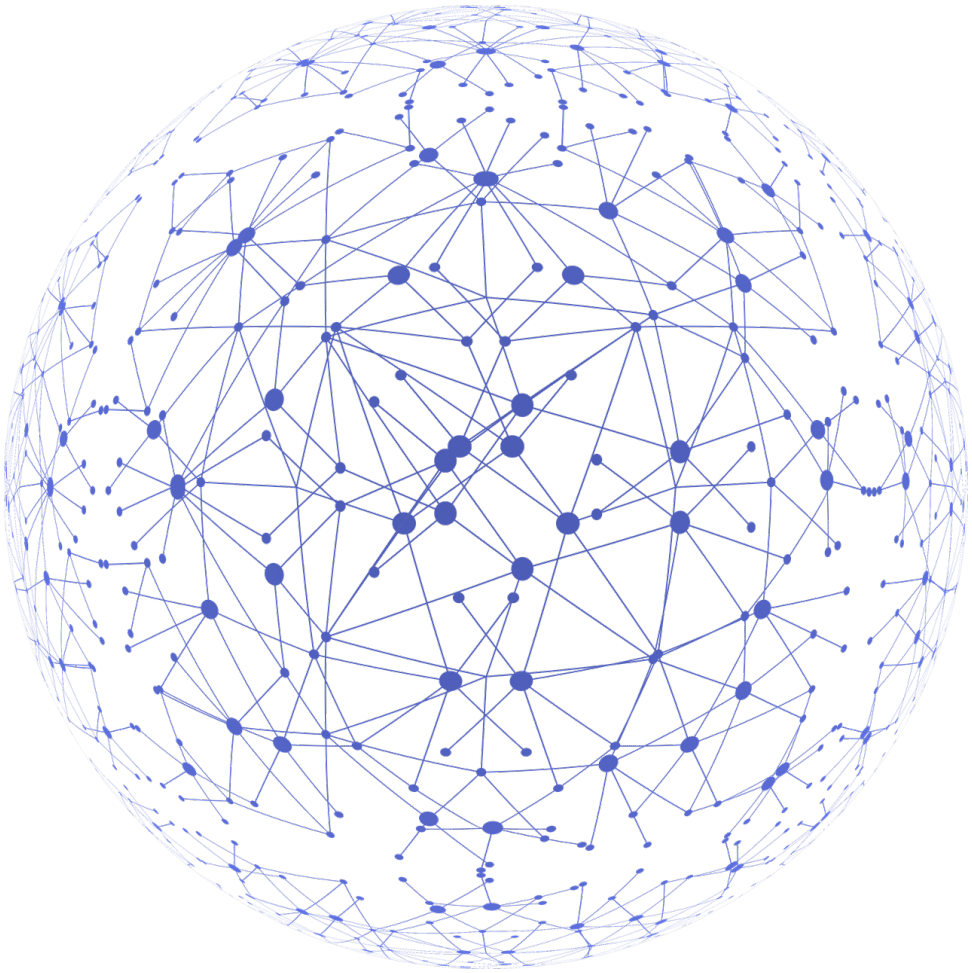
EECS544
Analysis of Societal Networks

Pingbang Hu

August 20, 2023

Abstract

This is a graduate-level course about social network analysis taught by Vijay G Subramanian at University of Michigan, covering topics across graph partitioning algorithms, community detection algorithms, stochastic process and Markov chain, Monte Carlo algorithm, random graph theory and game theory with auctions and matching market. Additionally, we'll use Easley, D. and Kleinberg, J. [EK10] as our main reference.



This course is taken in Fall 2021, and the date on the covering page is the last updated time.

Contents

Chapter 1

Graph and Network

Lecture 1: Introduction to Graph Theory

To analyze a social network, we should first formalize what counts as a network mathematically. In such a way, we're able to develop a mathematically complete and rigorous framework to analyze a given social network and infer some useful properties. 30 Aug. 12:30

1.1 Graph

Let's start with some of the most fundamental definitions.

Definition 1.1.1 (Vertex set)**.** A set is called a *vertex set* if we view its elements as vertices.[a]

[a] Sometimes *nodes*.

Definition 1.1.2 (Graph)**.** There are two kinds of *graph* we're interested in, undirected graph and directed graph.

Definition 1.1.3 (Undirected graph)**.** An *undirected graph* is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ is the edge set.

Definition 1.1.4 (Edge set)**.** An *edge set* $\mathcal{E}$ of an undirected graph $\mathcal{G}$ is a set of paired vertices, which we call edges.[a]

[a] Sometimes *links*.

Definition 1.1.5 (Directed grpah)**.** A *directed graph* is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ is the directed edge set.

Definition 1.1.6 (Directed edge set)**.** A *directed edge set* $\mathcal{E}$ of a directed graph $\mathcal{G}$ is a set of ordered-paired vertices, which we call directed edges.[a]

[a] Sometimes *directed links*.

Example (Undirected and directed graph)**.** Let $A = \{a, b, c, d, e\}$ be a vertex set, then an example of edge sets can be

$$B \coloneqq \{\{a, b\}, \{b, e\}, \{c, d\}\},$$

and

$$B' \coloneqq \{(a, b), (b, e), (c, d)\}$$

is an example of directed edge set. Together, (A, B) is a valid undirected graph, while (A, B') is a directed graph.

Notation. We some time will write an edge between a and b as (a, b) instead of $\{a, b\}$ and a directed edge from a to b as $a \to b$ if the content is clear.

Note. While the definition of a vertex set is purely abstract, we often just deal with concrete graph where the structure of the graph is clear (e.g., in social network, every people are vertices, and relationships between people are edges).

Definition. Given a directed graph $\mathcal{G}$, we can define the followings.

Definition 1.1.7 (In-degree). For a node $v \in \mathcal{V}$, the *in-degree* of v in a directed graph $\mathcal{G}$ is defined as the number of directed edges pointing in to v.

Definition 1.1.8 (Out-degree). For a node $v \in \mathcal{V}$, the *out-degree* of v in a directed graph $\mathcal{G}$ is defined as the number of directed edges pointing out from v.

Moreover, if we generalize above to an undirected graph we have the following.

Definition 1.1.9 (Degree). Given an undirected graph, for a node $v \in \mathcal{V}$, the *degree* of v in an undirected graph is defined as the number of edges link with v.

Definition 1.1.10 (Subgraph). A *subgraph*[a] $\mathcal{G}'$ is defined by a pair $(\mathcal{V}', \mathcal{E}')$ such that $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{E}' \subseteq \mathcal{E}$.

[a] Sometimes *component*.

Note. Notice that the edges in $\mathcal{E}'$ need to be well-defined. Namely, for every $e = (a, b) \in \mathcal{E}'$, a, b need to also be in $\mathcal{V}'$.

Finally, we introduce the following definition.

Definition 1.1.11 (Simple graph). A *simple graph* is a graph without loops and multiple edges.[a]

[a] i.e., no edges like $\{a, a\}$ and there is only one edge between any two nodes.

Lecture 2: Graph Theory

1 Sep. 12:30

Definition 1.1.12 (Path). Given a list of vertices indexed in order as $v_1, \ldots, v_n$ where $v_i \in \mathcal{V}$,

(a) A *path* P connecting v_i in an undirected graph is a subgraph $P = (V, E)$, where $V = \{v_i\}_{i=1}^{n}$, and $E = \{\{v_i, v_{i+1}\}\}_{i=1}^{n-1}$.

(b) A *path* P connecting v_i in a directed graph is a subgraph $P = (V, E)$, where $V = \{v_i\}_{i=1}^{n}$, and $E = \{(v_i, v_{i+1})\}_{i=1}^{n-1}$.

Then, in the same spirit of Definition 1.1.11, we have a so-called simple path.

Definition 1.1.13 (Simple path). A path is a *simple path* if it does not have repeating vertices.

Intuition. Basically, it says that simple paths can't have loops.

With the definitions we introduced, we can now try to characterize a graph.

Definition. We can then define connectivity of a graph as follows.

Definition 1.1.14 (Connected). An undirected graph is *connected* if for every two nodes, there exists at least one path connect them together.

Definition 1.1.15 (Strongly connected). A directed graph is *strongly connected* if for every two nodes, there exists at least one path connect them together.

Notation (Connected component). The above definitions can be generalized to a subgraph as well. In this case, we say that this subgraph is connected, or more often we'll say this subgraph is a *connected component*. Same for strongly connected.

Definition 1.1.16 (Giant component). A subgraph is called a *giant component* of a graph if it is a connected component and with a significant number of nodes of the original graph.

Finally, we introduce the first matrix associated with a graph.

Definition 1.1.17 (Adjacency matrix). For an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the *adjacency matrix of an undirected graph* is a matrix A such that

$$A_{ij} = \begin{cases} 1, & \text{if } \{i,j\} \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases}$$

For a directed graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, the *adjacency matrix of a directed graph* is a matrix A' such that

$$A'_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{E}' \\ 0, & \text{otherwise.} \end{cases}$$

Remark. The adjacency matrix of an undirected graph must be symmetric by definition.

Proof. Since if $\{i,j\} \in \mathcal{E}$, we know that $\{j,i\} \in \mathcal{E}$ as well since $\{j,i\} = \{i,j\}$, so $A_{ij} = A_{ji} = 1$.
On the other hand, if $\{i,j\} \notin \mathcal{E}$, then $\{j,i\} \notin \mathcal{E}$, hence in this case $A_{ij} = A_{ji} = 0$ clearly. ⊛

Lecture 3: Network Property

A network is essentially a graph with some interesting properties. We start with connectivity. 8 Sep. 12:30

1.2 Measuring Connectivity

Intuition. If there are more paths in the graph between different parts of the graph, then this graph is *more connected*.

To characterize paths, we have the following definitions.

Definition 1.2.1 (Edge independent). Two paths are said to be *edge independent* (EI) if they do not share a common edge.

Definition 1.2.2 (Vertex independent). To paths are said to be *vertex independent* (VI) if they do not share a common vertex except the starting vertex and the ending vertex.

1.2.1 Cut Set

Before we formulate the definition of cut set, we'll need to define so-called induced subgraph since they're highly related.

Definition 1.2.3 (Induced subgraph). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the *induced subgraph* with respect to

- $\mathcal{V}' \subseteq \mathcal{V}$ is defined as $\mathcal{G}[\mathcal{V}'] \coloneqq (\mathcal{V}', \hat{\mathcal{E}})$, where $\hat{\mathcal{E}} \coloneqq \{e \in \mathcal{E} : e = (i,j), i,j \in \mathcal{V}'\}$.[a]
- $\mathcal{E}' \subseteq \mathcal{E}$ is defined as $\mathcal{G}[\mathcal{E}'] \coloneqq (\mathcal{V}, \mathcal{E}')$.

[a] This is for directed graph, but the case for undirected graph is similar, just change (i,j) to $\{i,j\}$.

Intuition. This is just a convenient notion for graph subtraction which is essentially just the subtraction between the edge set and the vertex set. Notice that the case of edge-subtraction is trivial, while node-subtraction needs to be well-defined, e.g, it's nonsense to say (i,j) is still in the induced subgraph if one end of the edge is not in the node set anymore.

Definition 1.2.4 (Edge cut set). Given $u, v \in \mathcal{V}$, a set of edges $\mathcal{E}' \subseteq \mathcal{E}$ is called an *edge cut set* if u, v are in different connected components in the subgraph induced from $\mathcal{E}'$.

Definition 1.2.5 (Vertex cut set). Given $u, v \in \mathcal{V}$, a set of vertices $\mathcal{V}' \subseteq \mathcal{V}$ is called a *vertex cut set* if u, v are in different connected components in the subgraph induced from $\mathcal{V}'$.

Note. It's worth noting that in the case of vertex cut set, if u or v is not in $\mathcal{V}'$, then we still say u and v are not in the same connected component (even if neither u nor v is in $\mathcal{V}'$).

Remark. Note that the above definitions are for undirected graph since we only consider connected but not strongly connected. But it's obvious that one can generalize the notion to directed graph.

Theorem 1.2.1 (Mengur's theorem). If for any pair of nodes (i,j) we have $|\mathrm{ECS}(i,j)| > n$ where $\mathrm{ECS}(i,j)$ denotes the collection of edge cut set respect to (i,j), then for any pair of nodes (i,j),

$$\#\text{ edge independent paths between } i \text{ and } j > n.$$

Proof. A nice proof can be found here. ■

Lecture 4: Network Properties

We now try to further characterize the connectedness of a network. We first give a common notion. 13 Sep. 12:30

Definition 1.2.6 (Complete graph). A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is *complete* if $\mathcal{E}$ contains every possible edges.

Notation. We often denote a complete graph with n nodes as K_n.

1.2.2 Bridges and Local Bridges

Definition 1.2.7 (Bridge). A *bridge* is the only edge that connects together two subgraphs that are themselves connected.

Remark. Deleting a bridge may change the distance between two nodes from some finite value to ∞. (disconnects those two nodes)

Definition 1.2.8 (Local bridge)**.** An edge (i, j) is said to be a *local bridge* if i, j have no neighbors in common.

Note. Equivalently, a local bridge is an edge that is not included in any triangles.

Remark. If (i, j) is a local bridge, then we have

$$d_{\mathcal{G}}(i, j) \geq 3$$

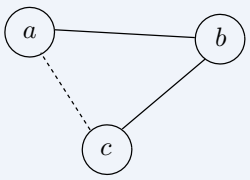
after removing this local bridge. Furthermore, a bridge must also be a local bridge by definition.

Definition 1.2.9 (Span of a local bridge)**.** The *span of a local bridge* b is defined as $d_{\mathcal{G}[\mathcal{E}\setminus\{b\}]}$.

1.3 Triadic Closure

Intuition ([EK10])**.** If two people in a social network have a friend in common, then there is an increased likelihood that they will become friends themselves at some point in the future.

Example (Triadic closure)**.** The *triadic closure* says that if a network has this property, then given $a, b, c \in \mathcal{V}$, if $\{a, b\}, \{b, c\} \in \mathcal{E}$, it's very likely that $\{a, c\} \in \mathcal{E}$ as well.



This motivates us that the number of triangles maybe can be used to quantify the connectedness of a network.

Definition 1.3.1 (Clustering coefficient)**.** For each node i, define the *clustering coefficient*

$$c_i = \frac{\#\text{ of triangles in the graph that include node } i}{d_i(d_i - 1)/2},$$

where d_i is the degree[a] of i.

[a] We assume our network is an undirected for simplicity.

Remark. We have, for any node i, $c_i \in [0, 1]$. If $\forall i\ c_i = 1$, then this graph is complete. If $\forall i\ c_i = 0$, then this graph is a tree.

The above intuition suggests the following characteristics of a network.

Definition 1.3.2 (Strong triadic closure)**.** If a is a node and there are two nodes b, c such that (a, b) and (a, c) are strong ties, then (b, c) will form an edge. If for any nodes a, b, c in this graph $\mathcal{G}$ satisfy this property, then we say this graph has *strong triadic closure property* (STC).

Definition 1.3.3 (Embeddedness)**.** The *embeddedness* of an edge is defined as the number of common neighbors shared by the end points.

Chapter 2

Network Partition

Lecture 5: Partition Networks Hierarchically

15 Sep. 12:30

In this chapter, we want to partition a network in order to study its structure. As we have seen before, the notion of bridges and local bridge is useful in this content. And from this chapter, we'll focus on algorithmic approach to reach our goal, as we'll see soon.

2.1 Shortest Path

Firstly, before we do anything serious, we should see fundamental algorithms on graph. These algorithms will essentially traverse a given graph $\mathcal{G}$ and can give us some interesting property while doing so. We first make the notion of distance between nodes rigorous, which help us proceed.

Definition 2.1.1 (Distance between nodes)**.** We usually let the length of the shortest path between two nodes in a graph $\mathcal{G}$ be the *distance between these two nodes*, denotes as

$$d_{\mathcal{G}}(\cdot,\cdot),$$

which takes two input for two corresponding nodes.

Definition 2.1.2 (Shortest path)**.** Given a graph, a *shortest path* between two nodes u, v is a path whose end nodes are u, u while the sum of the weights of its constituent edges is minimized.

Remark. The distance between nodes is exactly the sum of the weights of the shortest-path edges.

Note. We only discuss unweighted graph so far, where we treat the weight as 1 for all edges.

Now, observe that the distance function $d_{\mathcal{G}}$ need to follow some kind of recursion relation like

$$d_{\mathcal{G}}(i,j) = 1 + \min_{k \in N_{\mathcal{G}}(j)} d_{\mathcal{G}}(i,k)$$

if we're talking about an unweighted graph. Now, we see how we can utilize this idea and find the shortest paths algorithmically.

2.1.1 Breadth-First Search

Firstly, we see a way to traverse a given graph $\mathcal{G}$ called breadth-first search, which owns its name since its natural behavior.

Algorithm 2.1: Breadth-First Search

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a root v^*

```
1  Q ← empty queue                                          // Initialize
2  for v ∈ V do
3      explored[v] ← False                                  // Mark every node as unexplored
4
5  explored[v*] ← True                                      // Explore from root v*
6  Q.push(v)
7
8  while Q ≠ ∅ do
9      v ← Q.pop()
10     for (v, w) ∈ E do                                    // For all edges adjacent to v
11         if explored[w] = False then                      // If w is unexplored
12             explored[w] = True                           // Mark as explored
13             Q.push(w)
14 return
```

This is a simple algorithm which has the nice properties. Firstly, we see that we can interpret the edges $(v, w) \in \mathcal{E}$ in line 10 as either directed or undirected edges, hence the algorithm works on both directed or undirected graphs.

