1.

a)

The following are the assignments determined by the auction algorithm and GLPK in the form of (agent: object)

0: 9

1: 8

2: 3

3: 1

4: 7

5: 4

6: 6

7: 2

8: 0

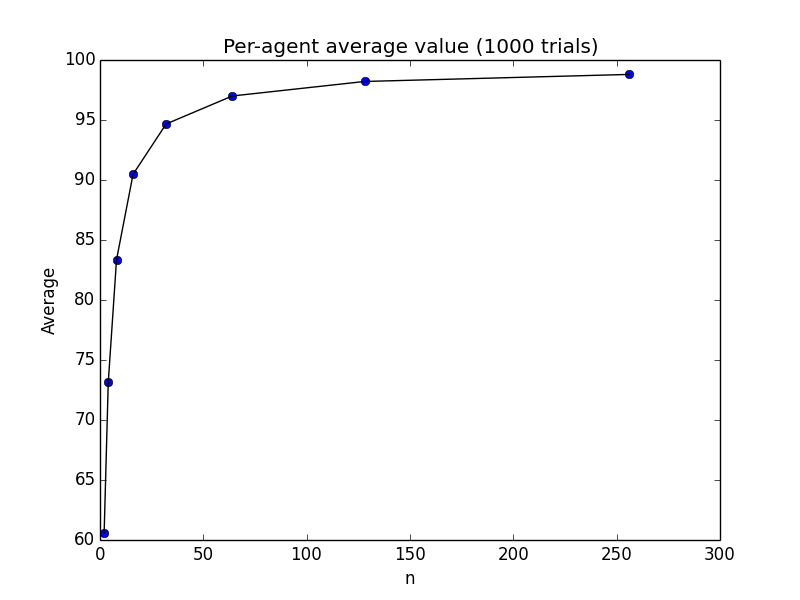
9: 5

The total value of these assignments is 847

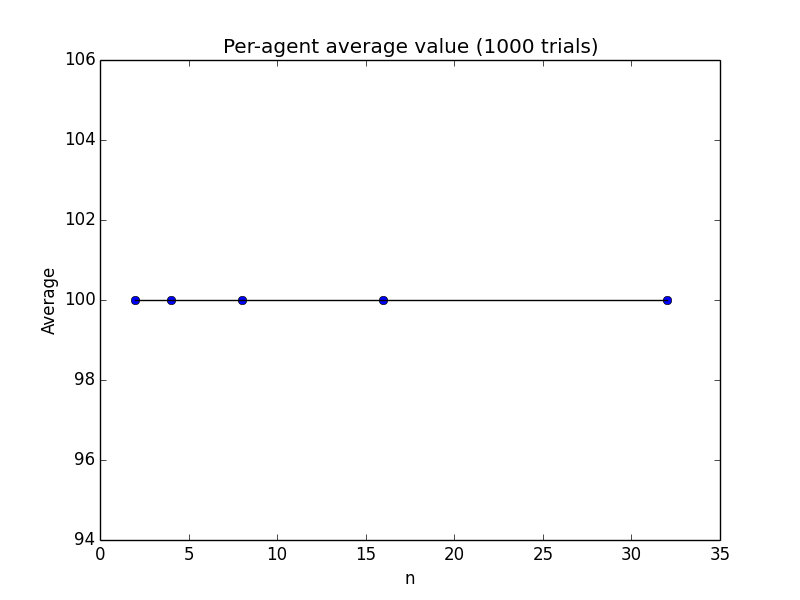
b)

Figure 1 shows the average value for each agent as the number of agents increases from 2 to 256 in powers of 2 and the value of objects is random between 0 and 100. As you can see, the plot asymptotically approaches the value of 100 as n increases. As the number of agents increases, so does the chance that somebody is willing to pay the maximum value of 100 for that item. The blue data point in the graph effectively represents the average maximum value for the 1000 trials for each object an auction of size n. As n increases, the chances that there exists a value closer to the maximum value of M also increases and that causes the plot to asymptotically approach the value of 100.

