15326 - Computational Microeconomics

Assignment 3: Kidney exchange and voting (due Nov. 1 before 5pm)

Please read the rules for assignments on the course web page (`http://www.cs.cmu.edu/~15326-f24/`). Use Piazza for questions and Gradescope to turn this in. For questions where this is appropriate, always hand in both code and output, typically .mod and .out files (and do not simply put everything in a .pdf).

Please use clear variable names and write comments in your code where appropriate (you can put comments between `/*` and `*/`, or start a line with `#`).

Please see Homework 1 for details about getting set up with GLPK, making a directory for this homework, etc.

For question 1, hand in discounted_kidney.mod and discounted_kidney.out (both using the second instance). For questions 2 and 3, your submission should be a PDF describing your solutions. For question 4, hand in modified_borda.mod and modified_borda.out.

Kidney exchange with discounting. In this problem, you will complete a version of the *third* integer programming formulation for solving the kidney exchange problem that is given in the slides on kidney exchange. This one is convenient because it does not require you to generate all paths/cycles of the appropriate length, and it will also be natural to extend it in the way described below.

A natural interpretation of that formulation is as follows. Suppose that the only reason to limit k is the number of operating rooms and medical personnel available. (In the real world, there are other reasons for limiting the length of a cycle, as we discussed in class, including the possibility that a donor or patient in the end isn't available after all, and the whole cycle collapses; but for the purposes of this assignment, we will ignore that, and think only about a logistical limit on the number of surgeries that can be performed in (say) one day.) In particular, with this formulation, if the limit on the number of simultaneous transplants is $k = 4$, then for the event on day 1 ($t = 1$), it is also possible to schedule *two* 2-cycles on day 1. That is, for this formulation, the limit k is more naturally interpreted as a limit on the number of simultaneous transplants – though this of course then also becomes a limit on the length of a

cycle, because all transplants in a cycle must be performed simultaneously.

With this interpretation in mind, we will extend the formulation to capture another aspect, as follows. Other things being equal, we would like to perform the transplants sooner rather than later, to improve the patients' quality of life as soon as possible. We will capture this through *discounting*, which makes transplants that are scheduled further in the future less valuable from the perspective of our objective function. Specifically, we will have a *discount factor* $\delta < 1$ and say that a transplant on day t contributes δ^t to our objective. For example, if $\delta = 0.9$, then a transplant on day 1 contributes 0.9 to our objective value, but a transplant on day 2 contributes only 0.81 to our objective value. (We will not allow transplants on "day 0" – let's say day 0 is the day that we are doing the optimization.) So, for example, if $\delta = 0.9$ and $k = 4$, and all we are trying to schedule is two 2-cycles, then it is better to schedule them both on day 1 and get $0.9 \cdot 4$, rather than spreading the two-cycles out over day 1 and 2 for only $0.9 \cdot 2 + 0.81 \cdot 2$. But if $k = 3$, then unfortunately we have to spread them out over the first two days, one two-cycle each day. (We are not allowed to do only part of a cycle in a day.)

1 (50 points). Please complete the following integer program. The first data section is to test your code on, with the solution provided below. You must turn it in with the solution to the **second** data section, further below.

```
param T;                  # Number of time periods (days)

param k;                  # Max simultaneous transplants per day

param delta;          # Discount factor (0 < delta < 1)

param n;                  # Number of nodes, i.e., (patient, donor) pairs

set N := 1..n;          # Set of nodes; 1..n means the set of integers {1, 2, ..., n}

set Time := 1..T;    # Set of time steps



# Compatibility matrix: 1 if node i can give to j, 0 otherwise

param compat{N, N}, binary;


# Decision variable: x[i,j,t] = 1 if i gives to j at time t, 0 otherwise

var x{N, N, Time}, binary;
```

```
# Objective: maximize the discounted total transplants

maximize DiscountedTransplants:     # YOUR TASK IS TO COMPLETE THIS


# Please refer to the lecture slides for implementing the constraints:

s.t.   # YOUR TASK IS TO COMPLETE THIS




# Data section (Instance 1 for testing)

data;


param T := 5;

param k := 3;

param delta := 0.8;

param n := 5;

param compat:

    1 2 3 4 5 :=

1   0 1 0 1 0

2   1 0 1 1 1

3   1 1 0 0 1

4   0 1 1 0 1

5   1 0 1 0 0;

end;
```

The optimal solution to Instance 1 has DiscountedTransplants = 3.68. Patients 1, 2, and 4 get a transplant on day 1; patients 3 and 5 get a transplant

on day 2. This leads to a total discounted value of $0.8 \cdot 3 + 0.8^2 \cdot 2 = 3.68$.