Intuition (Level graph)**.** And if we look closely, we're essentially constructing a level graph, which means that we're exploring the graph from the root v^* layer by layer (level by level) w.r.t. the distance between v^*. In other words, it'll first explore all nodes which has distance 1 from v^*, then 2, and so on and so forth until all nodes in this connected component are explored.

Remark (Depth-first search)**.** Just like breadth-first search, we can change the iterator container of Q by a stack, which will essentially make this algorithm turns into the so-called *depth-first search*. In contrast, BFS, DFS are not building level graph, i.e., it does not explore the graph layer-by-layer from the root v^*, it just randomly pick a direction and go as far as it can. Such property can have certain benefit in some cases, but since we're interested in finding shortest paths, hence we simply omit a detail explanation.

If we use the adjacency matrix to represent $\mathcal{G}$, then since we call for each vertex at most once, with the fact that the adjacency matrix for each vertex is visited at most once when searching for $(v, w) \in \mathcal{E}$ in line 10, which costs $O(|\mathcal{V}|)$, hence in total this algorithm has a runtime as $O(|\mathcal{V}|^2)$.

Remark (Adjacency list)**.** If we instead used something called adjacency list, then for each vertex, the list is visited at most once and if we assume that the set of edges is distributed over set of vertices uniformly, this step costs $O(1 + |\mathcal{E}| / |\mathcal{V}|)$, hence in total the algorithm costs

$$O(|\mathcal{V}| (1 + |\mathcal{E}| / |\mathcal{V}|)) = O(|\mathcal{V}| + |\mathcal{E}|),$$

rather than $O(|\mathcal{V}|^2)$. This is sometimes faster if the graph is dense, but we'll not go into this since this is not that relevant.

Now, since we have the intuition that the breadth-first search is exploring the graph in the fashion of constructing a level graph, then we can just check whether we get to our destination or not by the following modification.

Algorithm 2.2: Breadth-First Search ver.2

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a root v^*, a destination u^*
Result: Distance $d_{\mathcal{G}}(v^*, u^*)$

```
Q ← empty queue                                              // Initialize
for v ∈ V do
    explored[v] ← False                                      // Mark every node as unexplored
c ← 0                                                        // Set a counter

explored[v*] ← True                                          // Explore from root v*
Q.push(v)

while Q ≠ ∅ do
    v ← Q.pop()
    c ← c + 1
    if v = u* then                                           // If we reach the destination
        return c
    for (v, w) ∈ E do                                        // For all edges adjacent to v
        if explored[w] = False then                          // If w is unexplored
            explored[w] = True                               // Mark as explored
            Q.push(w)
return
```

Intuition (Counter). We're essentially adding a *counter* which counts how many layers we got, which corresponding to the distance between v^* and all nodes in that layer.

Remark (Find the path). If we somehow want the whole shortest path from v^* to u^* rather than just the distance between them, we can then maintain a list which records the successor of every node we're going to explore, and we just trace back.

2.1.2 Dijkstra's Algorithm

There are many approaches to calculate the distance between nodes, one way is *Djikstra's algorithm.*

Complete this section.

2.2 Girvan-Newman Algorithm

Firstly, a naive and simple way to partition a network is as follows.

- Remove bridges and local bridge.
- Assign importance for each node.

Problem 2.2.1. In what order?

Answer. Assign importance to local bridge. ⊛

This idea is the backbone of the Girvan-Newman algorithm. We first see a simple algorithmic to partition a network by utilizing the intuition.

1. Rank all local bridges.
2. Remove the highest ranked one, which yields a partition.
3. Re-calculate ranking.
4. Repeat from 2. until no edges remain.

Remark. Note that at the end all nodes become isolated. Also, layer on, we'll (always) use pseudocodes to describe such algorithmic procedure.

Now, the following question arises.

Problem 2.2.2. How to rank order local bridges?

Intuition. An edge is more important if more shortest paths use it. Notice that a *normalization* is needed: Between pairs of nodes, there should be lots of variation in the number of the shortest paths, and we want to balance these out.

This suggests the following notion.

Definition 2.2.1 (Betweenness). Imagine that there is a unit flow of water between each (distinct) pair of nodes. If the flow has k different ways to go to,[a] we divide this unit flow into $1/k$. We then equally divide this flow amongst the shortest paths coming out of the starting node.

[a] Actually not only depends on where can the flow go, but also where were that node added into BFS in the first place

With the definition of betweenness, we give the pseudocode of Girvan-Newman algorithm.

Algorithm 2.3: Girvan-Newman Algorithm

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

for $e \in \mathcal{E}$ **do**
  $\mathsf{between}_e \leftarrow 0$ `// Initialize betweenness variable for every edge`

while $\mathcal{E} \neq \varnothing$ **do**
  `Betweenness(`$\mathcal{G}$, $\{\mathsf{between}_e\}_{e\in\mathcal{E}}$`)` `// Calculate betweenness of all remaining edges`
  $\mathsf{between}_{\text{highest}} \leftarrow \max_{e\in\mathcal{E}} \mathsf{between}_e$
  $\mathcal{E}_{\text{highest}} \leftarrow \{e \in \mathcal{E} : \mathsf{between}_e = \mathsf{between}_{\text{highest}}\}$
  $\mathcal{E} \leftarrow \mathcal{E} \setminus \mathcal{E}_{\text{highest}}$
  $\mathcal{G} \leftarrow \mathcal{G}[\mathcal{E}]$
return

Remark. Run time analysis: $O(\ln n)$.

Lecture 6: Girvan-Newman Algorithm Analysis

20 Sep. 12:30

Problem 2.2.3. But how do we calculate betweenness in reality?

Answer. We see that the only obstacle we're facing is the fact that we don't know how to calculate the number of shortest paths. And to calculate how many shortest paths are there between nodes, this can be done by using breadth-first search in a slightly different manner. ⊛

The key is that any sub-path to destination is also a shortest path. To calculate this with an algorithmic procedure, we use BFS. In the BFS exploration from v^* to u^*, we have

$$\#\text{shortest path from } v^* \text{ to } u^* = \#\text{shortest path from } v^* \text{ to } j + \#\text{shortest path from } v^* \text{ to } k + \dots$$

suppose that there are edges between $j, k, \dots$ to u^*.

Remark. In this case, j, k are at same level $n-1$, and u^* is at level n if the distance between v^* and u^* is n.

Finally, to calculate this, we can add *another* counter when running BFS. This is certainly a different type of counter from c, which we call it d. This time, the counter d is induced by the above equation,

and that implies the updating rule being

$$d_v^n = \sum_{(u,v)\in\mathcal{E}} d_u^{n-1}$$

where d_v^n maintains the number of shortest path to v where v is at layer n, i.e., $d_{\mathcal{G}}(v^*, v) = n$.[1]

The unit flow flows from nodes back from the path it is added into the BFS algorithm needs to keep its own 1, and then distribute.

Remark. By adding this counter d, in the end, d_v (regardless of the supscript, since only one such $d_v^{n'}$ for any n' will be defined) is the number of shortest path from v^* to v. Then we can just calculate the betweenness from its definition.

2.3 Homophily

Links in a (social) network are formed between people that are *similar*. This is an important feature that leads to the formation of communities.

Problem 2.3.1. How can we define the notion of *like* and *opposite*?

Answer. Consider a network with two types of individuals. ⊛

Example (Middle school). Consider a school of boys and girls. Suppose the fraction of boy is p and the fraction of girls is $q = 1 - p$. We have

$$\frac{\binom{np}{2}}{\binom{n}{2}} \cong p^2$$

and also

$$1 - p^2 - q^2 = 2pq.$$

If the fraction of cross connection $\ll 2pq$, homophily is considered likely. Otherwise, if the fraction $\gg 2pq$, inverse-homophily is likely to occur.

We see that some social implications arise from this

This part is unclear, need to polish.

1. schelling model
2. parameters
   (a) satisfaction threshold
   (b) population of each partition
   (c) vacancy
3. a grid of X and O (denoting partition of agents).

- Satisfied Agent: A satisfied agent is one that is surrounded by at least t percent of agents that are like itself.
- Dynamics: When an agent is not satisfied, it can be moved to any vacant location in the grid.

[1] Note that only those d_v^n such that $n = d_{\mathcal{G}}(v^*, v) = n$ is defined. It's nonsense to define $d_v^{n'}$ where $d_{\mathcal{G}}(v^*, v) \neq n'$ from the meaning we're trying to put on d and maintained.

Chapter 3

Diagonalization

Lecture 7: Spectral Theorem

Our goal is to do network partition, and to do so, we now develop a power tool to decompose a matrix. Later, we'll see that by decomposing a special matrix called Laplacian which characterize a network, we'll get a corresponding decomposition on the network in an insightful way. 22 Sep. 12:30

3.1 Eigenvalue and Eigenvector

We now want to study a very useful theorem called spectral theorem. Before this, we need to study eigenvalue.

Definition. For a matrix A, if v is a nonzero vector such that we can write

$$Av = \lambda v,$$

then we call v the eigenvector while λ the eigenvalue.

Definition 3.1.1 (Eigenvalue)**.** The *eigenvalue* is the scalar multiple λ as described above.

Definition 3.1.2 (Eigenvector)**.** The *eigenvector* is the nonzero vector v as described above.

Remark. Indeed, eigenvector and eigenvalue is defined in a more general sense, i.e., for linear transformation T such that

$$T(u) = \lambda u.$$

This is a generalization since a linear transformation can come from an infinite-dimensional vector space, which don't have a vector representation in this case.

Note (Left eigenvector)**.** The notion of eigenvector is indeed also called a *right eigenvector*. As for the *left eigenvector*, it's the similar definition but with transpose. Rigorously, for a matrix A, the *left eigenvector* is the nonzero vector v which satisfies

$$v^\top A = \lambda v^\top.$$

But for simplicity, we do not make this into a definition.

Note (Characteristic equation)**.** To find eigenvalues of a matrix A, this is equivalent to solve $Av = \lambda v$

where v is undetermined. We can indeed use the so-called *characteristic equation*

$$\det(\lambda I - A) = 0$$

to help us solve for λ.

3.2 Spectral Theorem

With some assumption on structural properties of a matrix, we can say something about its eigenvalue. Hence, we start with the so-called symmetric matrices.

Definition 3.2.1 (Symmetric matrix). A *symmetric matrix* is a square matrix such that

$$A = A^\top.$$

With this assumption, we have the following strong theorem which we'll heavily rely on.

Theorem 3.2.1 (Spectral Theorem). If A is a symmetric matrix, then all eigenvalues are real. Furthermore, the left and right eigenvectors are the same, and they construct an orthonormal basis.

Proof. We verify the second part of the theorem. If we have A is symmetric, then suppose $\vec{x}$ is a right eigenvector, we have

$$A\vec{x} = \lambda\vec{x} \Rightarrow (A\vec{x})^\top = \lambda\vec{x}^\top \Rightarrow \vec{x}^\top A^\top = \lambda\vec{x}^\top \Rightarrow \vec{x}^\top A = \lambda\vec{x}^\top,$$

where we see that $\vec{x}$ is just left eigenvector, hence the left and right eigenvectors are the same. ■

Lecture 8: Diagonalization

3.3 Eigen Decomposition

27 Sep. 12:30

From spectral theorem, we can **diagonalize** a symmetric matrix A via so-called eigen decomposition as follows.

Theorem 3.3.1 (Eigen decomposition). For a symmetric matrix A, we can always write

$$A = UDU^\top,$$

where U is the eigenvalues matrix, and D is the diagonal matrix of eigenvalues. W

Proof. Firstly, from Theorem 3.2.1, denote n right eigenvectors of A by $u_1, \ldots, u_n$ where u_i corresponds to eigenvalue λ_i such that

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

Corresponding to this order, we define U as

$$U = \begin{pmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & & | \end{pmatrix},$$

then we have

(a) u_i are linear independent, and they form a basis in $\mathbb{R}^n$.

(b) $u_i^\top u_j (= \langle u_i, u_j \rangle) = 0$ for $i \neq j$.

(c) $u_i^\top u_i = \sum_{j=1}^n u_{ij}^2 = 1$ ($length = 1$).

Remark (U is invertible). We see that U is invertible. Indeed, we have $U^{-1} = U^\top$ since

$$U^\top U = \begin{pmatrix} — & u_1^\top & — \\ — & u_2^\top & — \\ & \vdots & \\ — & v_n^\top & — \end{pmatrix} \begin{pmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & & | \end{pmatrix} = I.$$

Then, we claim that we can write A as $A = UDU^\top$ where

$$D = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}.$$

To see this, from $U^\top = U^{-1}$, we have $AU = (UDU^\top)U = UD$ since

$$\begin{aligned} AU = A \begin{pmatrix} | & & | \\ u_1 & \cdots & u_n \\ | & & | \end{pmatrix} &= \begin{pmatrix} | & & | \\ Au_1 & \cdots & Au_n \\ | & & | \end{pmatrix} \\ &= \begin{pmatrix} | & & | \\ \lambda_1 u_1 & \cdots & \lambda_n u_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_n \\ | & & | \end{pmatrix} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} = UD \end{aligned}$$

Hence we're done. ■

Example. Find the eigenvalue of

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}.$$

Proof. The characteristic equation of A is

$$\begin{aligned} \det(\lambda I - A) = \det\left(\begin{pmatrix} \lambda - 1 & 0 & 0 \\ 0 & \lambda - 2 & -1 \\ 0 & -1 & \lambda - 2 \end{pmatrix} \right) \\ = (\lambda - 1)((\lambda - 2)^2 - 1) \\ = (\lambda - 1)(\lambda - 2 - 1)(\lambda - 2 + 1) \\ = (\lambda - 1)(\lambda - 3)(\lambda - 1) = 0. \end{aligned}$$

We see that the roots of the characteristic equation are $1, 1, 3$ with 1 has multiplicity 2. ⊛

Example. Give an example of a where A defined as

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & a & 2 \end{pmatrix}$$

has complex eigenvalues.

Proof. The characteristic equation of A is

$$\begin{aligned}\det(\lambda I - A) &= \det\left(\begin{pmatrix}\lambda-1 & 0 & 0\\ 0 & \lambda-2 & -1\\ 0 & -a & \lambda-2\end{pmatrix}\right)\\ &= (\lambda-1)((\lambda-2)^2 - a)\\ &= (\lambda-1)(\lambda^2-4\lambda+4-a) = 0\end{aligned}$$

Clearly, there is a root 1 for $\lambda-1=0$, as for the quadratic $\lambda^2-4\lambda+(4-a)=0$, $\triangle = 16-4(4-a) = 4a$. We see that for $a=-1$, $\triangle = -4 < 0 \Rightarrow$ the quadratic has complex roots. Hence,

$$A = \begin{pmatrix}1 & 0 & 0\\ 0 & 2 & 2\\ 0 & -1 & 2\end{pmatrix}$$

has a complex eigenvalue. ⊛

3.4 Network Decomposition

Now comes to our main goal, i.e., to do the network decomposition. First, let's define a matrix which capture the graph structure.

Definition 3.4.1 (Laplacian). For an undirected graph, the *Laplacian* is defined as $L \coloneqq D - A$, where A is the adjacent matrix, D is diagonal degree matrix.

Remark. Since

$$L^\top = (D-A)^\top = D^\top - A^\top = D - A = L,$$

we see that L is symmetric, hence its eigenvalues are real by Theorem 3.2.1. Many incidence matrices $B_{|\mathcal{V}|\times|\mathcal{E}|}$ which encode the relation between edges and nodes satisfies

$$L = BB^\top.$$

We'll show that L has non-negative eigenvalues later.

3.4.1 Positive Definite and Positive Semi-Definite

Definition 3.4.2 (Positive definite). A is *positive definite* (PD) if and only if

$$\forall \vec{x} \neq \vec{0}, \quad \vec{x}^\top A \vec{x} > 0.$$

Definition 3.4.3 (Positive semi-definite). A is *positive semi-definite* (PSD) if and only if

$$\forall \vec{x} \neq \vec{0}, \quad \vec{x}^\top A \vec{x} \geq 0.$$

Lemma 3.4.1. If λ is a real eigenvalue, then $\lambda \geq 0 \Leftrightarrow A$ is positive semi-definite; also, $\lambda > 0 \Leftrightarrow A$ is positive-definite.

Proof. Suppose λ is a real eigenvalue, let $\vec{u}$ be the eigenvector such that $A\vec{u} = \lambda\vec{u}$, and

$$\vec{u}^\top A \vec{u} = \vec{u}^\top(\lambda\vec{u}) = \lambda \underbrace{\vec{u}^\top \vec{u}}_{>0} \geq 0,$$

which implies A is positive semi-definite. Similarly, if $\lambda > 0 \Rightarrow \vec{u}^\top\vec{u} > 0$, A is positive definite. Lastly, the backward direction is obvious. ■

Lemma 3.4.2. A is positive semi-definite and symmetric $\Leftrightarrow \exists \widetilde{B}$ such that $A = \widetilde{B}\widetilde{B}^\top$.

