via a cycle of length 5), and that (c_1, c_3) is the *maximal pair exchange* for this example (all cycles are of length 2).

At first glance, you may conclude that we should implement the maximal exchange (c_4). But implementing an exchange with a cycle of length 5 is no simple matter. Kidney exchange is a sensitive issue; while the market structure is a barter economy, there are particular rules governing its operation. For one, to avoid a donor from backing out once his incompatible friend receives a kidney, all operations must happen simultaneously. With a large cycle this can be a logical nightmare, as each donor and patient requires 3 to 6 doctors, 4 nurses, and 2 operating rooms. For this reason, large cycles are impractical.

2 Enabling Kidney Exchange

As a starting point, we will focus on a "swaps-only" solution and formulate an integer program for the problem.

We can focus our attention to bi-directional edges, that is, links between two patient donor pairs who can perform a swap. Denote the set of such edges by E, and for all $(i, j) \in E$, let binary decision variables $X_{i,j} = 1$ if the donor associated with pair i gives a kidney to the patient ssociated with pair j and 0 otherwise. We have the following integer program:

$$\textit{maximize} \sum_{(i,j)\in E} X_{ij} \tag{1}$$

$$\sum_{j:(i,j)\in E} X_{ij} \leq 1 \qquad \forall i \tag{2}$$

$$\sum_{i:(i,j)\in E} X_{ij} \leq 1 \qquad \forall j \tag{3}$$

$$X_{ij} = X_{ji} \quad \forall (i,j) \in E \tag{4}$$

$$X_{ij} \in \{0,1\} \qquad \forall i,j \tag{5}$$

The first two constraints ensure that each patient receives no more than 1 kidney and each donor gives no more than 1 kidney. The third constraint ensures pair-wise exchanges, and the objective is to maximize the number of pair-wise swaps.

You will now implement formulation (1)–(5) in AMPL. When doing so, you should again make use of the `read` command for reading data from a file and the `print` command for outputing a solution matrix. We have also provided some basic MATLAB code for visualizing graphs; see `graphviz.zip`. Simply edit the top line of the script `graphvisualizer.m` to include the number of patient-donor pairs in the instance you are trying to visualize, and when prompted, select the AMPL output file (a binary matrix whose element (i, j) indicates whether the donor associated with pair i gives a transplant to the patient associated with pair j) and the input file (the raw data containing c_{ij}). To use the script, first place the required files and those provided in the zip file in your MATLAB directory.

If you run the `graphvisualizer.m` script with the correct inputs, MATLAB will display two graphs: one of the cycles in the exchange and another of all of the matches in the input compatibility graph. Note that arrows will point from a donor to a recipeint. These graphs may offer some insight into the solution of the problem and also provide a convenient method for verifying your solution's correctness. However, we do not recommend running this script for the datasets with more than 250 pairs, as the graphs will likely be too dense to be useful (though you can zoom in if desired). You can also save these graphs in standard image format. If you have any questions regarding the visualizer, please feel free to ask us![5]

[5] You are free to use a different software to perform the visualization as well. For example, Mathematica may be a good alternate choice.

Exercise Task 1

1. Implement the above model in AMPL. Create a model file, a data file, a run script, and output files containing the matches in the exchange. You need not create multiple versions of the data file for different instances and instead can just apply minor edits to the data file for running particular instances.

2. Run your model on data files of varying sizes. Describe your findings. In particular, be sure to at least address the following in your writeup:

   - How does the running time change as a function of the number of patient-donor pairs?
   - How many users are getting matched? As a percentage of the number of patient-donor pairs?
   - What can you tell about the way in which CPLEX is solving these instances?

   To answer these questions you will have to set some CPLEX parameters via AMPL to display the relevant information. For this, you should find Chapters 5 and 7 of the CPLEX-AMPL guide to be resourceful, which is available here on the course website.

   In presenting your analysis, you may wish to make use of tables or graphs. You are not required to run the model for every size of input given, but should at least attempt to run on a good range of problem sizes. You are also not required to use the visualizer, but like previously mentioned it is a good error checking tool and the output may look cool.

End Task 1

3 Allowing for 3-cycles

Having considered the problem of "swaps only," you notice that it may be possible to have more matches if we allowed for cycles of size 3. While there are still some concerns over larger cycles, the logistical issues are not limiting for cycles of size 3, and a few such exchanges have occurred in regional markets. In this study, you will aim to come up with and evaluate methods to solve the kidney exchange problem while allowing for cycles of length up to 3.

You are to come up with (**at least**) two formulations, one based on edges and another based on cycles.

Exercise Task 2

Edge-based formulation.

