TEST 3

Yashada Nikam

QUESTION 1
In class we discuss the assignment game (P, Q, α). Accordingly, show the following:
1. Show that if x is an optimal assignment, then it is compatible with any stable payoff (u, v).

Optimal assignment - minimizes the total cost (or maximizes the total valuation) of assigning n resources to n tasks on a one-to-one basis.

A stable payoff (u, v) - is a pair of vectors such that u represents the utilities of the resources and v represents the utilities of the tasks, and they satisfy some conditions.

Now, let x be an optimal assignment and (u, v) be a stable payoff. We want to show that x is compatible with (u, v).

Suppose for the sake of contradiction that x is not compatible with (u, v). Then, there must exist a resource i and a task j such that the cost of assigning resource i to task j under x is greater than the sum of ui and vj.

However, this contradicts the definition of a stable payoff, which requires that the cost of assigning resource i to task j is no greater than the sum of ui and vj for every i and j.

Therefore, x must be compatible with (u, v) if x is an optimal assignment and (u, v) is a stable payoff.

2. If ((u, v), x) is a stable outcome, then x is an optimal assignment.

A stable outcome - is a pair ((u, v), x) such that u represents the utilities of the resources, v represents the utilities of the tasks, x is an assignment of resources to tasks, and they satisfy some conditions.

Now, suppose ((u, v), x) is a stable outcome. We want to show that x is an optimal assignment.

By the definition of a stable outcome, for every resource i and task j, the cost of assigning resource i to task j is no greater than the sum of ui and vj. This implies that the total cost of the assignment x is no greater than the sum of the total utilities of the resources and the total utilities of the tasks.

If x were not an optimal assignment, then there would exist another assignment y that has a lower cost than x. However, this would mean that for some resource i and task j, the cost of assigning resource i to task j under y is less than the sum of ui and vj. This would contradict the definition of a stable outcome, which requires that the cost of assigning resource i to task j is no less than the sum of ui and vj for every i and j.

Therefore, x must be an optimal assignment if ((u, v), x) is a stable outcome.

3. Let ((u, v), x) and ((u′, v′), x′) be stable outcomes of the assignment game (P, Q, α). Show that if x′i j = 1 and u′i > ui implies v′j < v j.

Consider ((u, v), x) and ((u′, v′), x′) are stable outcomes of the assignment game (P, Q, α), and x'ij=1 and u'i > ui. We want to show that v'j < vj.

Suppose for the sake of contradiction that v'j >= vj. Since x'ij=1 and u'i > ui, the cost of assigning resource i to task j under x' is greater than the cost of assigning resource i to task j under x.

Therefore, the difference in cost must be made up by a decrease in vj, since the total cost must remain the same under both stable outcomes ((u, v), x) and ((u′, v′), x′) due to the third condition of stable outcomes.

However, this contradicts the second condition of stable outcomes, which requires that the utility vj is at least as large under x as it is under any other assignment.

Therefore, our initial assumption that v'j >= vj must be false, and we conclude that v'j < vj.

QUESTION 2

In class we discuss the assignment game (P, Q, α). In this question we interpret P as a set of bidders
and Q as a set of objects. Each object has a reservation price of c j. The value of object j to bidder
i is αi j ≥ 0.
A feasible price vector p is a function from Q to R+ such that p(j) = p j satisfies p j ≥ c j. We
will assume that Q contains a null object O whose value is αiO = 0 for all bidders and whose price
pO is always zero. Then if a bidder is unmatched we will say that she or he is assigned the null
object (note that more than one bidder may be assigned to the null object).
1. For a given price vector p, define the demand set Di(p) for bidder i.

For a given price vector p, the demand set Di(p) for bidder i is the set of objects in Q that bidder i is willing to pay for at the given prices.

Di(p) = {j in Q: αij ≥ p(j) for bidder i}

The demand set for bidder i consists of all objects j in Q such that the value of j to bidder i is greater than or equal to the price of j under the given price vector p.

If bidder i is unmatched, then they are assigned the null object O, so Di(p) includes the null object if αiO ≥ pO (i.e., if bidder i is willing to pay the price of the null object). If αiO < pO, then bidder i remains unmatched and is not included in the demand set for any object.