Proof. We see that if $A = \widetilde{B}\widetilde{B}^\top$, it's clearly symmetric and

$$x^\top Ax = x^\top \widetilde{B}\widetilde{B}^\top x = (\widetilde{B}x)^\top \widetilde{B}x = y^\top y \geq 0,$$

hence A is positive semi-definite.

For another direction, from Theorem 3.2.1, since A is symmetric, we know that $A = UDU^\top$ where all entries of D is ≥ 0 by Lemma 3.4.1. We see that we can decompose D into $D = CC^\top$ where

$$D = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} = \begin{pmatrix} \sqrt{\lambda_1} & & & \\ & \sqrt{\lambda_2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda_n} \end{pmatrix} \begin{pmatrix} \sqrt{\lambda_1} & & & \\ & \sqrt{\lambda_2} & & \\ & & \ddots & \\ & & & \sqrt{\lambda_n} \end{pmatrix} =: CC^\top,$$

then by defining $\widetilde{B} \coloneqq UC$, we see that

$$A = UDU^\top = UCC^\top U^\top = (UC)(UC)^\top = \widetilde{B}\widetilde{B}^\top$$

as desired. ■

Remark (Properties of Laplacian). Given a Laplacian $L = BB^\top$, from Lemma 3.4.2, L is positive semi-definite, which further implies

$$0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

But $\lambda_1 = 0$ since we have $L = D - A$ where $\mathrm{diag}(d_1, \ldots, d_n)$ being the diagonal degree matrix, and furthermore,

$$L\vec{1} = (D - A)\vec{1} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} - A\vec{1} = \vec{0}$$

since $d_i = \sum_{j=1}^n A_{ij}$ from the definition of A, so

$$L\vec{1} = 0\vec{1}.$$

Hence, 0 is an eigenvalue with eigenvector $\vec{1}$, so $\lambda_1 = 0$.

Now we study the multiplicity of an eigenvalue of L. We first introduce the following definition.

Definition 3.4.4 (Multiplicity). We say the eigenvalue λ^* of A has *multiplicity* k if

$$\det(\lambda I - A) = (\lambda - \lambda^*)^k g(\lambda)$$

where $g(\cdot)$ is a polynomial in λ with $g(\lambda^*) \neq 0$.

If the network has two connected components, we can then relabel the nodes and obtain a concatenation of two adjacency matrices that doesn't intersect

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$$

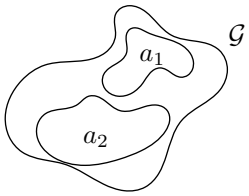


Figure 3.1: Two connected components graph.

Then we see that we can also represent the Laplacian in block matrix form as

$$L = \begin{bmatrix} D_1 - A_1 & 0 \\ 0 & D_2 - A_2 \end{bmatrix} =: \begin{bmatrix} L_1 & 0 \\ 0 & L_2 \end{bmatrix}.$$

We then see that the characteristic equation of L is[1]

$$\det(\lambda I - L) = \det(\lambda I_1 - L_2)\det(\lambda I_2 - L_2)$$

Then, as we just remarked, both L_1 and L_2 has an eigenvalue being 0, which implies that 0 is an eigenvalue of L with multiplicity 2.

We see that multiplicity of eigenvalue of Laplacian can help us find communities. This suggests the following definition.

Definition 3.4.5 (Fiedler eigenvalue)**.** The *Fiedler eigenvalue* of a graph $\mathcal{G}$ is the second-smallest eigenvalue (neglect multiplicity) of the Laplacian of $\mathcal{G}$.

We see that by first doing the eigen-decomposition on Laplacian and find all eigenvalues that are close to zero, which can be used to find communities, and nearly all community detection algorithms rely on this fact.

As previously seen. Turning back to Givran-Newman algorithm, though it works at social network, but what about other tasks like internet?

Problem 3.4.1. Hyperlinks (directed): How important is the edge?

Answer. Internet search seems to rank nodes, not edges. (the website itself is the point?) ⊛

An immediate question arises.

Problem 3.4.2. How does one rank nodes?

Answer. Network centrality. ⊛

Example (Network centrality)**.** There are many examples of methods to quantify network centrality, e.g.,

- Degree. Ranks nodes by edges.
- Eigen-centrality. Adjacency matrix, finds an eigenvector and use its entries.

We now study this in depth.

[1] Detailed derivation is omitted. It's essentially followed from the block structure.

Chapter 4

Page Rank

In this chapter, we still try to analyze a given network, but now we're focusing on nodes-importance rather than links-importance. Our goal is to develop the so-called **page rank**, which is inspired by the following algorithm.

4.1 HITS Algorithm

HITS is a hyperlinks-induced topic search algorithm proposed by John Kleinberg, whose backbones are the idea of **hubs** and **authorities** and the **voting mechanism**.

- Hubs: nodes that *aggregate*.
- Authorities: nodes that are *important*.

The key idea is that

1. If many websites (nodes) point to it, then this website is important (authority property).
2. If a node points to many authority nodes, then it is important as well.

Then by iteratively identifying hubs and authorities, we can rank the nodes in the network. We now describe in detailed how this algorithm works. We first see the pseudocode of the algorithm.

Algorithm 4.1: HITS Algorithm

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Result: hub, auth

for $v \in \mathcal{V}$ **do** // Initialize
  $\mathsf{auth}(v) \leftarrow 1$
  $\mathsf{hub}(v) \leftarrow 1$

while *not converge* **do**
  for $v \in \mathcal{V}$ **do** // auth update
    $\mathsf{auth}(v) \leftarrow \sum_{u \in \mathcal{V}:\, u \to v \in \mathcal{E}} \mathsf{hub}(u)$
  for $v \in \mathcal{V}$ **do** // hub update
    $\mathsf{hub}(v) \leftarrow \sum_{u \in \mathcal{V}:\, v \to u \in \mathcal{E}} \mathsf{auth}(u)$
  for $v \in \mathcal{V}$ **do** // auth normalize
    $\mathsf{auth}(v) \leftarrow \mathsf{auth}(v) / \sum_{u \in \mathcal{V}} \mathsf{auth}(u)$
  for $v \in \mathcal{V}$ **do** // hub normalize
    $\mathsf{hub}(v) \leftarrow \mathsf{hub}(v) / \sum_{u \in \mathcal{V}} \mathsf{hub}(u)$
return hub, auth

Note. There are two facts are worth noting.

1. $\texttt{hub}(v)$ and $\texttt{auth}(v)$ will converge as long as initial score are positive.
2. Final scores will be independent of the initial scores.

Lecture 9: Analysis of HITS

4.2 Analysis of HITS

29 Sep. 12:30

Though the HITS algorithm is clear and simple enough, but we can indeed get a closed form solution to both the authority and the hub scores of the algorithm.

- Authority update:

$$\mathtt{a}_k(i) = \sum_{j\in\mathcal{V}:\ j\to i\in\mathcal{E}} \mathtt{h}_{k-1}(j) = \sum_{j\in\mathcal{V}:\ A_{ji}=1} \mathtt{h}_{k-1}(j) = \sum_{j\in\mathcal{V}} A_{ji}\mathtt{h}_{k-1}(j) = \left(A^\top \mathtt{h}_{k-1}\right)_i.$$

  We see that

$$\mathtt{a}_k = A^\top \mathtt{h}_{k-1}.$$

- Hub update:

$$\mathtt{h}_k(j) = \sum_{j\in\mathcal{V}:\ i\to j\in\mathcal{E}} \mathtt{a}_k(j) = \sum_{j\in\mathcal{V}} \mathtt{a}_k(j) = \sum_{j\in\mathcal{V}} A_{ij}\mathtt{a}_k(j) = \left(A\mathtt{a}_k\right)_i.$$

  We see that

$$\mathtt{h}_k = A\mathtt{a}_k.$$

  Furthermore,

$$\mathtt{h}_k = A\mathtt{a}_k \Rightarrow \mathtt{h}_k = A\mathtt{a}_k = A(A^\top \mathtt{h}_{k-1}) = AA^\top \mathtt{h}_{k-1}.$$

If we ignore the normalization for now, then

$$\mathtt{h}_k = AA^\top \mathtt{h}_{k-1} = (AA^\top)AA^\top h_{k-2} = \cdots = (AA^\top)^k \mathtt{h}_0.$$

Hence, we need to study $AA^\top$. Firstly, from Lemma 3.4.2, we see that $AA^\top$ is positive semi-definite and symmetric. Furthermore, from Theorem 3.2.1, since $AA^\top$ is symmetric, all eigenvalues are real. Then from Lemma 3.4.1, we see that all eigenvalue of $AA^\top$ are positive, i.e.,

$$0 \le \lambda_1 \le \lambda_2 \le \cdots \le \lambda_{|\mathcal{V}|}.$$

From the Theorem 3.2.1, let the eigenvectors of $AA^\top$ be $u_1, \ldots, u_{|\mathcal{V}|}$, then

$$\mathtt{h}_0 = \sum_{i=1}^{|\mathcal{V}|} \alpha_i u_i$$

where α_i are unique since $\{u_i\}$ form a basis, and

$$\alpha_i = u_i^\top \mathtt{h}_0 = \sum_{j=1}^{|\mathcal{V}|} \alpha_j u_i^\top u_j.$$

Moreover, we have

$$\mathtt{h}_1 = (AA^\top)\mathtt{h}_0 = \sum_{i=1}^{|\mathcal{V}|} \alpha_i \lambda_i u_i;$$

$$\vdots$$

$$\mathtt{h}_k = (AA^\top)^k \mathtt{h}_0 = \sum_{i=1}^{|\mathcal{V}|} \alpha_i \lambda_i^k u_i.$$

With the above formula, we now consider

$$0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \underbrace{\lambda_{|\mathcal{V}|}}_{\text{largest}}.$$

Case i. Unique largest eigenvalue: $\lambda_{|\mathcal{V}|} > \lambda_{|\mathcal{V}|-1} \geq \cdots \geq \lambda_1 \geq 0$, then the term λ^k will dominate, namely

$$\mathtt{h_k} = \sum_{i=1}^{|\mathcal{V}|} \alpha_i \lambda_i^k u_i.$$

Dividing both sides by $\lambda_{|\mathcal{V}|}^k$, we have

$$\frac{\mathtt{h_k}}{\lambda_{|\mathcal{V}|}^k} = \sum_{i=1}^{|\mathcal{V}|-1} \alpha_i \left(\frac{\lambda_i}{\lambda_{|\mathcal{V}|}}\right)^k u_i + \alpha_{|\mathcal{V}|} u_{|\mathcal{V}|}.$$

Then since $\lambda_i/\lambda_{|\mathcal{V}|} < 1$ for $1 \leq i \leq |\mathcal{V}| - 1$, hence when $k \to \infty$, all terms in the sum will converge to 0, leaving us with

$$\lim_{k\to\infty} \frac{\mathtt{h_k}}{\lambda_{|\mathcal{V}|}^k} = \alpha_{|\mathcal{V}|} u_{|\mathcal{V}|}.$$

Since $u_{|\mathcal{V}|}$ is a non-zero vector, we consider the following two cases:

(a) All terms are non-negative with one positive. Firstly, suppose $\mathtt{h_0} = x > 0$ such that $u_{|\mathcal{V}|}^\top x = \alpha_{|\mathcal{V}|} > 0$. Then we have

$$\lim_{h\to\infty} \frac{\left(AA^\top\right)^k x}{\lambda_{|\mathcal{V}|}^k} = \alpha_{|\mathcal{V}|} u_{|\mathcal{V}|}.$$

We see that the left-hand side is greater or equal to 0, and stays in positive for every $k \Rightarrow$ the limit is also positive. Now, since $\alpha_{|\mathcal{V}|} > 0$, we can conclude that $u_{|\mathcal{V}|} \geq 0$. In other words, $\mathtt{h_0}$ can be any vector with all coefficients positive and then $\alpha_{|\mathcal{V}|} = u_{|\mathcal{V}|}^\top \mathtt{h_0} > 0$.

(b) At least one term is positive. $\mathtt{h_0} = x > 0$ such that $u_{|\mathcal{V}|}^\top x > 0$.

> **Remark.** To rank nodes, all we need is the vector $u_{|\mathcal{V}|}$. And the limiting value of $u_{|\mathcal{V}|}$ is called the **hub score**.

Case ii. Non-unique largest eigenvalues: For discussion, we assume that the largest eigenvalue has algebraic multiplicity of 2. Namely,

$$\lambda_{|\mathcal{V}|} = \lambda_{|\mathcal{V}|-1} \geq \cdots \geq \lambda_1 \geq 0.$$

Again, from the same discussion, we see that

$$\mathtt{h_0} = \sum_{i=1}^{|\mathcal{V}|} \alpha_i u_i.$$

Furthermore, we see that

$$\frac{\left(AA^\top\right)^k h_0}{\lambda_{|\mathcal{V}|}^k} = \sum_{i=1}^{|\mathcal{V}|-2} \alpha_i \left(\frac{\lambda_i}{\lambda_{|\mathcal{V}|}}\right)^k u_i + \alpha_{|\mathcal{V}|-1} u_{|\mathcal{V}|-1} + \alpha_{|\mathcal{V}|} u_{|\mathcal{V}|}.$$

Since $\lambda_i/\lambda_{|\mathcal{V}|} < 1$ for $1 \leq i \leq V - 2$, hence when $k \to \infty$, all terms in the sum will converge to 0, leaving us with

$$\lim_{k\to\infty} \frac{(AA^\top)^k \mathtt{h_0}}{\lambda_V^k} = \alpha_{|\mathcal{V}|-1} u_{|\mathcal{V}|-1} + \alpha_{|\mathcal{V}|} u_{|\mathcal{V}|}.$$

Remark. It still converges but it may not be $u_{|\mathcal{V}|}$.

Lecture 10: Page Rank

4.3 Page Rank Algorithm

4 Oct. 12:30

As previously seen (Voting procedure)**.** The hubs vote authorities, while authorities vote for hubs in HITS algorithm.

But this is not always the case. Rankings that can be collaborated as the following new algorithm will show, which will determine the importance of a node we called *page rank*.

Algorithm 4.2: Basic Page Rank Algorithm

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, iteration L
Result: pageRank

```
for v ∈ V do                                                    // Initialize
    pageRank(v) ← 1/|V|

for i = 1, ..., L do
    for v ∈ V do                                                // Send
        neighborNum ← |{v → u ∈ E}|
        if neighborNum ≠ 0 then          // divides pageRank(v) to its (out)neighbor
            for v → u ∈ E do
                pageRankNew(u) ← pageRankNew(u) + pageRank(v)/neighborNum
        else                                                    // no outgoing links
            pageRankNew(v) ← pageRankNew(v) + pageRank(v)

    for v ∈ V do                                                // Receive
        pageRank(v) ← pageRankNew(v)                  // Update the page rank
        pageRankNew(v) ← 0          // Clean up for the next iteration's updating

return pageRank
```

Remark (Algorithm explanation)**.** The following is the detailed explanation of the basic page rank algorithm.

(a) Initialize: For all nodes in $\mathcal{V}$, set page rank to be $1/|\mathcal{V}|$.

(b) Send: Basic page rank updates: each node divides of its current page rank equally across its outgoing links and sends the value to the node at the end. If there are no outgoing links, then the node retains its page rank.

(c) Receive: Each node updates its page rank to be the sum of the values it receives on its incoming links. If a node has retained its page rank, then it adds it to this value.

Note (Conservation of the page rank)**.** Page rank is never lost over gained in the network, it is just passed around. This means no normalization is required, the page ranks always sum to 1, with all terms non-negative.

But then there are some problems we care about.

Problem 4.3.1. Will it converge?

Answer. It will converge in most cases. ⊛

Example. Consider a cycle graph, with different initial page rank. It will not converge! All values just cycling around.

Problem 4.3.2. To what if it does?

Answer. Eigenvector of some eigenvalues of some matrix associated with the network. ⊛

Problem 4.3.3. Is the final state always interesting?

Answer. It may not always be interesting. ⊛

Our goal is to have

1. convergence for all starting vectors;
2. the final vector to be relevant to the graph.

4.4 Analysis of Page Rank

Let's denote the page rank of node i as r_i throughout the following discussion.

(a) Initialize: Denote r^{old} as the column vector of page ranks.

(b) Send:

- If $d_i^{\text{out}} > 0$: then $\frac{r_i^{\text{old}}}{d_i^{\text{out}}}$ is sent to neighbor at the end of the outgoing links.
- If $d_i^{\text{out}} = 0$: then r_i^{old} sent to itself.