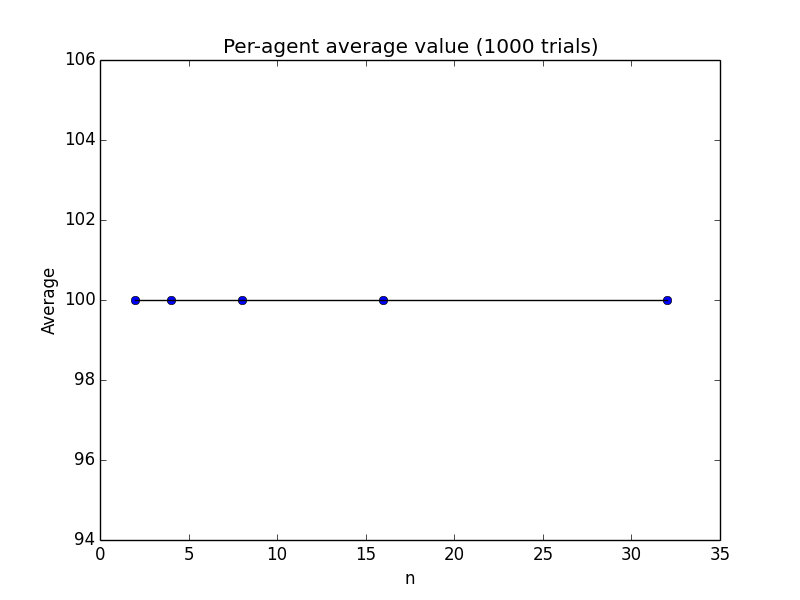
In a smaller run sample experiment, I set the value equal to 100 for all items for all objects for values of n from 2 to 32 and plot the results in Figure 2. In this experiment, the average value for all of the agents is exactly M. This supports my original thesis since the average maximum value for all of the trials is the same value of 100.   
 To further demonstrate this thesis I did an experiment in which I ran the auction algorithm on the identity matrix multiplied by M which can be seen in Figure 3. This gave me a value matrix with diagonals equal to M and 0 everywhere else. The maximum value for the object is 100 and the average maximum value across 100 trials. Note that this plot looks exactly like Figure 2.



**Figure 1:** Average value per-agent for the auction algorithm



**Figure 2:** Average value per-agent for the auction algorithm with a value matrix equal to M at every index.

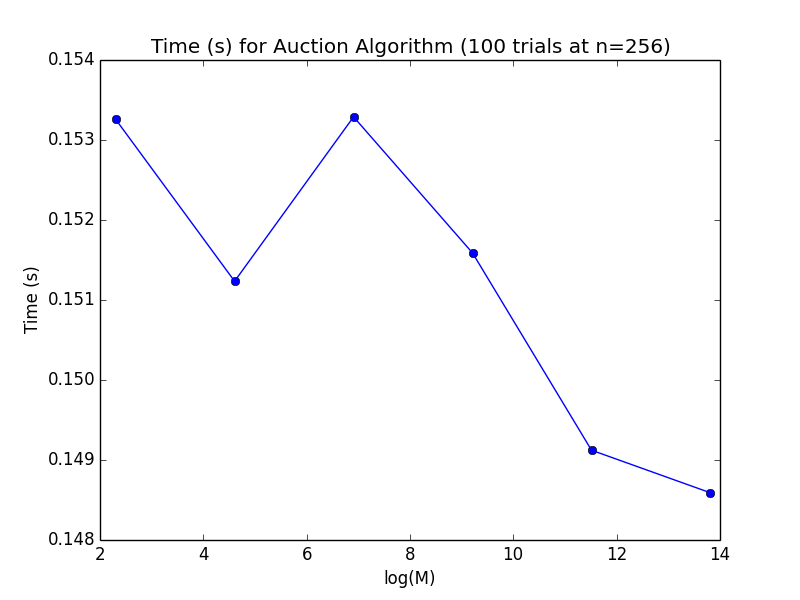


**Figure 3:** Average value per-agent for the auction algorithm with value matrix equal to I\*M where I is the identity matrix (Diagonals are M, everything else is 0)