2. A price vector p is called quasi-competitive if there is a matching μ from P to Q such that μ(i) = j then j is in Di(p), and if i is unmatched under μ then O is in Di(p).

In other words, at a quasi-competitive prices p each buyer can be assigned to an object in his or her demand set. In this case, μ is said to be compatible with p.
The pair (p, μ) is a competitive equilibrium if p is quasi-competitive, μ is compatible with p, and p j = c j for all j /∈ μ(P). In this case we denote (p, μ) is a competitive equilibrium and p is called an equilibrium price vector.

Show that if (p, μ) is a competitive equilibrium then the corresponding payoffs are stable.

Under a competitive equilibrium (p, μ), the prices are quasi-competitive, which means that for each buyer i, there exists an object j in their demand set Di(p) that they can be assigned to, and if they are unmatched, then they are assigned to the null object O.

Therefore, the assignment xij under μ is compatible with p, and the price of every object j in μ(P) is equal to its reservation price cj. Also, the price of the null object O is always zero.

Thus, the total cost of the assignment x under (p, μ) is given by:

Σi ui + Σj vj = Σi αi(xij) + Σj pj = Σi αi(xij) + Σj cj ;

where ui and vj are the utilities of bidder i and object j, respectively, and xij is the assignment of bidder i to object j.

Since p j = c j for all j /∈ μ(P), the cost of assigning any object outside of μ(P) is the same as its reservation price. Therefore, the third condition of stable outcomes is satisfied.

Tthe utilities ui and vj in the assignment xij are determined by the values αij and the prices pj, which means that if we change the price of an object or the value of a bidder, the utilities of the bidders and objects will change accordingly. This satisfies the first and second conditions of stable outcomes.

Therefore, the pair ((u, v), x) = ((ui, vj), xij) is a stable outcome, and the corresponding payoffs are stable.

3. Describe an algorithm to compute an equilibrium price vector.

One algorithm to compute an equilibrium price vector in the assignment game (P, Q, α) with reservation prices c j is the following:

1. Set an initial price vector p(0) such that p j (0) > c j for all j in Q, and let t = 0.

2. For each bidder i, compute their demand set Di(p(t)) using the current price vector p(t).

3. Find a matching μ(t) from P to Q such that μ(i) is the object j in Di(p(t)) that maximizes αij/pj(t), and if a bidder is unmatched under μ(t), they are assigned to the null object O. If there is more than one bidder for an object, choose the bidder with the highest value-to-price ratio αij/pj(t).

4. If μ(t) is compatible with p(t) (i.e., every bidder is assigned to an object in their demand set), then p(t) is quasi-competitive. Check if p j (t) = c j for all j not in μ(t)(P). If this condition is satisfied, then (p(t), μ(t)) is a competitive equilibrium. Otherwise, set t = t + 1 and go to step 2 with the updated price vector p(t) such that p j (t) = c j for all j not in μ(t)(P).

5. If μ(t) is not compatible with p(t), update the price vector p(t) as follows: for each unmatched bidder i, set p j (t+1) = αij/μ(i) for the object j in Di(p(t)) that maximizes αij/pj(t), and set p j (t+1) = c j for all objects j not in Di(p(t)) for any bidder. Then, go to step 2 with the updated price vector p(t+1).

The algorithm terminates in a finite number of steps because at each step, the price vector is updated to move closer to a quasi-competitive vector, and there are only finitely many vectors in the space of feasible price vectors.

The output of the algorithm is a competitive equilibrium (p*, μ*) where p* is the final price vector obtained in step 4 and μ* is the corresponding matching obtained in the last iteration of step 3.

4. In your previous answer: does the algorithm generate an equilibrium price vector that maximizes total happiness?

No, the algorithm does not guarantee that the equilibrium price vector maximizes total happiness. The algorithm only guarantees that it finds a price vector that satisfies the conditions for a competitive equilibrium, which ensures that there is no incentive for any buyer or seller to deviate from their assigned match.

The concept of happiness or utility is not explicitly considered in the algorithm, and the equilibrium price vector may not necessarily maximize total happiness. It is possible that a different price vector could result in a higher total happiness, but that price vector may not satisfy the conditions for a competitive equilibrium. Therefore, the algorithm only guarantees that it finds a price vector that satisfies the conditions for a competitive equilibrium, not necessarily one that maximizes total happiness.