(c) Receive: Collect all the page rank values. Therefore,

$$r_i^{\text{new}} = \sum_{j:j\to i} \frac{r_j^{\text{old}}}{d_j^{\text{out}}} + \begin{cases} r_i^{\text{old}} & , \text{ if } d_i^{\text{out}} = 0; \\ 0 & , \text{ if } d_i^{\text{out}} > 0. \end{cases}$$

We can indeed write it in a more compact form as follows. We have $r^{\text{new}} = N^\top r^{\text{old}}$, where

$$\text{if } i \neq j,\ (N^\top)_{ij} = \begin{cases} \frac{1}{d_j^{\text{out}}} & , \text{ if } A_{ji} = 1; \\ 0 & , \text{ if } A_{ji} = 0, \end{cases} = \begin{cases} \frac{A_{ji}}{d_j^{\text{out}}} & , \text{ if } d_j^{\text{out}} > 0; \\ A_{ji} & , \text{ otherwise.} \end{cases}$$

$$\text{if } i = j,\ (N^\top)_{ij} = \begin{cases} 1 & , \text{ if } d_i^{\text{out}} = 0; \\ 0 & , \text{ if } d_i^{\text{out}} > 0, \end{cases} = \begin{cases} 1 & , \text{ if } d_i^{\text{out}} = 0; \\ A_{ii} & , \text{ if } d_i^{\text{out}} > 0. \end{cases}$$

Hence,

$$N_{ij} = \begin{cases} \frac{A_{ij}}{d_i^{\text{out}}} & , \text{ if } d_i^{\text{out}} > 0; \\ A_{ij} & , \text{ if } d_i^{\text{out}} = 0 \wedge i \neq j; \\ 1 & , \text{ if } d_i^{\text{out}} = 0 \wedge i = j. \end{cases}$$

4.4.1 Eigenvalues of N

After L steps, we have

$$r(L) = (N^\top)^L r(0).$$

To study what happens as $L \to \infty$, we suppose there is a convergence vector r such that

$$r = (N^\top) r.$$

We see that r must be an eigenvector for $N^\top$ for eigenvalue 1. Notice that u should be *non-negative* and sums to 1 over entries. Now we have

$$(N^\top) r = r \Rightarrow r^\top N = r^\top,$$

r is a left eigenvector of N with eigenvalue 1.

Note. N has an eigenvalue 1.

Proof. From what we have derived for N_{ij}, we have

$$\sum_{j\in\mathcal{V}} N_{ij} = \begin{cases} \sum_{j\in\mathcal{V}} \frac{A_{ij}}{d_i^{\text{out}}} = \frac{\sum_{j\in\mathcal{V}} A_{ij}}{d_i^{\text{out}}} = \frac{d_i^{\text{out}}}{d_i^{\text{out}}} = 1 & \text{, if } d_i^{\text{out}} > 0; \\ \underbrace{1}_{j=i} + \underbrace{\sum_{j\neq i} A_{ij}}_{=d_i^{\text{out}}=0} = 1 + 0 = 1 & \text{, if } d_i^{\text{out}} = 0. \end{cases}$$

Hence, we conclude that N is a matrix such that the row sums are always 1, so

$$N\vec{1} = \vec{1},$$

which shows that 1 is an eigenvalue of N, with $\vec{1}$ being a right eigenvector! ⊛

We can ask about the left eigenvector — it exists. Corresponding to $\vec{1}$.

Problem 4.4.1. What do we know about the entries?

Answer. We have

(a) row sums on 1

(b) non-negative entries

(c) row stochastic matrix (associated with Markov chains)

⊛

Remark. Each row vector is a probability mass function (PMF) on $\{1, 2, \ldots, |\mathcal{V}|\}$.

4.5 Perron-Frobenius Theorem

In the following lectures, we will study so-called Perron-Frobenius theorem, which is a theorem about non-negative matrices.

4.5.1 Irreducible

We start with irreducible matrix.

Definition 4.5.1 (Irreducible). A non-negative matrix $A_{n\times n}$ is *irreducible* if for all $i, j = 1, \ldots, n$, there exists $k(i, j) \in \mathbb{Z}$ depends on i, j such that $k(i, j) > 0$ and

$$(A^k)_{ij} > 0.$$

A simple observation leads to the following equivalent characterization of irreducibility.

Remark. We can express the condition to be irreducible as there exists a $k > 0$ such that

$$\underbrace{A_{im_1}A_{m_1m_2}\ldots A_{m_{k-1}j}}_{k \text{ terms}} > 0.$$

Proof. We have

$$(A^2)_{ij} = \sum_k A_{ik}A_{kj}$$
$$(A^3)_{ij} = (AA^2)_{ij} = \sum_k A_{ik}(A^2)_{kj} = \sum_k \sum_l A_{ik}A_{kl}A_{lj}$$
$$\vdots$$
$$(A^n)_{ij} = \sum_{k_1}\sum_{k_2}\cdots\sum_{k_n} A_{ik_1}A_{k_1k_2}A_{k_2k_3}\dots A_{k_nj}$$

A is non-negative so $A^{n+1}_{ij} > 0 \Rightarrow$ at least one of the sums > 0, by re-indexing we have the result. ⊛

We see that the above equivalent condition is still not easy to work with, hence we introduce the following definition.

Definition 4.5.2 (Adjacency matrix of a matrix)**.** Given a matrix A, we define the *adjacency matrix of* A as $\widetilde{A}$ such that

$$\widetilde{A}_{ij} = \begin{cases} 1, & \text{if } A_{ij} > 0 \\ 0, & \text{if } A_{ij} = 0 \end{cases}.$$

Note. $\widetilde{A}$ is an adjacency matrix of some directed graph with n nodes.

We then see that

$$A_{im_1}A_{m_1m_2}\dots A_{m_{k-1}j} > 0 \Leftrightarrow \widetilde{A}_{im_1} = \widetilde{A}_{m_1m_2} = \cdots = \widetilde{A}_{m_{k-1}j} = 1.$$

This implies the irreducibility of $A \Leftrightarrow$ irreducibility of $\widetilde{A} \Leftrightarrow \mathcal{G}$ corresponding to $\widetilde{A}$ is strongly connected, i.e., $\forall i, j$ we can find a path on $\mathcal{G}$.

Remark. This graph $\mathcal{G}$ can be non-simple graph.

We see that to check for a non-negative matrix A, whether $(A^{n+1})_{ij} > 0$ or not, we first find the graph $\mathcal{G}$ corresponding to $\widetilde{A}$. If $(A^{n+1})_{ij}$ is positive $\Leftrightarrow$ there is a path in $\mathcal{G}$ from i to j.

Lecture 11: Scaled Page Rank

6 Oct. 12:30

As previously seen. Basic page rank algorithm: If the page rank converges, then we must have

$$r^* = (N^\top)r^* \Leftrightarrow (r^*)^\top N = (r^*)^\top,$$

where r^* is a left eigenvector of N for eigenvalue 1.

Remark. Row stochastic property $\Rightarrow$ 1 is eigenvalue.

4.5.2 Periodicity

Definition 4.5.3 (Period)**.** Given $A_{n\times n}$ and $i \in \{1, \dots, n\}$, find all $m > 0$ such that

$$(A^m)_{ii} > 0.$$

And the greatest common divisor of them is called the *period of* i.

Definition 4.5.4 (Aperiodic). Given $A_{n\times n}$ and $i \in \{1,\dots,n\}$, if the period is 1 for node i, then i is said to be *aperiodic*. And A is said to be aperiodic all i are *aperiodic*.

Remark. We see that

- A sufficient condition for aperiodic is $A_{ii} > 0$ since it implies the period is 1 for i.
- If A is a positive matrix[a], then we know that

$$\begin{aligned} A_{ij} > 0 \ \forall i,j &\Rightarrow \text{irreducible} \\ A_{ii} > 0 \ \forall i \quad &\Rightarrow \text{aperiodic}. \end{aligned}$$

Notice that this is only a simple sufficient condition for a matrix to be irreducible and aperiodic.

[a] All entries are positive

4.5.3 Perron-Frobenius Theorem

We now start to study our main theorem, the Perron-Frobenius theorem.

Definition 4.5.5 (Spectral radius). The eigenvalue of A with the largest absolute value is called the *spectral radius* or the *Perron-Frobenius eigenvalue* of A, denoted as $\rho(A)$.

Theorem 4.5.1 (Perron-Frobenius theorem). Let A be an irreducible non-negative matrix, then the spectral radius is positive and has multiplicity 1.

Both the right eigenvector $(\vec{v}, A\vec{v} = \rho(A)\vec{v})$ and the left eigenvector $(\vec{w}, \vec{w}^\top A = \rho(A)\vec{w}^\top)$ can be taken to be positive with

$$\vec{w}^\top \vec{v} = 1.$$

Furthermore, we have

(a) The following holds

$$\lim_{m\to\infty} \frac{1}{m} \sum_{k=0}^{m-1} \frac{A^k}{(\rho(A))^k} = \vec{v}\vec{w}^\top_{(n\times n)}.$$

(b) The following holds

- Row sum bound:

$$\min_i \sum_j A_{ij} \le \rho(A) \le \max_i \sum_j A_{ij}.$$

- Column sum bound:

$$\min_j \sum_i A_{ij} \le \rho(A) \le \max_j \sum_i A_{ij}.$$

(c) If A is also aperiodic, then

$$\lim_{m\to\infty} \frac{A^m}{(\rho(A))^m} = \vec{v}\vec{w}^\top.$$

Proof. We omit the proof, but it's worth checking out. See here for a short proof. ■

As previously seen. For the basic page rank algorithm:

1. N is non-negative and row stochastic, so we know $\rho(N) \le 1$. But since we already showed 1 is one of the eigenvalues of $N \Rightarrow \rho(N) = 1$.
2. $N \to \widetilde{A}$: Adjacency matrix of original graph plus self-loops for any nodes with no outgoing links.

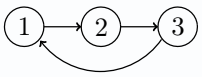
We can now say something about N.

Remark. We don't certainly have that N is irreducible.

Proof. Since $\rho(N) = 1$ but the multiplicity can be > 1 or eigenvectors may not all be positive. From Theorem 4.5.1, N cannot be irreducible in both cases. ⊛

Remark. We don't certainly have that N is aperiodic.

Proof. Consider the following network:



For this network, we have

$$N = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad N^2 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad N^3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and for higher power it just repeats. We see that the period for all three nodes are all 3, which means that N is not aperiodic. ⊛

This is problematic, since for basic page rank algorithm, we have

$$r(k) = (N^\top)^k r(0),$$

hence we are interested in the case that $(N^\top)^k$ converges.

4.6 Scaled Page Rank Algorithm

Scaled page rank is a modification of page rank that gives a matrix $\widetilde{N}(s)$ that is always irreducible and aperiodic with $s \in (0, 1)$. We first give the algorithm to calculate it.

Algorithm 4.3: Scaled Page Rank Algorithm

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, scale factor $s \in (0, 1)$, iteration L
Result: SPageRank

for $v \in \mathcal{V}$ **do** // Initialize
  SPageRank$(v) \leftarrow 1/|\mathcal{V}|$

for $i = 1, \ldots, L$ **do**
  for $v \in \mathcal{V}$ **do** // Send
    neighborNum $\leftarrow |\{v \to u \in \mathcal{E}\}|$
    if neighborNum $\neq 0$ **then** // divides SPageRank(v) to its (out)neighbor
      for $v \to u \in \mathcal{E}$ **do**
        SPageRankNew$(u) \leftarrow$ SPageRankNew(u) + SPageRank(v)/neighborNum
    else // no outgoing links
      SPageRankNew$(v) \leftarrow$ SPageRankNew(v) + SPageRank(v)

  for $v \in \mathcal{V}$ **do** // Receive
    SPageRank$(v) \leftarrow$ SPageRankNew(v) // Update the page rank
    SPageRankNew$(v) \leftarrow 0$ // Clean up for the next iteration's updating
  for $v \in \mathcal{V}$ **do** // Scale
    SPageRank$(v) \leftarrow s \times$ SPageRankNew(v) // Scaled down by s
    SPageRank$(v) \leftarrow \frac{1-s}{|\mathcal{V}|}$ // Re-distribute page rank

return SPageRank

Remark (Algorithm explanation). The following is the detailed explanation of the scaled page rank algorithm.

(a) Initialization: same as basic page rank algorithm.

(b) Send: same as basic page rank algorithm.

(c) Receive: same as basic page rank algorithm.

(d) Scale: Scale down all page ranks by s (multiply by s) and add $(1-s)/|\mathcal{V}|$ to the page ranks of all nodes.

4.7 Analysis on Scaled Page Rank

Again, let's denote the scaled page rank of node i as r_i throughout the following discussion.

$$r_i^{\text{new}} = s\left(\sum_{j:\ A_{ji}=1} \frac{u_j^{\text{old}}}{d_j^{\text{old}}} + \begin{cases} u_i^{\text{old}} & , \text{ if } d_i^{\text{out}} = 0; \\ 0 & , \text{ otherwise;} \end{cases}\right) + \frac{1-s}{|\mathcal{V}|}.$$

Then we have

$$r^{\text{new}} = s(N^\top r^{\text{old}}) + \frac{1-s}{|\mathcal{V}|}\vec{1} = s(N^\top u^{\text{old}}) + \frac{1-s}{|\mathcal{V}|}\vec{1}\underbrace{(\vec{1}^\top r^{\text{old}})}_{\text{add to 1}} = \underbrace{\left(sN + \frac{1-s}{|\mathcal{V}|}\vec{1}\vec{1}^\top\right)^\top}_{\widetilde{N}(s)^\top} r^{\text{old}}.$$

Finally, we have the rule of updating scaled page rank being $r^{\text{new}} = (\widetilde{N}(s))^\top r^{\text{old}}$, hence

$$r(k) = \left(\widetilde{N}(s)^\top\right)^k r(0).$$

Note. We see that $\widetilde{N}(s)$ is

- non-negative
- row stochastic
- strictly positive with all entries at least $(1-s)/|\mathcal{V}|$

Firstly, since $\widetilde{N}(s)$ is non-negative and is row-stochastic, we see that $\widetilde{N}(s)\vec{1} = \vec{1}$, hence $\rho(\widetilde{N}(s)) = 1$.

Moreover, since $\widetilde{N}(s)$ is strictly positive, from the remark, we have that $\widetilde{N}(s)$ is both irreducible and aperiodic. With the non-negativity, we can then apply Theorem 4.5.1 (c) to get

$$\lim_{m\to\infty} \frac{(\widetilde{N}(s))^m}{(\rho(\widetilde{N}(s)))^m} = \lim_{m\to\infty} (\widetilde{N}(s))^m = \vec{v}\vec{w}^\top,$$

where $\vec{v}$ is the right eigenvector of $\widetilde{N}(s)$ and $\vec{w}$ is the left eigenvector of $\widetilde{N}(s)$ with $\vec{w}^\top\vec{v} = 1$.

And since $\widetilde{N}(s)$ is row stochastic, we have $\vec{v} = \vec{1}$, hence $\vec{w}^\top\vec{1} = 1$, which means the sum of all entries is 1 and $\vec{w}$ has all positive entries.

If $(\vec{v}, \vec{w})$ are right and left eigenvectors of $\widetilde{N}(s) \Leftrightarrow (\vec{w}, \vec{v})$ are the right and left eigenvectors of $\widetilde{N}(s)^\top$.

Remark. Scaled page rank converges to r^* where

$$r^* = \widetilde{N}(s)^\top r^*,$$

which is the right eigenvector of $\widetilde{N}(s)^\top$. But it's the same as the left eigenvector of $\widetilde{N}(s)$, which is just $\vec{w}(s)$.

Note. In practice, the scale factor is usually set as $s \cong 0.8 \sim 0.85$.

Lecture 12: Page Rank, Markov Chain and Randomness

11 Oct. 12:30

Recall that we want to measure the centrality of a network. We did this by introducing Theorem 3.2.1 and Theorem 4.5.1.

We first look at following two problems.

Problem 4.7.1 (Eigenvector centrality). For A be an adjacency matrix, which is a non-negative matrix with $r(0) = \vec{1}$ and

$$r(k) = Ar(k-1), \qquad \forall k = 1 \dots$$

Then if it is irreducible and aperiodic, then

$$\lim_{k\to\infty} \frac{r(k)}{(\rho(A))^k} = v(w^\top r(0))$$

where $\rho(A)$ is the unique spectral radius, v is the right eigenvector of A and w is the left eigenvector of A, and $w^\top v = 1$ for $\rho(A)$.

We then have

$$\lim_{k\to\infty} \frac{A^k}{(\rho(A))^k} = vw^\top.$$

Remark. Notice that

- v would be the eigenvector centrality measure.
- $r^* = Ar^*$ need not hold because 1 need not be an eigenvalue.
- Since
$$(r(k))_i = \sum_j A_{ij} r_j(k-1) = \sum_{j\,:\,i\to j} r_j(k-1),$$
so we see that nodes that either point to many nodes, or important nodes or both, will have higher value.

Problem 4.7.2 (Katz centrality). Again, consider $r(k) = Ar(k-1)$, but this time we modify the equation to be

$$r(k) = \alpha A r(k-1) + \beta\vec{1}$$

where α is aiming to dampen the degree, and β is aiming to add a constants weight for each node to increase the fairness. Similarly, any solution will satisfy

$$r^* = \alpha A r^* + \beta\vec{1} \Leftrightarrow (I - \alpha A)r^* = \beta\vec{1}.$$

Definition 4.7.1 (Katz centrality). The solution r^* of $r^* = \alpha A r^* + \beta\vec{1}$ is called the *Katz centrality* if it exits.

Now, if $I - \alpha A$ is invertible, then

$$r^* = \beta(I - \alpha A)^{-1}\vec{1}.$$

Noting that

$$\det(\lambda I - A) = 0 \Leftrightarrow \det(I - \lambda A) = 0,$$

which implies α should not be the inverse of an eigenvalue of A. Also, we need that r^* to be positive.