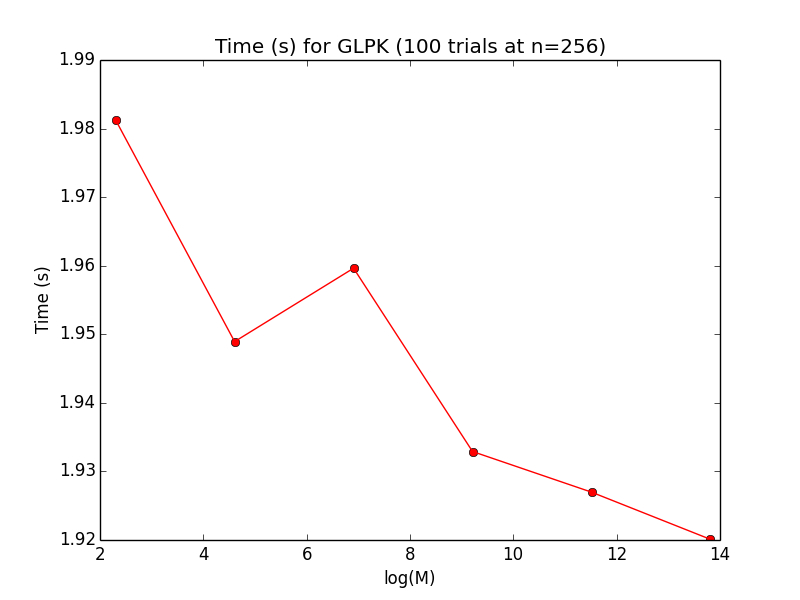
c.

Figure 4 and Figure 5 show the time taken for the auction algorithm and GLPK for log values of M from 10 to 107 when n = 128, respectively. Figure 6 shows the times for the auction and the GLPK algorithms on the same plot with the y-axis starting at 0 for reference. There is a trend in the plot in which the average time taken for both the auction algorithm and GLPK starts at its highest value, immediately drops, reaches a local maximum at its third data point and decreases as M increases.

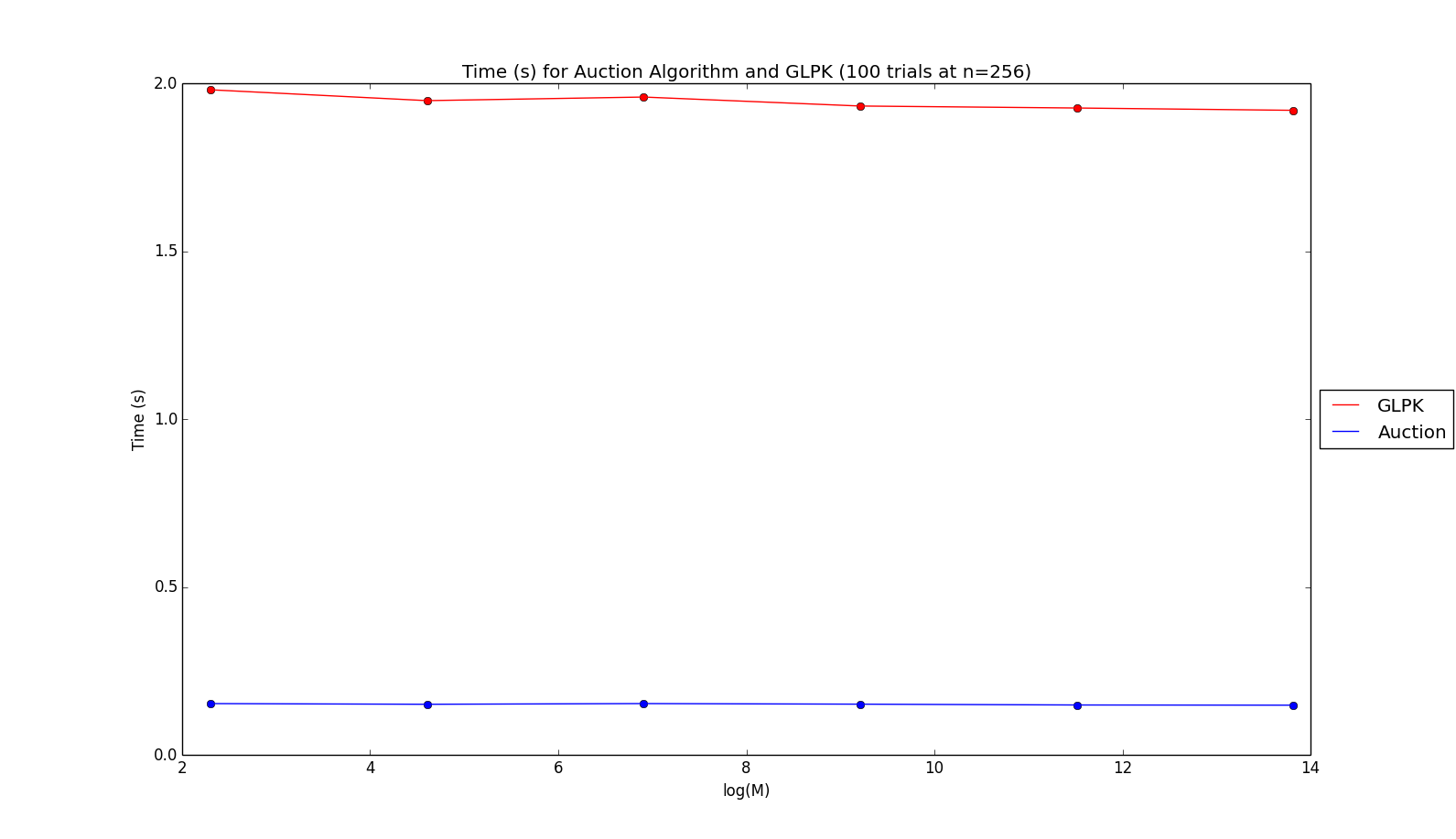
As M increases and n is held constant, there is a larger variance in the values for the matrix of an object. Informally this means that there is less competition for each object. Since the run time of these algorithms is dependent on how many bids (how competitive) there are for a single object, there will be a smaller run time when there is a large variance in the object’s values.