1. Formulate a mathematical model (not using AMPL at this point) to find an exchange that leads to the most patients receiving transplants while only allowing for cycles of length up to 3. In this formulation, let your decision variables be binary variables defined over the set of edges in the graph, such that the variable corresponding to edge (v_i, v_j) is 1 if and only if the donor associated with pair i offers a kidney to the patient associated with pair j.
2. Verify that the formulation is (probably) correct by ensuring that every member of your team agrees with the formulation. (No writeup necessary here.)
3. How many variables and constraints are in your edge-based formulation?
4. What do you think are the potential problems in using this formulation? Consider how well your formulation scales and the associated computation requirements. (It might be helpful to work through 3.1-3.3 first and then answer this).
5. Can you think of ways to address these problems? Read this lecture notes, then (informally) describe a method by which we may be able to solve larger instances than using standard techniques (e.g., other than simply giving your formulation of the problem to CPLEX).
6. Can you think of alternate ways of formulating the constraints of the problem? Describe an alternate formulation, and compare it to your original formulation. Is any formulation stronger than another ('stronger' in the technical sense)? Do you believe one may scale better than the other? Explain.

End Task 2

Exercise Task 3

Cycle-based formulation.

1. Formulate a mathematical model (not using AMPL at this point) to find an exchange that leads to the most patients receiving transplants while only allowing for cycles of length up to 3. In this formulation, let your decision variables be binary variables defined over the set of cycles of length at most 3, such that a variable corresponding to cycle c is 1 if and only if c is a cycle in the solution to the exchange.
2. Verify that the formulation is (probably) correct by ensuring that every member of your team agrees with the formulation. (No writeup necessary here.)
3. How many constraints and variables are in your cycle-based formulation?
4. What do you think are the potential problems in using this formulation?
5. Can you think of ways to address these problems? Again, read this lecture notes and (informally) describe a method by which we may be able to solve larger instances than using standard techniques (e.g., giving the problem to CPLEX).

End Task 3

Exercise Task 4

1. Can you construct a simple example to show 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?
2. (**extra credit**) Show that the cycle formulation is stronger than the edge formulation, i.e. show that any solution to the LP relaxation of the cycle formulation is also a solution to the LP relaxation of the edge formulation.

(Note: focus on the rest of the assignment before doing this exercise. In particular, gather some experimental results that will help guide your thinking.)

End Task 4

To understand which formulation is better and how well these formulations perform in practice, there is no better way to start than to implement the formulations in AMPL.

Exercise Task 5

1. Implement your edge-based formulation in AMPL. Create a model file, a data file, a run script, and output files containing the matches in the exchange. You need not create multiple versions of the data file for different instances and instead can just apply minor edits to the data file for running particular instances. Run it on some small instances (feel free to hand construct examples as necessary) to make sure it is working the way you intend it to.

2. Implement any alternate edge-based formulations you have come up with in AMPL.

3. Implement your cycle-based formulation in AMPL. Create a model file, a data file, a run script, and output files containing the matches in the exchange. You need not create multiple versions of the data file for different instances and instead can just apply minor edits to the data file for running particular instances. Run it on some small instances (feel free to hand construct examples as necessary) to make sure it is working the way you intend it to.

4. Run your models on data files of varying sizes. Describe your findings. In particular, be sure to at least address the following in your writeup:

   - 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(**s**), and the cycle-based formulation?
   - 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?
   - 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.
   - Are there other difficulties you encountered? All in all, which formulation scales better?

   In presenting your analysis, you may wish to use tables or plot some data on graphs. You are not required to run the model for every size of input given, but should at least attempt to run on a good range of problem sizes. You are also not required to use the visualizer, but it may give you some intuition (you will have to do a little work to use it with the cycle-based formulation, though).

5. Provide intuition for the scability of your formulations. If certain formulations can be used to solve larger instances than others, explain why you think this is the case. Avoid using general statements such as "this formulation is stronger", but instead say why you think one formulation is stronger, solves more quickly, or scales better, etc.

6. Attempt to improve the solve time and scalability of the two formulations by tuning the parameters within CPLEX. There are many parameters that can be tuned, but stick with parameters that govern general solve strategies and don't worry about tuning the finer details. Provide a brief discussion of what you tried and why you tried it. You will want to experiment both with the parameters for generating cuts and for the branch and bound search.

End Task 5

4 Thinking ahead

Exercise Task 6

1. Can you identify the major bottlenecks in applying optimization techniques to enabling large-scale kidney exchange? What do you think can be done about it? Try to be as specific as you can in your answer.

2. Do your formulations generalize to solving the problem of finding all cycles of length at most L? If so, provide the generalization. If not, explain why not, and attempt to come up with a formulation that will generalize.

3. Close with a few words about the state of kidney exchange optimization technology (as developed by you). If the news is good, let us know how good. If the news is grim, let us know what to hope for in the future.

End Task 6

5 Turning in extreme optimization.

You must submit an electronic submission by 5pm Friday, November 15, 2019. Please submit 2 files. For the first file, please attach a pdf of your writeup that describes and explains how you approached the problem, what you set out to model, complete mathematical formulations, and results and accompanying analysis on test data. For the second file, gather all AMPL model and data files you have created and any outputs or graphs you have generated. Create a README file that will let us know which files refer to which problems. Put all these files in a zip archive. Submit the zipfile and the pdf to GradeScope under the "Extreme 2" folder. Make sure you note in your zipfile, in the README, or some other obvious place, the names of your group members and a team name. You can choose any team name you'd like as long as it isn't boring.

Congratulations on completing your second AM121 extreme optimization assignment!