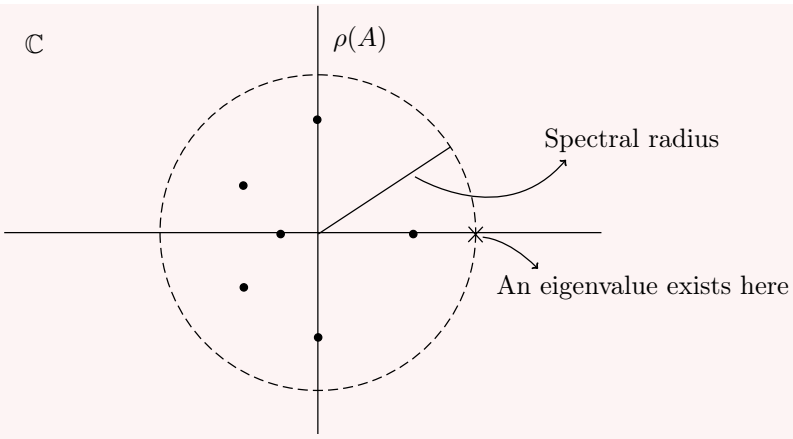


Figure 4.1: Katz Centrality

Then for all $1/\alpha > \rho(A) \Leftrightarrow \alpha\rho(A) < 1 \Leftrightarrow \rho(\alpha A) < 1$, α can't be the inverse of any eigenvalues of A since we see that αA is non-negative with spectral radius < 1.

Now, if αA is non-negative with spectral radius strictly less than one, then we have

$$(I - \alpha A)^{-1} = I + (\alpha A) + (\alpha A)^2 + \dots$$

just like the geometry series of $\frac{1}{1-k}$ for $|k| < 1$. We then have

$$r^* = \beta \left(\sum_{i=0}^{\infty} (\alpha A)^i \right) \vec{1}.$$

Remark. Since

$$\rho(A) \leq \text{ the maximum of row sum of } A = \text{ the highest out-degree},$$

we see that

$$\underline{\alpha \times \text{the highest out-degree} < 1}$$

is a sufficient condition for this to hold.

Note. Katz centrality is commonly used.

Chapter 5

Stochastic Process

The motivation is simple: we want to understand the scaled page rank via another viewpoint, i.e., via Markov chain. This is a special kind of stochastic process, and surprisingly, this can interpret the scaled page rank algorithm nicely.

As previously seen. We have seen that $\widetilde{N}(s)$ is formed in a way such that

$$\widetilde{r}(k) = \widetilde{N}(s)^\top \widetilde{r}(k-1)$$

with $\widetilde{r}(0) = \frac{1}{|\mathcal{V}|}\vec{1}$, where $\widetilde{r}(k)$ is the vector form of the scaled page rank. Then, we have

$$\widetilde{r}(k) = \left(\widetilde{N}(s)^\top\right)^k r(0),$$

which suggests that an **iterated algorithm** known as *power iteration* or *power method*. We see that this will provide a uniform decrease in the relative error of the entries. Specifically, let v be the left eigenvalues, then we have

$$\max_{i\in\mathcal{V}} |(r(k))_i - v_i| \leq c_0 s^k$$

where s is the scaled page rank factor. Then, by using this, if we set the error tolerance as ϵ, i.e., we want the right-hand side to be less than ϵ, we have

$$\#\text{iteration} = \frac{\log(\epsilon)}{\log(s)} \times \text{nnz}(N),$$

where $\text{nnz}(N)$ is the number of non-zero elements in N.

Remark. A sparse graph like Internet will typically have

$$\text{nnz}(N) \sim \Theta(n^2),$$

where n is the number of the nodes.

5.1 Probability

To introduce Markov chain, we start with probability.

As previously seen (Probaility review)**.** We let

- Σ being our sample space.
- $\mathcal{F}$ being the σ-algebra (σ-field). This is essentially just a collection of events subsets of the

sample space.

- $\Pr\colon \mathcal{F} \to [0,1]$ is the probability with values in $[0,1]$ assigned to events.

For further and rigorous review, see Real Analysis note (MATH 597).

We first see some examples of discrete cases.

Example (Coin toss). Let $\Omega = \{\mathrm{H}, \mathrm{T}\}$ corresponding to coin toss, where H is head, T is tail. We see that

$$\mathcal{F} = \{\varnothing, \{\mathrm{H}\}, \{\mathrm{T}\}, \{\mathrm{H}, \mathrm{T}\}\}.$$

If we let the probability of event H as p, then we have

$$\Pr(\varnothing) = 0, \quad \Pr(\{\mathrm{H}\}) = p, \quad \Pr(\{\mathrm{T}\}) = 1 - p, \quad \Pr(\{\mathrm{H}, \mathrm{T}\}) = 1.$$

Example (Dice toss). Let $\Omega = \{1,2,3,4,5,6\}$ corresponding to faces of a die. Then $\mathcal{F}$ is the power set of Ω. If the die is fair, then $\mathbb{P}$ is so-called uniform distribution, which assign an equal value to every event in Ω. For example,

$$\Pr(\{1,2,3\}) = \frac{|\{1,2,3\}|}{6} = \frac{3}{6} = \frac{1}{2}.$$

For continuous cases, we have the following examples.

Example (Uniform distribution). Let $\Omega = [0,1]$, and $\mathcal{F}$ is the Borel[a] σ-algebra which includes all intervals in $[0,1]$. If we let $\mathbb{P}$ be the uniform distribution on $[0,1]$, for every $x \in [0,1]$, we have

$$\Pr([0,x]) = x.$$

If we define $F(x) \coloneqq \Pr([0,x])$, this is known as the cumulative distribution function, of CDF in short.

[a] Borel usually means the standard topology on Euclidean space.

Example (Exponential distribution). Let $\Omega = [0,\infty)$, and let $\mathcal{F}$ be the Borel σ-algebra with $\mathbb{P}$ the exponential distribution with parameter $\lambda \in (0,\infty)$. Then for all $x \in [0,\infty)$, the CDF is defined as

$$F(x) = \Pr([0,x]) = 1 - e^{-\lambda x}.$$

Example (Normal distribution). Let $\Omega = \mathbb{R} = (-\infty, +\infty)$, and $\mathcal{F}$ be the Borel σ-algebra with $\mathbb{P}$ be the normal distribution with mean μ and variance σ^2. Then for all $x \in (-\infty, +\infty)$, the CDF is defined as

$$F(x) = \Pr\left((-\infty, x]\right) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x'-\mu)^2}{2\sigma^2}} \, \mathrm{d}x'.$$

Since this is an important distribution, we denote the normal distribution with mean μ and variance σ^2 as $\mathcal{N}(\mu, \sigma^2)$.

There are some important properties of probability. Firstly, for two mutually disjoint sets A and $B \in \mathcal{F}$ such that $A \cap B = \varnothing$, we have

$$\Pr(A \cup B) = \Pr(A) + \Pr(B).$$

This can be easily generalized to any finite collection of mutually disjoint subsets

$$A_1, A_2, \ldots, A_n$$

such that

$$A_i \cup A_j = \begin{cases} A_i, & \text{if } i = j \\ \varnothing, & \text{if } i \neq j. \end{cases}$$

We then have

$$\Pr\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \Pr(A_i).$$

Remark. Axioms of probability says that this extends to countably infinity collections of mutually disjoint sets. Namely, consider mutually disjoint sets

$$A_1, A_2, \ldots,$$

we have

$$\Pr\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \Pr(A_i) = \lim_{n\to\infty} \sum_{i=1}^{\infty} \Pr(A_i).$$

Another important property is that $\mathcal{F}$ is closed under (countably many) intersections.

Also, Let Ω be the sample space, then we have $\Pr(\Omega) = 1$. This implies

$$\Pr(\varnothing) = 0.$$

Further, if $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$ and $A \cap A^c = \varnothing$. We then see

$$\Pr(\Omega) = 1 = \Pr(A \cup A^c) = \Pr(A) + \Pr(A^c) \Rightarrow \Pr(A^c) = 1 - \Pr(A).$$

Note. Notice that $\varnothing = \Omega^c$ in our convention.

Lastly, we have the following definition.

Definition 5.1.1 (Mutually independent). We say two sets A, B are *mutually independent* sets if

$$\Pr(A \cap B) = \Pr(A)\Pr(B).$$

Also, we have similar definition for countably infinite collections of sets. We say a countably infinite collection of sets is mutually independent if any finite sub-collection is independent. Namely, for any finite index set $K \subsetneq \mathbb{N}$ with $|K| < \infty$,

$$\Pr\left(\bigcap_{k\in K} A_k\right) = \prod_{k\in K} \Pr(A_k).$$

5.1.1 Random Variable

Let's start with a definition.

Definition 5.1.2 (Random variable). A *random variable* is just a mappings (functions) from the sample space to real number $\mathbb{R}$.

Typically, we let $X\colon \Omega \to \mathbb{R}$ be measurable and denote the Borel σ-algebra on $\mathbb{R}$ as $\mathcal{B}(\mathbb{R})$. In particular, we have $(-\infty, x] \in \mathcal{B}(\mathbb{R})$, or more generally, for all $x \in (-\infty, +\infty)$, $\{\omega\colon X(\omega) \le x\} \in \mathcal{F}$. This allows us to define the cumulative density function of x. In particular,

$$\Pr_X((-\infty, x)) = \Pr(\{\omega\colon X(\omega) \le x\}).$$

Same as probability, we can define independence of a collection of random variables.

Definition 5.1.3 (Independent). Let X and Y be two random variables. We say X, Y are *independent*

if for all $A, B \in \mathcal{B}(\mathbb{R})$,

$$\Pr(\{x \in A\} \cap \{y \in B\}) = \Pr(\{x \in A\}) \Pr(\{y \in B\}).$$

Remark. Notice that we are abusing the notation here. In the definition, x is sampled from X and y is sampled from Y without explicitly mentioning. In reality, we have

$$\{x \in A\} \cup \{y \in B\} = \{\omega \in \Omega \colon X(\omega) \in A \wedge Y(\omega) \in B\},$$

we see that we need

$$\{\omega \in \Omega \colon X(\omega) \in A\} \in \mathcal{F} \text{ and } \{\omega \in \Omega \colon Y(\omega) \in B\} \in \mathcal{F}.$$

Note (Independence via expectation). We can also view independence from the probability density function $f_X(x)$. We have

$$\mathbb{E}[f(x)] \coloneqq \begin{cases} \sum_{i=1}^{n} f_X(x_i) \Pr(x = x_i) \\ \int_{-\infty}^{\infty} f_X(x) \, \mathrm{d} \underbrace{F_X(x)}_{\text{CDF}} \end{cases}$$

where $f_X(x)$ is probability density function. Then in this case, we say X and Y are independent if for any f and g,

$$\mathbb{E}[f(x)g(y)] = \mathbb{E}[f(x)] \, \mathbb{E}[g(y)].$$

Example. Let $f_X(x) = x^2$ and $g_Y(y) = y^3$. Then if X and Y are independent, we will have

$$\mathbb{E}\left[x^2 y^3\right] = \mathbb{E}\left[x^2\right] \mathbb{E}\left[y^3\right].$$

Remark. Again, we have similar definition for countably infinite collection of random variables being independent. Let random variables being

$$X_1, X_2, \ldots$$

associated with the probability density function $f_1, f_2, \ldots$. Then, we say this collection of random variables is independent if for any finite sub-collection is independent. Namely, for any finite index set $K \subsetneq \mathbb{N}$ such that $|K| < \infty$,

$$\mathbb{E}\left[\prod_{k \in K} f_k(x_k)\right] = \prod_{k \in K} \mathbb{E}[f_k(x_k)].$$

5.2 Stochastic Process

We now define

Definition 5.2.1 (Stochastic process). *Stochastic process*[a], is a collection of random variable induced by a discrete (time) index

$$X_1, X_2, X_3, \ldots.$$

In a compact way, we can write $\{X_i\}_{i=1}^{\infty}$.

[a] Sometimes random process

Similarly to the independent in probability, there is an analogous definition.

Definition 5.2.2 (i.i.d.). A stochastic process is said to be *i.i.d.* if its collection of random variable are all independent and identically distributed.

Lecture 13: Stochastic Process and Markov Chain

Let's first see some examples of i.i.d. stochastic process, where i is the time index unspecified. 20 Oct. 12:30

Example (Biased coin toss). For all i, let $X_i \in \{0, 1\}$ and

$$\Pr(X_i = 1) = p.$$

We denote this as $X_i \sim \text{Beronulli}(p)$.

Example (Dice toss). For all i, let $X_i \in \{1, 2, 3, 4, 5, 6\}$ and

$$\Pr(X_i = j) = \frac{1}{6}.$$

We denote this as $X_i \sim \text{Uniform}(\{1, 2, 3, 4, 5, 6\})$.

Example (Uniform distribution). For all i, let $X_i \sim \text{Uniform}([0, 1])$, where $X_i \in [0, 1]$.

Example (Exponential distirbution). For all i, let $X_i \sim \exp(\lambda)$, then

$$\Pr(X_i > u) = e^{-\lambda u} \qquad \forall u \geq 0.$$

Example (Normal distribution). For all i, let $X_i \sim \mathcal{N}(\mu, \sigma^2)$.

Remark. Notice that our discussion will focus on $\mathbb{E}[X_i] < +\infty$.

The main reason we study i.i.d. stochastic process is because of the following theorem.

Theorem 5.2.1 (Strong law of large numbers). For an i.i.d. sequence $\{X_i\}$, it follows that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} X_i = \mathbb{E}[X].$$

This is so-called the *strong law of large numbers* (SLLN).

Example (Coin toss). If an unfair coin will face up with probability p, then strong law of large numbers tells us that

$$\frac{\#\text{H}}{\#\text{H} + \#\text{T}} \cong p$$

with many enough tosses. Mathematically, we let $Y_i = \mathbb{1}_{\{X_i \in B\}}$, where

$$\mathbb{1}_{\{X_i \in B\}}(\omega) = \begin{cases} 0, & \text{if } X_i(\omega) \in B; \\ 1, & \text{otherwise,} \end{cases}$$

hence $y_i \sim \text{Bernoulli}(p = \Pr(X_i \in B))$. Then we can use the i.i.d. sequence Y_i to describe a series of coin tosses.

5.2.1 Conditional Property

As previously seen. We first note that if $\Pr(A \cap B) = \Pr(A)\Pr(B)$, then A and B are independent.

Now, if A and B are dependent, then given B is happened, we should be able to say something about the probability under the current condition. This suggests the following definition.

Definition 5.2.3 (Conditional probability). Let $\Pr(B) > 0$, then the *conditional probability* of event A under condition B is defined as

$$\Pr(A \mid B) \coloneqq \frac{\Pr(A \cap B)}{\Pr(B)}.$$

Similarly, we have the conditional expectation

Definition 5.2.4 (Conditional expectation). The *conditional expectation* is defined as

$$\mathbb{E}[X \mid X \in A] = \mathbb{E}[X \mid A].$$

Example. We look at an example for discrete random variable X. Let $X \in \Omega$, then

$$\begin{aligned}
\mathbb{E}[X \mid X \in A] &= \sum_{\omega \in \Omega} \omega \Pr(X = \omega \mid X \in A) \\
&= \sum_{\omega \in \Omega} \omega \frac{\Pr(\{X = \omega\} \cap \{x \in A\})}{\Pr(X \in A)} \\
&= \sum_{\omega \in \Omega} \omega \frac{\Pr(X = \omega)}{\Pr(X \in A)} \\
&= \frac{\sum\limits_{\omega \in \Omega} \omega \Pr(X = \omega)}{\Pr(X \in A)}
\end{aligned}$$

Now, consider three events A, B, C, then we have

$$\Pr(X \in A, X \in B, X \in C) \coloneqq \Pr(X \in A \cap B \cap C).$$

Then if A, B, C are independent, then we further have

$$\Pr(X \in A \cap B \cap C) = \Pr(X \in A)\Pr(X \in B)\Pr(X \in C)$$

as we have seen. If they are dependent, given $\Pr(X \in C) > 0$, then

$$\Pr(X \in A, X \in B, X \in C) = \Pr(X \in A, X \in B \mid X \in C)\Pr(X \in C).$$

This suggests the notion of conditional independence as follows to help us further factorize the above equation.

Definition 5.2.5 (Conditionally independent). For events A, B, C, if

$$\Pr(X \in A, X \in B \mid X \in C) = \Pr(X \in A \mid X \in C)\Pr(X \in B \mid X \in C),$$

then A and B are *conditionally independent* given C.

5.3 Markov Chain

We now formally define Markov chain.

Definition 5.3.1 (Markov chain). A *Markov chain* is a dependent stochastic process with a specific form of conditional independence defined as follows. Consider a random variables sequence

$$X_1, X_2, X_3, \dots$$

with discrete time index $1, 2, 3, \dots$.[a] At time $n > 1$, we have $X_n = a_n$ where a_n takes values in a discrete set. Then

$$\begin{aligned}&\Pr(X_1 = a_1, X_2 = a_2, \dots X_{n-1} = a_{n-1}, X_{n+1} = a_{n+1}, \dots, X_{n+m} = a_{n+m} \mid X_n = a_n)\\ =&\Pr(X_1 = a_1, X_2 = a_2, \dots X_{n-1} = a_{n-1} \mid X_n = a_n)\\ &\cdot \Pr(X_{n+1} = a_{n+1}, \dots, X_{n+m} = a_{n+m} \mid X_n = a_n).\end{aligned}$$

[a] Noting that they lie on a discrete space, like real number, labels, etc.

