CSE 516A: Homework 1

Due: February 14, 2017

Note: Please keep in mind the collaboration policy as specified in the course syllabus. If you discuss questions with others you **must** write their names on your submission, and if you use any outside resources you **must** reference them. Please submit your writeup, but not your code. However, we may ask you to email us your code, and if you do not then you will not receive any credit. Finally, keep in mind that homework (in hardcopy) is due **at the beginning of lecture.** Note that several of the questions are quite open-ended. You will be graded in part on the quality of your analysis and in part on the quality of your writeup, so please write your answers up carefully. There are five questions on three pages.

1. (60 points) **The Assignment Problem:** In this problem you will explore the properties of the assignment problem and compare two different ways of solving it. We have broken it up into discrete tasks that build on each other.

   (a) (20 points) **Implementation:** Implement the auction algorithm in the language of your choice, using a representation that you deem appropriate. Also implement a general method for encoding assignment problems as linear programs using the linear programming modeling language of your choice (we recommend GLPK). Test these on small instances to make sure that you get correct answers. Once you're sure that your algorithms are working, solve the following instance of the assignment problem with ten agents and ten objects (agents are rows, objects are columns):

	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10
A1	89	42	0	2	24	20	40	37	30	77
A2	66	75	9	59	69	66	52	14	85	36
A3	82	68	0	81	36	25	48	53	11	68
A4	6	96	82	53	17	70	26	12	91	82
A5	34	86	22	18	66	73	82	88	18	36
A6	90	43	43	93	80	96	12	28	74	93
A7	19	75	30	48	31	76	84	29	20	15
A8	29	73	88	9	36	40	40	19	1	45
A9	77	31	6	68	36	40	22	43	27	61
A10	70	21	2	89	30	91	66	74	79	92

   Report the actual assignment and the total value of that assignment.

   (b) (20 points) **Random problems and assignment values:** Now implement a way of generating a random assignment problem given two parameters, n and M, where n is the number of agents (there should be an equal number of objects), and the value of each assignment is an integer sampled uniformly at random between 0 and $M - 1$

(or 1 and M if you prefer). You should figure out a way to feed this problem to both the auction algorithm solver and the LP solver (if you are using GLPK, the best way to do this is to generate a separate data file, while keeping the model file to specify the common parts of the model). Now, using either solution technique, compute the per-agent average value of assignments as you increase n, in powers of 2, from 2 to 256, setting M to 100. Average at least 1000 runs for each case of n. Plot the results and include this plot in your writeup. Explain why you see the pattern you see, backing up your claim with any specific evidence that you may want to gather from the instances you generate or the execution of your code.

(c) (20 points) **Timing:** Now, holding the number of agents constant at 256, change M, this time going up in orders of 10, from 10 to 100 to 1000 and all the way up to 10^7. Solve each generated instance using both the auction algorithm approach and the linear programming approach, keeping track of how long each solver takes on average to solve an instance as a function of M. You should run the code for at least 100 instances for any particular M to get a stable estimate. Plot the results for each of the two approaches. Why do you see the results you see? Are they what you expected from the worst case bounds on time discussed in class? Again, you should back up your claims with any specific evidence you deem appropriate from the instances you generate or the execution of your code.

2. (5 points) **What problem?** Suppose I give you a list of numbers $k_1, k_2, \ldots, k_n$. Consider the following linear program:
Maximize $\sum_{i,j} j \cdot k_i \cdot x_{ij}$
Subject to the constraints

$$\sum_j x_{ij} = 1 \quad \forall i$$
$$\sum_i x_{ij} = 1 \quad \forall j$$
$$x_{ij} \geq 0 \quad \forall i, j$$

where the i, j range from 1 to n. What standard problem is this LP solving? What is the appropriate interpretation of the x_{ij} (you can assume they are all 0 or 1 in thinking about this)?

3. (10 points) **Different objectives in stable matching:** Consider a standard stable matching problem with complete preference lists for both men and women. I want to find the stable matching that minimizes the average rank in the preference list of each agent's assigned partner. For example, if there are three men and three women and a stable matching assigns partners ranked 1, 2, and 2 (in their own preference lists) to the three men, and 1, 3, and 3 to the three women, then the average rank is 2. Write down a linear program (using the same conventions as the basic stable matching LP discussed in lecture) for achieving this objective. (What you write down can be an integer program, actually, but the resulting LP relaxation will be guaranteed to have an integral solution as discussed in class).

4. (10 points) **Manipulation through permutation:** Consider a matching market with three men and three women with the following true preferences:

Men			
1	1	2	3
2	2	1	3
3	1	2	3

Women			
1	2	1	3
2	1	2	3
3	1	2	3

Suppose the mechanism is Gale-Shapley with men proposing and that all agents must submit complete preference lists. If the agents are all truthful in the preferences they submit, what matching will results? Is there a woman who can misrepresent her preferences and end up with a more preferred partner, assuming others are truthful? If so, which one, what preferences should she submit, and what will the resulting matching be? If not, why not?

5. (15 points) **Invariance of those matched:** Consider a variation of the stable matching problem discussed in class. Now let there be n men and m women where n is not necessarily equal to m. Assume that each agent has complete, strict preferences over the other side plus the null match (so she or he can express that (s)he would prefer being left unmatched over some possible matches). Note that the notion of a stable matching extends quite easily to this scenario (no blocking pairs is still the key criterion). Prove that the set of agents left unmatched in *any* stable matching is the same.