**Figure 4**: Average time taken for 100 trials of the auction algorithm with increasing M



**Figure 5**: Average time taken for 100 trials of GLPK while n=256 and M increases from 10 – 107



**Figure 6**: Average time taken for 100 trials of GLPK and auction algorithm while n=256 and M increases from 10 – 107

2. Possible Matches

Stable Match 1:

Equally Optimal, men and women each get their second choice

1,2

2,3

3,1

Stable Match 2:

Man Optimal, men get 1st choice and women get 3rd choice

1,1

2,2

3,3

Stable Match 3:

Woman Optimal, men get 3rd choice and women get 1st choice

1,3

2,1

3,2

3.

Let M be the matrix of male preferences

Let W be the matrix of female preferences

Let and is the value of the assignment of man i to woman j

Let be the assignment. Where

We want to Maximize

Subject to

4.

Match results with men proposing:

(1,1)

(2,2)

(3,3)

In the scenario presented for this question, woman 1 can lie about her preferences and end up being better off. If woman 1 had the preference: M2 > M3 > M1 the resulting assignment would be:

(1,2)

(2,1)

(3,3)

In this situation, both woman 1 and woman 2 wound up with their top choices while woman 3 had the same results.

5.

Preferences:

A1: h2 > h1 > h3 h1: a1 ~ a1 ~ a3

A2: h1 > h2 > h3 h2: a1 > a1 > a3

A3: h1 > h2 > h3 h3: a3 > a1 > a2

Agent proposing Gale-Shapley algorithm:

Where Red=Rejected, Blue=New Pair, Black=Old Pair

Iteration 1: Iteration 2: Iteration 3: Iteration 4 & Termination

A1 – h2 A1 – h2 A3 – h1 A1 – h1

A2 – h1 A3 – h1 A2 – h2 A2 – h2

A3 – h1 A2 – h2 A1 – h1 A3 – h3

Weakly stable:

**A1 – h1**: A1 prefers h1 but h1 has no strict preference

**A2 – h2**: A2 prefers h1 > h2 but h1 has no strict preference

**A3 – h3**: A3 prefers h1 > h3 but h1 has no strict preference

A3 prefers h2 > h3 but h2 prefers a2 > a3

In the variant of the TTC algorithm, there are two possible resulting assignments:

Assignment 1 Assignment 2

A1 – h2 A1 – h2

A2 – h1 A2 – h3

A3 – h3 A3 – h1

It is impossible for the outcome of the TTC variant to not be agent-optimal because once a house is assigned, it is taken out. This will lead to a situation in which an agent that most prefers the house with no preferences will keep that house forever. In the Gale-Shapley algorithm when a house is assigned it is left on the market awaiting a proposal from a more preferred option. Since there was a house with no preferences in the Gale-Shapley algorithm and ties were broken based on who last proposed, it resulted in a non-agent optimal assignment since it continually switched assignments with whoever proposed to it last.