Intuition (Conditional independence property of Markov chains). Knowing where we are, the progress in the future is conditionally independent of how we got to the present, i.e., knowing the present, the past and the future are conditionally independent.

Note. Markov chains will not be i.i.d. sequences of random variables since there will be dependence. But still, the strong law of large number will hold under conditions.

5.3.1 Time Homogeneous Markov Chain

We need to know two things:

- initial distribution $X_0 \sim \mu$ (or $X_1 \sim \mu$)
- one-step transition matrix

$$P_{ij} \coloneqq \Pr(X_{n+1} = j \mid X_n = i) \qquad \forall i, j.$$

Note. Note that

1. Notice that this is not a function of n.
2. The condition here is just for an example from the lecture.

Review the example in lecture

Then we see that the total probability of $\Pr(X_n = i)$ is

$$\begin{aligned}\Pr(X_n = i) &= \sum_j \Pr(X_n = i, X_0 = j)\\ &= \sum_j \Pr(X_0 = j)\Pr(X_n = i \mid X_0 = j)\\ &= \sum_j \mu_j (P^n)_{ji}\\ &= (\mu^\top P^n)_i\end{aligned}$$

where the first step is from the definition of *total probability*. Recall that

$$P_{ij} = \Pr(X_{n+1} = j \mid X_n = i),$$

hence

$$\begin{aligned}\sum_{j\in\Omega} P_{ij} &= \sum_{j\in\Omega} \Pr(X_{n+1} = j \mid X_n = i)\\ &= \Pr(X_{n+1} \in \Omega \mid X_n = i)\\ &= \frac{\Pr(X_{n+1} \in \Omega, X_n = i)}{\Pr(X_n = i)} = \frac{\Pr(X_n = i)}{\Pr(X_n = i)} = 1.\end{aligned}$$

We see that P is row stochastic and $P_{i\cdot}$ is a probability distribution on Ω. So $\rho(P) = 1 \Rightarrow \exists \pi(\text{distribution})$ such that

$$\pi^\top P = \pi^\top.$$

We see that the left eigenvector is a distribution. We have P is irreducible and Ω is finite, then

$$\lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{1}_{\{X_m = i\}} = \pi_i$$

almost surely.

Refinement with aperiodic P,

$$\lim_{n\to\infty} \Pr(X_n = i) = \pi_i$$

Lecture 14: Random Walker and Monte Carlo Estimator

5.4 Page Rank via Markov Chain

25 Oct. 12:30

Recall that the basic page rank $r(n)$ is updated by the equation $r(n+1) = N^\top r(n)$, i.e.,

$$r^\top(n+1) = r^\top(n) N$$

where N is a row stochastic matrix. On the other hand, compare to Markov Chain,

$$\Pr(X_n = \cdot)^\top = \mu^\top P^n = \mu^\top P^{n-1} P = \Pr(X_{n-1} = \cdot)^\top P$$

with $\mu \sim \Pr(X_0 = \cdot)$. We see that the above two equations are the same with

$$r(0) \coloneqq \begin{pmatrix} 1/V \\ \vdots \\ 1/V \end{pmatrix}.$$

Remark. It turns out that basic page rank is looking at a Markov chain! Recall that N_{ij} is defined as

$$N_{ij} = \begin{cases} \dfrac{1}{d_i^{out}}, & \text{if } A_{ij} = 1; \\ 0, & \text{otherwise.} \end{cases}$$

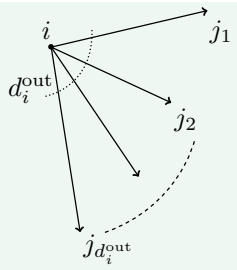
We pick an outgoing neighbor uniformly at random, but no one else.

This is indeed the idea of a random walker, and we formalize it as follows.

Definition 5.4.1 (Random walker)**.** Given a graph, a *random walker* will traverse the graph following some rule with randomness.

In this case, our random walker will follow the simple random walk rule defined as follows.

Definition 5.4.2 (Simple random walk)**.** Suppose the random walker is now at location (node) i, then regardless of how he came to i, he'll pick the next destination uniformly at random from outgoing neighbors of i.



Note. If $d_i^{\text{out}} = 0$, then the random walker following simple random walk will get stuck.

Now, just like in the basic page rank cases, we have

$$\widetilde{N}(s) = sN + \frac{1-s}{|\mathcal{V}|}\vec{1}\vec{1}^{\top}$$

for scaled page rank update equation.

Intuition. Compare to basic page rank, scaled page rank can be thought as follows. When a random walker at location i, we toss a biased coin with $\Pr(\{\text{H}\}) = s$. If we get heads, then we follow the simple random walk rule. Otherwise, if we get tails, we pick any node on the network at random as the next destination.

5.5 Personalized Page Rank via Markov Chain

As previously seen. Recall that we want to associate page rank with random walker.

- Basic page rank: random walker on the graph.
- Scaled page rank: random walker with reset to a uniformly chosen node.

This is a good view point for the **population as a whole user** surfing the webs. Across population reset is to a randomly chosen webpage. We can rank webpages across the population.

Problem 5.5.1. Can we be more specific?

Answer. With the help of reset, yes.

Remark. Personalizing the restart distribution is what allows this.

⊛

Algorithmically, we have the following.

Algorithm 5.1: Personalized Page Rank Scheme

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, iteration T, $v^* \in \mathcal{V}$

for $i = 1, \ldots, T$ **do**
    `RandomWalk(`v`)` `// Perform a random walk starting at` v
return

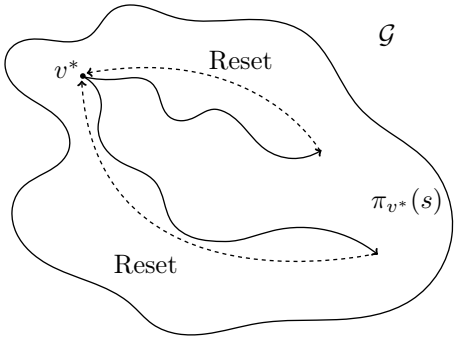


Figure 5.1: Random Walk for personalized page rank

We denote the personalized page rank at v^* as $\pi_{v^*}(s)$. Collect $\pi_{v^*}(s)$ for all $v^* \in \mathcal{V}$, then if we reset uniformly,

$$\pi_{\mathcal{V}}(s) = \frac{1}{|\mathcal{V}|} \sum_{v^* \in \mathcal{V}} \pi_{v^*}(s).$$

Remark. Average of all the personalized page rank, one can get the scaled page rank since averaging with respect to any specific distribution μ distributed on $\mathcal{V}$, we have

$$\pi_{\mu}(s) = \sum_{v^* \in \mathcal{V}} \mu(v^*)\pi_{v^*}(s).$$

We see that if μ is uniformly distributed, we get scaled page rank.

5.6 Monte Carlo Algorithm

In this section, we'll see some simulation-based algorithms which can help us estimate page rank empirically. This kind of simulation-based algorithm which estimate some kind of randomness is called *Monte Carlo algorithm*. We'll analyze the corresponding estimators' quality in terms of their bias, variance, and so on. Let's start with the first one.

5.6.1 First Page Rank Estimator

We now introduce the first simulation-based algorithm to estimate page rank. The very first one we should do is the random walk where we essentially implement simple random walk.

Algorithm 5.2: Random Walk

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, steps T, s
Result: A list arrival, ending node $v \in \mathcal{V}$

for $v \in \mathcal{V}$ **do** // Initialize
  arrival$(v) \leftarrow 0$
$v \leftarrow$`random(`$\mathcal{V}$`)` // select a uniformly random node to start
arrival$(v) \leftarrow 1$

for $i = 1, \ldots, T$ **do**
  result $\leftarrow$`TossCoin(`s`)` // Toss a biased coin with $\Pr(\{\mathrm{H}\}) = s$, $\Pr(\{\mathrm{T}\}) = 1 - s$
  if result $=$ H **then** // If heads
    if $d_i^{out} \neq 0$ **then** $v \leftarrow$ `random(`N_i^{out}`)` // Go to v's neighbor randomly
  else // If tails
    $v \leftarrow$`random(`$\mathcal{V}$`)` // reset to a uniformly random node
  arrival$(v) \leftarrow$ arrival$(v) + 1$ // Record arrival at v
return arrival, v

Note. The result of random walk records the time the random walker arrives v for all $v \in \mathcal{V}$ throughout the simulation.

We can describe random walk as a Markov chain. Consider $\{X_n\}_{n\geq 0}$ where X_n is the location of the random walker at time n. We see that

$$
\begin{aligned}
X_0 &\sim \text{uniform}(\{1,2,\dots,|\mathcal{V}|\}) \\
X_1 &= \begin{cases} \text{uniform}(\{1,2,\dots,|\mathcal{V}|\}), & \text{with probability } 1-s; \\ \text{NextHop}(x_0), & \text{with probability } s, \end{cases} \\
&\vdots \\
X_t &= \begin{cases} \text{uniform}(\{1,2,\dots,|\mathcal{V}|\}), & \text{with probability } 1-s; \\ \text{NextHop}(x_{t-1}), & \text{with probability } s \end{cases}
\end{aligned}
$$

with NextHop$(\cdot)$ being

$$
\text{NextHop}(i) = \begin{cases} \text{uniform}(\{N_i^{\text{out}}\}), & \text{if } d_i^{\text{out}} > 0; \\ i, & \text{if } d_i^{\text{out}} = 0. \end{cases}
$$

Note. This is irreducible and aperiodic.

Suppose we run the random walk for T iterations, we now analyze our first version of random walk by Markov Chain with the help of the strong law of large numbers. A naive estimation of π_v suggests by random walk is

$$
\hat{\pi}_v^{\top}(s) = \frac{1}{T}\sum_{n=0}^{T-1} \mathbb{1}_{\{X_n = v\}} \qquad \forall v \in \mathcal{V}.
$$

When $T \to \infty$, we have

$$
\lim_{T\to\infty} \frac{1}{T}\sum_{n=0}^{T-1} \mathbb{1}_{\{X_n = v\}} = \pi_v(s) \qquad \forall v \in \mathcal{V}
$$

guaranteed by SLLN almost surely.

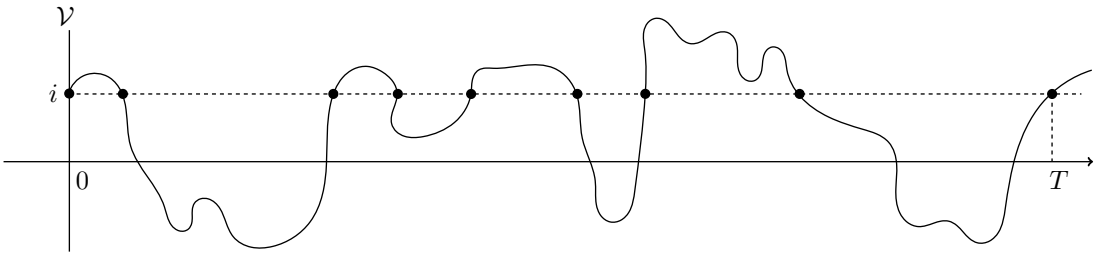


Figure 5.2: First Estimator with $\hat{\pi}_i(s) = 9/T$.

But this estimator has two main issues.

Problem 5.6.1 (Speed of convergence). Not clear how long we should run this for. Namely, we need to be able to say something about

$$
\mathbb{E}\left[\left|\hat{\pi}_v^{\top}(s) - \pi_v(s)\right|^2\right]
$$

in T to ensure the speed of convergence.

Problem 5.6.2 (Biased). The estimator is a **biased** estimator. Recall that this estimator is unbiased if

$$
\mathrm{E}\left[\hat{\pi}_v^{\top}(s)\right] = \pi_v(s);
$$

and if it is biased,

$$\hat{\pi}_v^\top(s) - \pi_v(s) \neq 0.$$

We see that $\hat{\pi}_v^\top(s)$ is indeed biased since

$$\mathrm{E}\left[\hat{\pi}_v^\top(s)\right] = \frac{1}{T}\sum_{n=0}^{T-1}\mathbb{E}\left[\mathbb{1}_{\{X_n=v\}}\right] = \frac{1}{T}\sum_{n=0}^{T-1}\Pr(X_n = v) = \frac{1}{T}\sum_{n=0}^{T-1}(\mu_0 P^n)_v \underset{T\to\infty}{\to} \pi_v(s),$$

where P is the one-step transformation matrix. This only converges when $T \to \infty$ by Perron-Frobenius theorem since the aperiodicity.

5.6.2 Second Page Rank Estimator

We now try to improve random walk we just proposed. Recall that

$$\widetilde{N}(s) = sN + (1-s)\frac{1}{|\mathcal{V}|}\vec{1}\vec{1}^\top.$$

Notice that $\pi(s)$ is the left eigenvector of $\widetilde{N}(s)$ with eigenvalue 1, i.e.,

$$\pi^\top(s)\widetilde{N}(s) = \pi^\top(s).$$

Therefore,

$$\pi^\top(s) = \pi^\top(s)sN + (1-s)\underbrace{\pi^\top(s)\vec{1}}_{=1}\frac{1}{|\mathcal{V}|}\vec{1}^\top = \pi^\top(s)sN + (1-s)\frac{1}{|\mathcal{V}|}\vec{1}^\top,$$

where we group $\pi^\top(s)\vec{1} = 1$ because $\pi(s)$ is a probability vector, which sums to 1. Then

$$\pi^\top(s)(I - sN) = (1-s)\frac{1}{|\mathcal{V}|}\vec{1}^\top.$$

Hence,

$$\pi^\top(s) = (1-s)\frac{1}{|\mathcal{V}|}\vec{1}^\top(I-sN)^{-1} = (1-s)\frac{1}{|\mathcal{V}|}\vec{1}^\top\left(\sum_{i=0}^{\infty}s^iN^i\right) = \sum_{i=0}^{\infty}s^i(1-s)\frac{1}{|\mathcal{V}|}\vec{1}^\top N^i$$

where $N^0 \coloneqq I$.

As previously seen. We have $(I-A)^{-1} = I + A + A^2 + \ldots$.

We now analysis the expression

$$\pi^\top(s) = \sum_{i=0}^{\infty}\underbrace{s^i(1-s)}_{\tau}\frac{1}{|\mathcal{V}|}\vec{1}^\top N^i.$$

Firstly, the term $s^i(1-s) \coloneqq \tau$ is the so-called geometric random variable.

Intuition (Geometric random variable). We can think of τ as the probability of tossing a series of independent biased coins until first tail occurs, since we have

$$\Pr(\tau = 0) = 1 - s, \quad \Pr(\tau = 1) = s(1-s), \ldots, \Pr(\tau = i) = s^i(1-s).$$

Remark. We see that reset happens at time N, hence we look at time $N-1$.

Next, we analyze the term $\frac{1}{|\mathcal{V}|}\vec{1}^\top N^i$, where

- $u = \frac{1}{|\mathcal{V}|}\vec{1}^\top$ is the uniform distribution on the nodes of the graph.
- N^i represents random walk repeated i times.

Intuition. In all, this term represents that we start at a uniformly random node and take i steps as per random walk. Namely, this term represents the distribution of locations of the random walk after i steps.

Now, the suggested algorithm becomes follows.

Algorithm 5.3: Estimate Page Rank ver.2

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, iteration K, s
Result: A list arrival

```
for v ∈ V do                                              // Initialize
    arrival(v) ← 0

for i = 1, ..., K do
    τ ← GeoRandom(s)                          // Toss a geometric random variable
    v ←random(V)                                // select a uniformly random node
    _, v ←RandomWalk(v, τ, s)                 // Random walk starting at v with τ steps
    arrival(v) ← arrival(v) + 1               // Record where the random walker ends up at
return arrival
```

Note. The result of the above algorithm records the time the random walker **ends up** at v for all $v \in \mathcal{V}$ in every random walk.

Denote those K random variables τ as $\tau_1, \tau_2, \ldots, \tau_K$ and the associated X_τ as $X_{\tau_1}, X_{\tau_2}, \ldots, X_{\tau_K}$. We see that

$$\hat{\pi}_v^K(s) = \frac{1}{K} \sum_{k=1}^{K} \mathbb{1}_{\{X_{\tau_k} = v\}}.$$

This is an unbiased estimator since

$$\mathrm{E}\left[\mathbb{1}_{\{X_{\tau_k} = v\}}\right] = \Pr(X_{\tau_k} = v) = \pi_v(s).$$

Also, the variance can be derived as

$$\mathrm{Var}\left[\hat{\pi}_v^K(s)\right] = \frac{\pi_v(s)(1 - \pi_v(s))}{K} \leq \frac{1}{4K}.$$

Remark. Since $n(1-n) \leq 1/4$ for every $n \in [0, 1]$ with equality if $x = 1/2$.

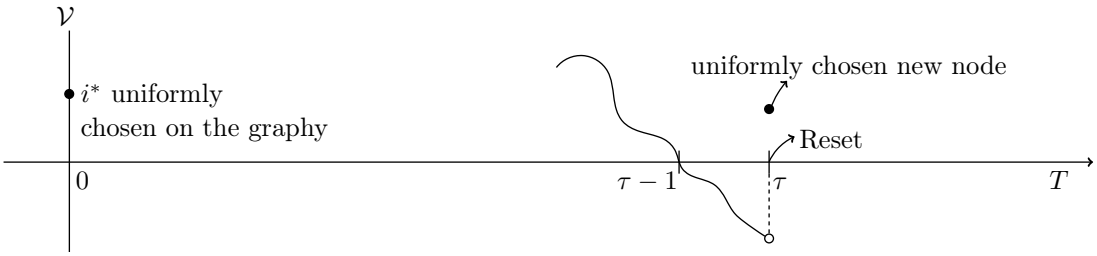


Figure 5.3: Second Estimator with $X_{\tau-1} \sim \pi(s)$

Note. We note that the empirical mean is unbiased while the empirical variance is biased. Let $X_1, X_2, \ldots, X_N$ are i.i.d. with mean μ and variance σ^2.