Next is the data section for Instance 2. You must turn in your code with this one.

```
# Data section (Instance 2 for submitting)

data;



param T := 6;

param k := 3;

param delta := 0.8;

param n := 12;

param compat:

    1 2 3 4 5 6 7 8 9 10 11 12 :=

1   0 1 0 1 0 1 1 0 0 1 0 0

2   1 0 1 0 1 0 0 1 0 0 1 0

3   0 1 0 1 0 1 0 0 1 0 0 1

4   1 0 1 0 1 0 0 1 0 0 1 0

5   0 1 0 1 0 1 0 0 1 0 0 1

6   1 0 1 0 1 0 1 0 0 1 0 0

7   0 1 0 1 0 1 0 1 0 0 1 0

8   1 0 1 0 1 0 1 0 0 1 0 1

9   0 1 0 1 0 1 0 0 1 0 1 0

10  1 0 1 0 1 0 0 1 0 1 0 0

11  0 1 0 1 0 1 0 0 1 0 1 0

12  1 0 1 0 1 0 1 0 0 1 0 1;
```

```
end;
```

Voting. For the purpose of this assignment, we made up a new voting rule that we will call "ModifiedBorda." An alternative j's score under ModifiedBorda is: the smallest number of votes that need to be removed so that j becomes the Borda winner (so lower ModifiedBorda scores are better). Actually, being tied for the Borda win is enough; so really, the goal is to compute the smallest number of votes that need to be removed so that there is no longer any other alternative j' that has a strictly higher score than j. Hence, an alternative's ModifiedBorda score is never larger than the total number of votes, because if we remove all votes, then all alternatives are tied for the win.

Consider the following example with the following four votes:

1. $a \succ c \succ b$
2. $a \succ c \succ b$
3. $b \succ a \succ c$
4. $b \succ c \succ a$

The regular Borda scores of the alternatives are 5 for a, 4 for b, and 3 for c. The ModifiedBorda score of a is 0, because a is one of the Borda winners without removing any votes. The ModifiedBorda score of b is 1, because by removing (e.g.) the first vote, the Borda scores become 4 for b, 3 for a, and 2 for c. The ModifiedBorda score of c is 2, because no single vote can be removed to make c have at least as many Borda points as a (no vote gives a 2 more Borda points than c), and removing (e.g.) the second and third votes will result in Borda scores of 2 for all alternatives. In this particular example, it so happens that the ModifiedBorda ranking of the alternatives is the same as the original Borda ranking: a wins, b is second, and c is third.

2. (5 points.) Give a quick argument why the winning alternative(s) will always (in any example) be the same under Borda and ModifiedBorda.

3. (10 points.) Give an example, i.e., a set of votes, in which the Borda ranking and the ModifiedBorda ranking of the alternatives are *not* the same, even though the winners (top-ranked alternatives) are. (If one of the rules produces a tie in the ranking and the other does not, we will count that as not the same, so that's enough. Try to make your example as small as possible—it can be very small!)

4. (35 points.) Create an integer program for calculating the ModifiedBorda score of a single candidate as a .mod file. To make things easy: We will think of the alternative for which we want to calculate the score as special, so we will distinguish between that alternative and the "other" alternatives. We will also assume that we are explicitly given as input the Borda score that each alternative receives from each vote. For vote i and the special alternative, we will refer to the Borda score that the special alternative gets from i as

`special_score[i]`

For vote i and some other alternative j, we will refer to the Borda score that j gets from i as

`other_score[i,j]`

For example, for the votes above and treating c as the special alternative (i.e., the one for which we would like to calculate the score), the data part of the file should look as follows:

```
data;
set VOTES := v1 v2 v3 v4;
set OTHER_ALTERNATIVES := a b;
param special_score := v1 1 v2 1 v3 0 v4 1;
param other_score: a b :=
                v1 2 0
                v2 2 0
                v3 1 2
                v4 0 2;
end;
```

You should also test your code on your own example from part 3. But only turn in the output resulting from the data in part 4.