- $\hat{\mu}_N$ is the empirical mean which is defined as

$$\hat{\mu}_N \coloneqq \frac{1}{N}\sum_{i=1}^{N} X_i$$

with

$$\mathbb{E}\left[\hat{\mu}_N\right] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[X_i\right] = \frac{1}{N}\sum_{i=1}^{N}\mu = \mu,$$

which implies this is an unbiased estimator.

- Denote $\hat{\mathrm{Var}}(N)$ as the empirical variance estimator. It's defined as

$$\hat{\mathrm{Var}}(N) \coloneqq \frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{\mu}_N)^2.$$

This is indeed a biased estimator since

$$\mathbb{E}\left[\hat{\mathrm{Var}}(N)\right] = \left(\frac{N-1}{N}\right)\sigma^2,$$

which is slightly less than the true variance. We then have an unbiased estimator defined as

$$\overline{\mathrm{Var}}(N) \coloneqq \frac{1}{N-1}\sum_{i=1}^{N}(X_{i-\hat{\mu}_N})^2$$

with $\mathbb{E}\left[\overline{\mathrm{Var}}(N)\right] = \sigma^2$.

5.6.3 Third Page Rank Estimator

Recall that

$$\pi(s) = \frac{1-s}{|\mathcal{V}|}\vec{1}^{\top}(I - sN)^{-1}.$$

Now, we define

$$Z \coloneqq (I - sN)^{-1} = \sum_{i=0}^{\infty} s^i N^i$$

with $s^0 \coloneqq 1$ and $N^0 \coloneqq I$. Z is a non-negative matrix since $Z_{ij} \geq 0$. We further define $\sigma_{ij} \coloneqq (1-s)Z_{ij}$ with a matrix Σ whose entries are defined as $(\Sigma)_{ij} \coloneqq \sigma_{ij}$ and $\sigma_{i\cdot}$ is the row of Σ corresponding to node i.

Remark. We see that $\lambda_{i\cdot}$ is the page rank corresponding to the reset distribution where we always reset to node i.

Then, we proposed the following algorithm.

Algorithm 5.4: Estimate Page Rank ver.3

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, iteration M, s
Result: A list arrival

```
for v ∈ V do                                              // Initialize
    arrival(v) ← 0

for i ∈ V do
    for j ∈ 1, ..., M do
        τ^i ← GeoRandom(s)                          // Toss a geometric random variable
        _, v ← RandomWalk(i, τ^i)                 // random walk starting at i with steps τ^i
        arrival(v) ← arrival(v) + 1               // Record where the random walker ends up at
return arrival
```

Note. The result of the above algorithm records the time the random walker **ends up** at v for all $v \in \mathcal{V}$ in every random walk.

If we only take one walk, namely $M = 1$, then

$$\pi_v(s) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \sigma_{iv}(s).$$

Now, for every node i in the network, we run M times with $\tau_1^i, \tau_2^i, \ldots, \tau_M^i$ and $X_{\tau_1}^i, X_{\tau_2}^i, \ldots, X_{\tau_M}^i$ and count the number of arrivals in node v and find the average. Explicitly, we have

$$\hat{\pi}_v(s) = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}_{\left\{X_{\tau_m^i}^i = v\right\}} = \frac{1}{|\mathcal{V}|\, M} \sum_{i=1}^{|\mathcal{V}|} \sum_{m=1}^{M} \mathbb{1}_{\left\{X_{\tau_m^i}^i = v\right\}}.$$

This is another unbiased estimator for the same reason given in the second estimator, but we see that this converges faster and with smaller variance. Specifically, we have

$$\operatorname{Var}\left[\hat{\pi}_v(s)\right] = \frac{1}{|\mathcal{V}|\, M} \left(\pi_j - \frac{\sum_{i=1}^{|\mathcal{V}|} \sigma_{ij}^2}{|\mathcal{V}|} \right)$$

for all $v \in \mathcal{V}$. Hence, this is a better estimator.

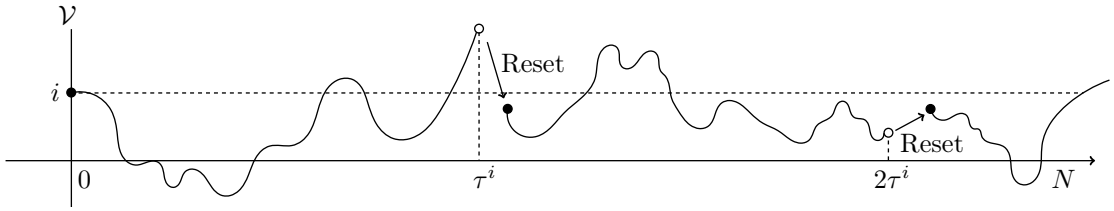


Figure 5.4: Third Page Rank Estimator with $X_{\tau_j - 1}^i \sim \sigma_i(s)$

Problem 5.6.3. Although the estimator found by this algorithm is better than the other two, but it still has its flaws. The main thing is, it throws away so much data, which means, which means it will converge slower.

Lecture 15: Final Page Rank Estimator

27 Oct. 12:30

As previously seen. Main idea is that to construct a Markov chain on nodes of the graph. At none i, the random walker toss a biased coin and determine where to go.

5.6.4 Forth Page Rank Estimator

We introduce the final estimator, which make the full use of the data given, which avoids the problem we faced when using the third estimator.

Formally, to further utilize the data thrown away by the previous algorithm, we just record more data as follows.

Algorithm 5.5: Estimate Page Rank ver.4

Data: A network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, iteration M, s
Result: A list N_t

```
for v ∈ V do                                                    // Initialize
    for t ∈ 1, ... M × |V| do
        N_t(v) ← 0
t ← 0

for i ∈ V do
    for j ∈ 1, ..., M do
        τ^i ← GeoRandom(s)                          // Toss a geometric random variable
        arrival, v ← RandomWalk(i, τ^i)        // random walk starting at i with steps τ^i
        for k ∈ V do                  // Record every node the random walker has been to
            N_t(k) ← N_t(k) + arrival(k)
        t ← t + 1
return N_t
```

Intuition. We see that this algorithm just record every information about the random walker! We see that it utilize the data in the most efficient way.

In this case, we don't need to analyze this estimator, since there is an off-the-shell theorem we can use, Renewal theorem.

Theorem 5.6.1 (Renewal Theorem)**.** We have

$$\pi_v(s) = \frac{\mathbb{E}\left[\#\text{ of visits to node } v \text{ in this reset}\right]}{\mathbb{E}\left[\text{Duration of reset}\right]}$$

for every v, where we have $\mathbb{E}\left[\text{Duration of reset}\right] = \frac{1}{1-s}$, which implies

$$\hat{\pi}_v(s) = \frac{\frac{1}{T}\sum_{i=1}^{T} N_t(k)}{\frac{1}{1-s}} = \frac{1-s}{T}\sum_{t=1}^{T} N_t(v)$$

where $N_t(v)$ is the number of time we visited v in reset duration.

Note. Note that T is $M \times |\mathcal{V}|$ since there are these many reset duration in the algorithm. Also, we see that each reset duration is independent of the other T rest duration starting at uniformly chosen node.

We see that this is also a non-biased estimator, and also, the random variables sequence $\{N_t(v)\}_t$ is i.i.d.

Remark. This estimator does not throw out those useful data, hence the estimator computed by this algorithm converges faster.

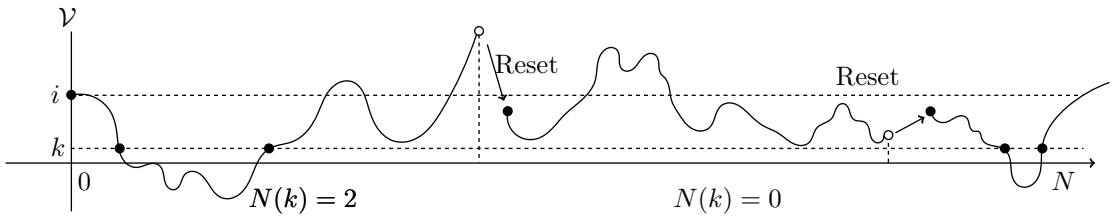


Figure 5.5: Forth Page Rank Estimator which records $N(v)$ for every node.

Chapter 6

Random Graph

We are interested in some functions depend on the number of nodes n in the graph such that $\lim_{n\to\infty} f(n)$ can be understood, namely we are interested in so-called *large graph property* since nearly all networks we have in practice are large. Furthermore, by Theorem 5.2.1, if we assume that the real-work network is formed following some underlying distribution, we can then study some nice random graph models and use it to help us understand the world.

6.1 Order Relationship of Functions

We first introduce some common notations for asymptomatic comparison.

Definition. Given $f(n)$ and $g(n)$, the following notion compare f and g *loosely*.

Definition 6.1.1 (Big-O). $f(n)$ is said to be $O(g(n))$ if there is a positive k such that

$$f(n) \leq k \cdot g(n)$$

for all large enough n.

Definition 6.1.2 (Big-Ω). $f(n)$ is said to be $\Omega(g(n))$ if there is a positive k such that

$$f(n) \geq k \cdot g(n)$$

for all large enough n.

Definition 6.1.3 (Big-Θ). $f(n)$ is said to be $\Theta(g(n))$ if there are positive $k_1 \leq k_2$ such that

$$k_1 \cdot g(n) \leq f(n) \leq k_2 \cdot g(n)$$

for all large enough n.

Definition. Given $f(n)$ and $g(n)$, the following notion compare f and g *strictly*.

Definition 6.1.4 (Small-o). $f(n)$ is said to be $o(g(n))$ if for every positive ϵ, there exists an N large enough such that

$$f(n) \geq \epsilon \cdot g(n)$$

for all $n \geq N$.

Definition 6.1.5 (Small-ω). $f(n)$ is said to be $\omega(g(n))$ if for every positive ϵ, there exists an N large enough such that

$$f(n) \leq \epsilon \cdot g(n)$$

for all $n \geq N$.

Definition 6.1.6 (Same order). $f(n)$ is said to be of the *same order* as $g(n)$ asymptotically if

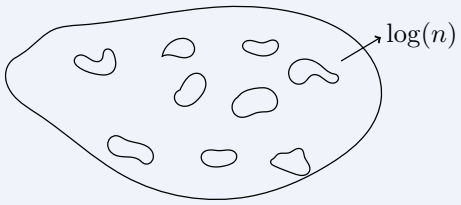
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1.$$

We can write $f(n) \sim g(n)$ for short in this case.

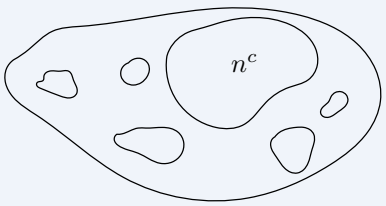
Remark. If $f(n)$ or $g(n)$ are not non-negative, we simply use $|f(n)|$ and $|g(n)|$ and apply the above definitions.

We first consider the size of the largest connected component, and see how does this relation behave as $n \to \infty$. Formally, for a function $f(n)$ which represent the size of a graph, where n is the number of nodes in the graph.

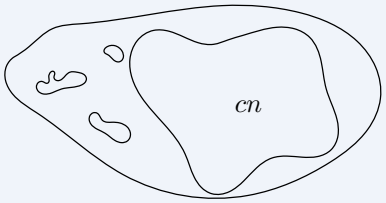
Example (Island). If $f(n) = \log n$, we'll see small islands with $\frac{n}{\log(n)}$ of them without big continent. This is because when $n \to \infty$, $\frac{n}{\log(n)} \to 0$.



Example. If $f(n) = n^c$ with $c < 1$, we'll see a continent but'll shrink as $n \to \infty$. This is because that $\frac{n^c}{n} \to 0$ for $c < 1$ as $n \to \infty$. Better than island but not that great.



Example (Giant component). If $f(n) = cn$, there will be a giant component roughly c percent of nodes in the graph. Since as $n \to \infty$, $\frac{cn}{n} \to c$.



Lecture 16: Random Graph

1 Nov. 12:30

6.2 Erdős-Rényi Random Graphs Family

We introduce two main models for generating random graphs.

- Erdős-Rényi. The random graph is given by $G(n, M)$. A graph is chosen uniformly at random from the collection of all graphs with n nodes and M edges.
- Edward-Gilbert. The random graph is given by $G(n, p)$. The graph is constructed by connecting nodes randomly, with each edge is included in the graph with probability p independently.

Remark. We compare these two models.

- The number of expected edges in $G(n, p)$ is $\binom{n}{2}p$. Hence, by SLLN, nearly all graphs in $G(n, p)$ will have $\binom{n}{2}p$ edges. Then if $pn^2 \to \infty$ and $M = \binom{n}{2}p$, then as n grows, the behavior of $G(n, p)$ should be similar with $G(n, M)$.
- The most commonly used model is $G(n, p)$ since the independent property of edges simplifies lots of analysis.

We first focus on Erdős-Rényi random graphs family.

6.2.1 Uniform and Undirected Model

In this case, the adjacency matrix A is symmetric. This can be seen from constructing the adjacency matrix A such that

$$A = \begin{pmatrix} 0 & & U \\ & \ddots & \\ L & & 0 \end{pmatrix},$$

where we need to specify the upper triangular part U, and the lower triangular part L can then be obtained by symmetry.

Now, let V denotes the number of nodes of the graph we're going to generate, also let $p \in (0, 1)$, we have the following algorithm to generate a random graph.

Algorithm 6.1: Uniform and Undirected Random Graph Generator

Data: $p \in (0, 1)$, V
Result: Adjacency matrix A

$A_{V\times V} = 0_{V\times V}$ // Initialize
for $i = 1, \dots, V-1$ **do**
  for $j = i+1, \dots, V$ **do**
    $A_{ij} \leftarrow$ `Bernoulli(`p`)` // $\Pr(A_{ij} = 1) = p$, $\Pr(A_{ij} = 0) = 1 - p$
return A

Soon, we will study some properties as $V \to \infty$.

6.2.2 Non-Uniform and Undirected Model

Given a non-uniform distribution matrix P with $P = P^\top$, where P is the probability matrix with diagonal entries being 0. Then, we have the following algorithm to generate a random graph.

Algorithm 6.2: Non-Uniform and Undirected Random Graph Generator

Data: P, V
Result: Adjacency matrix A

$A_{V\times V} = 0_{V\times V}$ // Initialize
for $i = 1, \dots, V-1$ **do**
  for $j = i+1, \dots, V$ **do**
    $A_{ij} \leftarrow$ `Bernoulli(`P_{ij}`)` // $\Pr(A_{ij} = 1) = P_{ij}$, $\Pr(A_{ij} = 0) = 1 - P_{ij}$
return A

6.2.3 Uniform and Directed Model

Again, with $p \in (0,1)$, we have the following algorithm to generate a random graph.

Algorithm 6.3: Uniform and Directed Random Graph Generator

Data: $p \in (0,1)$, V
Result: Adjacency matrix A

$A_{V\times V} = 0_{V\times V}$ // Initialize

for $i = 1, \dots, V-1$ **do**
  for $j = 1, \dots, V$ **do**
    $A_{ij} \leftarrow$ `Bernoulli(`p`)` // $\Pr(A_{ij}=1)=p$, $\Pr(A_{ij}=0)=1-p$

return A

We see that for all $i \neq j$,

$$i \neq j \quad A_{ij} = \text{Bernoulli}(p)$$

chosen independently. And because of independence, $A_{ij} \neq A_{ji}$ can occur and

$$\Pr(A_{ij} = A_{ji} = 1) = \Pr(A_{ij}=1)\Pr(A_{ji}=1) = p^2.$$

Compared to undirected case, we see that

$$\Pr(A_{ij} = A_{ji} = 1) = p$$

in an undirected graph.

6.2.4 Non-Uniform and Directed Model

Denote P as the probability matrix with diagonal entries being 0. Then we have the following algorithm to generate a random graph.

Algorithm 6.4: Non-Uniform and Directed Random Graph Generator

Data: P, V
Result: Adjacency matrix A

$A_{V\times V} = 0_{V\times V}$ // Initialize

for $i = 1, \dots, V-1$ **do**
  for $j = 1, \dots, V$ **do**
    $A_{ij} \leftarrow$ `Bernoulli(`P_{ij}`)` // $\Pr(A_{ij}=1)=P_{ij}$, $\Pr(A_{ij}=0)=1-P_{ij}$

return A

We see that for all $i \neq j$,

$$A_{ij} = \text{Bernoulli}(P_{ij})$$

chosen independently.

Example (Stochastic block model). We consider a specific model called *stochastic block model* (SBM) with the probability matrix being:

$$P = \begin{matrix} \\ V_1 \\ V_2 \end{matrix} \begin{matrix} \begin{matrix} V_1 & V_2 \end{matrix} \\ \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \end{matrix}$$

with $P_{12} = P_{21}^\top$, $P_{11} = P_{11}^\top$ and $P_{22} = P_{22}^\top$. In particular,

$$P_{11} = p_{11} \begin{pmatrix} 0 & 1 & \dots & 1 & 1 \\ 1 & 0 & \dots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 0 & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix}_{V_1 \times V_1}, P_{22} = p_{22} \begin{pmatrix} 0 & 1 & \dots & 1 & 1 \\ 1 & 0 & \dots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 0 & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix}_{V_2 \times V_2}$$

$$P_{12} = p_{12} \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix}_{V_1 \times V_2}, P_{21} = p_{21} \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix}_{V_2 \times V_1},$$

where each block matrix is uniform within the block. In the typical set up, we will have $p_{11}, p_{22} \gg p_{12}$. This creates a community structure such that

- p_{11} governs connectivity in community 1.
- p_{22} governs connectivity in community 2.
- p_{12} governs the *cross* connectivity structure.

Remark (Bipartite graph generalization). If we instead set

$$p_{12} \gg p_{11}, p_{22},$$

The generated random graph generalizes to multiple communities by an appropriate block structure.

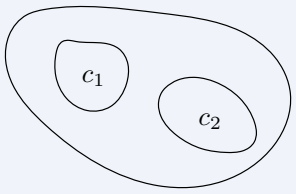


Figure 6.1: For $p_{12} = 0$ and $p_{12}, p_{22} > 0$

Also, this relates to the community detection algorithm. We look at the Laplacian $L = D - A$ and find spectral decomposition and find the smallest eigenvalues and eigenvectors to study community structure. See Section 3.4 for detailed discussion.

6.3 Analysis of Erdős-Rényi Random Graphs Family

As previously seen. To summarize all cases we have seen, we have

- Uniform case. With $p \in (0, 1)$,
  - $p = 0$ - V has isolated nodes.
  - $p = 1$ - Complete graph.
- Non-uniform case. Some values can be 1 or 0, but not all.

Now, let's look at the properties follows the above discussion. In particular, we consider the uniform directed case with $p \in (0, 1)$. We then see that each undirected edge is chosen with probability p. For node i, we have

- The degree is associated with Binomial$(V-1,p)$ such that

$$\Pr(\deg(i)=k)=\binom{V-1}{k}p^k(1-p)^{V-1-k}.$$

- The mean degree is just $(V-1)p$.
- The total number of edges is associated with Binomial $\left(\frac{V(V-1)}{2},p\right)$.
- The mean number of edges is $\frac{V(V-1)}{2}p$.

Now, we can discuss different regimes.

6.3.1 Extremely Sparse Regime

With $p=\min\{c/V,1\}$ for some $c>0$. Then for node i, we have

- The degree is associated with Binomial$(V-1,\frac{c}{V})\to$ Poisson(c) as $V\to\infty$.
- The mean degree is $c\frac{V-1}{V}\to c$ as $V\to\infty$.

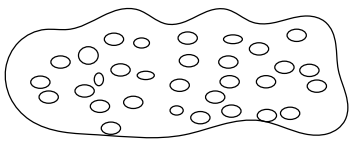
> **Note** (Poisson distribution). Recall the Poisson distribution is defined as
>
> $$\Pr(\text{Poisson}(c)=k)=e^{-c}\frac{c^k}{k!},\qquad k=0,1,\dots$$

We see that for an isolated node i, $\Pr(\deg(i)=0)=e^{-c}>0$, hence as $V\to\infty$, e^{-c} fraction of nodes will be isolated since

$$\Pr(\text{isolated nodes})=(1-p)^{V-1}=\left(1-\frac{c}{v}\right)^{V-1}\to e^{-c},$$

where $\lim_{n\to\infty}\left(1-\frac{1}{x}\right)^x=e^{-1}$.

- The total number of edges is associated with Binomial $\left(\frac{V(V-1)}{2},\frac{c}{V}\right)$.
- The mean number of edges is $\frac{V(V-1)}{2}\frac{c}{V}=\frac{c(V-1)}{2}\sim\Theta(\frac{c}{2}V)$, which is linear in V.

> **Remark.** We can further divide this cases into following subcases:
>
> 1. Sub-critical regime with $0<c<1$. Essentially **isolated nodes**. The largest connected component will have size $O(\log V)$ as $V\to\infty$, and the number of connected components will be
>
> $$\Omega\left(\frac{V}{\log V}\right)$$
>
> as $V\to\infty$.
>
> 
>
> Figure 6.2: Sub-critical regime, at least some islands are $O(\log V)$ size
>
> 2. Super-critical regime with $c>1$. This essentially leads to a (single) giant component. There will exist a c_1 associated with the largest component such that
>
> $$|c_1|\sim f(c_1)V,\qquad f\in(0,1)$$

where f is an increasing function of c with $f(0) = 0$ and $\lim_{c\to\infty} f(c) = 1$. And since $e^{-c} > 0$, so isolated nodes exist. Further, the second-largest component will have

$$|c_2| \sim O(\log V),$$

while all other components are very small.

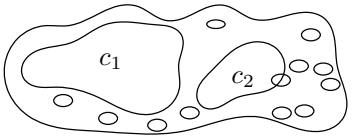


Figure 6.3: Super-critical regime, $e^{-c} > 0$, so isolated nodes exist

3. Critical regime such that $c = 1$. We see that $|c_1| \sim \Theta(V^{\frac{2}{3}})$, $|c_2| O(\log V)$.

Remark (Branching processes). These analyses are using so-called *branching process*.

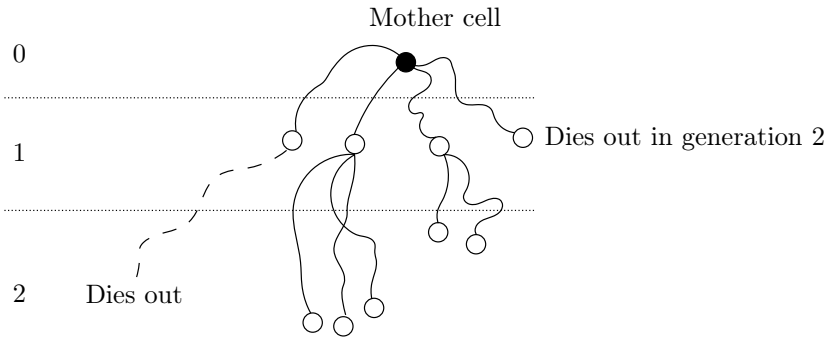


Figure 6.4: Branching processes.

Let c be the mean number of children. Then we have

- $c < 1$: dies out fast
- $c > 1$: can keep growing
- $c = 1$: This is the critical case: dies out but goes much further compare to the first case

6.3.2 Sparse Regime

This is also known as connectivity regime. With $p = \frac{c \log V + a}{V}$ where $a \in \mathbb{R}$ and $c > 0$, more precisely, we have

$$p \sim \frac{c \log V + a}{V} \text{ as } V \to \infty.$$

Asymptotically, as $V \to \infty$, node degrees go to infinity (not necessarily for all nodes). Now, for node i, we have

- The mean degree is

$$(V-1)p = \frac{V-1}{V}(c \log V + a) \sim c \log V + a = \Theta(\log V).$$

- The number of edges is

$$\frac{V(V-1)}{2} p = (V-1)(c \log V + a) = \Theta(V \log V).$$

Remark. Now consider following subcases.

1. $0 < c < 1$: As $V \to \infty$, there will always be isolated nodes. There will be one giant component with almost all nodes.
2. $c > 1$: As $V \to \infty$, the graph is **always connected**.
3. $c = 1$:
   - With probability $e^{-e^{-a}}$, there are isolated nodes.
   - With probability $1 - e^{-e^{-a}}$, the graph is connected.

For these graphs, isolated nodes are the ones stopping, preventing connectivity. The moment you eliminate isolated nodes, connectivity emerges.

6.3.3 Extremely Dense Regime

Now, if $p \sim \Omega(\frac{\log V}{V})$, the graph is extremely dense with connectivity. In this case, the degree distributions as Poisson

$$\Pr(\text{Poisson}(c) > k) \sim e^{-g(c)k},$$

which drop exponentially or geometrically in k since its has geometric tail.

Remark. In practice, we have

$$\Pr(\deg(i) > k) \sim \frac{1}{k^\alpha}$$

for $\alpha > 1$, which has only power-law tail.

Lecture 17: Random Graph

6.4 Real World Graphs

3 Nov. 12:30

In this section, we compare the random graph models with the real-world graphs.

Remark (Dense and sparse). Generally, we say a graph is sparse when the number of edges is linear with respect to V, namely

$$E \sim \Theta(V).$$

And we say a graph is dense when the number of edges is quadratic with respect to V, namely

$$E \sim \Theta(V^2),$$

note that this case is not so common. We can also see this by considering the number of triangles. In the connectivity regime, the number of edges is

$$\Theta(V \log V),$$

which is small relative to what's observed in real-world graphs.

In practice, a real-world graph will have the following properties:

1. The number of edges is linear in V.
2. Giant component usually entice graph(connectivity)
3. Lots of triangles

Compare to the real-world graphs, the Erdős-Rényi random graphs family usually have the following properties, for each different settings:

- Extremely sparse regime:

  - edges is linear in V
  - giant component
  - isolated nodes
  - no triangles as $V \to \infty$
- Sparse regime:
  - edges is $\sim \Theta(V \log V)$
  - most nodes in the giant component
  - can have connectivity
  - Many triangles
- Extremely dense regime:
  - edges is $\sim \Omega(V^{1+\epsilon})$ with $\epsilon > 0$
  - fully connected
  - lots of triangles

6.5 R-MAT Graphs

Besides Erdős-Rényi random graph family, we can use different approaches to generate a random graph. We now introduce so-called *R-MAT graphs*.

> **Intuition.** The main idea is to recursively set an entry of the adjacency matrix.

We assume that the number of nodes is $V = 2^k$ for some k. Now, given four numbers $a, b, c, d \geq 0$ with $a + b + c + d = 1$, we let

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

be a distribution on $\{1, 2, 3, 4\}$. Further, we denote

- E: Target number of edges to be placed on a directed graph. Furthermore, we add entries in the adjacency matrix one edge of a time.
- A: $2^k \times 2^k$ matrix such that

$$A = \begin{array}{c} \\ 2^{k-1} \\ 2^{k-1} \end{array} \overset{\begin{array}{cc} 2^{k-1} & 2^{k-1} \end{array}}{\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}}_{2^k \times 2^k}$$

Then the algorithm to generate an R-MAT graph is as follows.

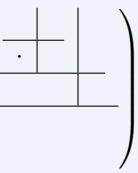
Algorithm 6.5: R-MAT Graph Generator

Data: (a, b, c, d), $V = 2^k$
Result: Adjacency matrix A

1 $A_{V\times V} = 0_{V\times V}$ // Initialize
2
3 **for** $e = 1, \dots, E$ **do**
4 $A' \leftarrow A$
5 **for** $\ell = k, \dots, 2$ **do** // Iterated backwards, choose $A_{2^l\times 2^l}$ with size $2^{l-1}\times 2^{l-1}$
6 r $\leftarrow$rand$([0, 1])$
7 **if** $r \le a$ **then** // $1 \sim a$
8 region $\leftarrow 1$
9 **else if** $a < r \le a + b$ **then** // $2 \sim b$
10 region $\leftarrow 2$
11 **else if** $a + b < r \le a + b + c$ **then** // $3 \sim c$
12 region $\leftarrow 3$
13 **else** // $4 \sim d$
14 region $\leftarrow 4$
15 $A' \leftarrow A'_i$ // Choose the region i
16 **if** $A' = [0]$ **then**
17 $A' = [1]$ // Note that this also changes A's corresponding entry
18
19 **for** $i = 1, \dots, V$ **do** // Clean of to make A a simple directed graph
20 $A_{ii} = 0$
21 **return** A

Example. In picture, R-MAT algorithm is doing the following:

$$M = \left(\quad \right),$$



where we choose $k = 3$ with the sequence $2, 1, 3$.

Note that in line 20, we delete any entries on the diagonal and produce the adjacency matrix A to produce a simple directed graph.

Example (Clean up). Let A be

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix},$$

which has self-loop hence not simple. After cleaning up, we have

$$A' = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Remark. We can do some comparison between what we have already seen.

- Erdős-Rényi Random Graph: Poisson degree distribution with $p = \frac{c}{V}$. Then the tail proba-

bility (CCDF) is

$$\Pr(\deg(i) \geq k) \cong ce^{-\alpha k},$$

which decreases exponentially.

- Real-World graphs. Tail probability is

$$\Pr(\deg(i) \geq k) = \frac{c}{k^{\alpha-1}},$$

which is polynomial decrease if $\alpha \in (2, 3)$. More specifically,

  - $\alpha > 2$:
    - mean degree exists, such that

$$\text{mean degree} \cong \sum_{k=1}^{\infty} k \frac{\widetilde{c}}{k^{\alpha}} = \sum_{k=1}^{\infty} \frac{\widetilde{c}}{k^{\alpha-1}}.$$

      We see that $\alpha - 1 > 1$ is needed for above to converge to a finite number.
    - mean squared of degree exists, such that

$$\text{mean squared of degree} \cong \sum_{k=1}^{\infty} k^2 \frac{\widetilde{c}}{k^{\alpha}} = \sum_{k=1}^{\infty} \frac{\widetilde{c}}{k^{\alpha-2}}.$$

  - $\alpha < 3$: This implies $\alpha - 2 < 1$, hence above will just diverge to $+\infty$.

- In practice, the variance of degrees is exponentially high.

6.6 Preferential Attachment Directed Graph

Yet, we have another algorithm to produce a random graph. Let the (undirected) degree distribution on V being

$$d_1, d_2, \ldots, d_V$$

with $\sum_{i=1}^{V} d_i$ being even. Then by considering a histogram, for $k = \{0, 1, \ldots, V-1\}$, we have

$$P_k^{\alpha} = \frac{1}{V} \sum_{i=1}^{V} \mathbb{1}_{\{d_i = k\}} = \frac{\#\text{ of nodes with degree } k}{V}.$$

We make the following assumptions for constructing a preferential attachment graph.

- Nodes arrive in sequence $1, 2, \ldots, V$
- Each node has an out-degree. Out going edges that need to be paired with nodes earlier in sequence such that

$$d_i^{\text{out}} \leq i - 1.$$

Now, with $p \in (0, 1)$, the following algorithm will generate a preferential attachment graph.

Algorithm 6.6: Preferential Attachment Graph Generator

Data: $p \in (0,1)$, V, $\{d_i\}_{i=1}^{V}$
Result: $\mathcal{G}$

$\mathcal{V} \leftarrow \{1, 2, \ldots, V\}$
$\mathcal{E} \leftarrow \varnothing$
$\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$ // Initialize

for $i = 1, \ldots, V$ **do** // Assume nodes arrive in order
  unpaired $\leftarrow \{1, 2, \ldots, i-1\}$ // Unpaired nodes earlier to i
  for $e = 1, \ldots, d_i^{out}$ **do**
    result $\leftarrow$ `TossCoin(`p`)` // $\Pr(\{\mathrm{H}\}) = p$, $\Pr(\{\mathrm{T}\}) =: q = 1 - p$
    if result $=$ H **then** // If heads
      $j \leftarrow$ `rand(unpaired)`
    else // If tails
      $j \leftarrow$ `randWithWeight(unpaired,` $\{d_k^{in}\}_{k \in \text{unpaired}}$`)` [a]
    $\mathcal{E} \leftarrow \mathcal{E} + \{i \to j\}$ // Add the edge $i \to t$ to $\mathcal{G}$
    unpaired $\leftarrow$ unpaired $\setminus \{j\}$ // j is now paired with i
return $\mathcal{G}$

[a] In this case, we choose one of the unpaired nodes in **proportion** to their incoming edges.

Remark. Node 1 always has out-degree 0.

6.7 Analysis of Preferential Attachment Directed Graph

Now, let us analyze the situation such that if $d_i^{\text{out}} = 1$ for all $i = 2, 3, \ldots$ with $t = 2, 3, \ldots, V$. Define

$$X_i(t) \coloneqq \#\text{ of incoming edges for node } i \text{ after node } t \text{ is added to the graph.}$$

We then see

$$\sum_{i=1}^{V} X_i(t) = t - 1 = \sum_{i=1}^{t-1} X_i(t)$$

since $X_i(t) = 0$ for all i such that $t \le i \le V$. Look at the change of $X_i(t)$ with respect to $X_i(t-1)$, we further have

$$X_j(t) = \begin{cases} X_j(t-1) = 0, & \text{if } t \le j \le V; \\ X_j(t-1), & \text{w.p. } p\dfrac{t-2}{t-1} + q\dfrac{t-1-X_j(t-1)}{t-1}; \\ X_j(t-1)+1 & \text{w.p. } p\dfrac{1}{t-1} + q\dfrac{X_j(t-1)}{t-1}. \end{cases}$$

Remark. This is a vector valued random process.

In order to simply the analysis, we take a look at the expectation. We have $x_j(t) = \mathbb{E}[X_j(t)]$, which is just

$$x_j(t) = \begin{cases} x_j(t-1) = 0, & \text{if } t \le j \le V; \\ x_j(t-1) + p\dfrac{1}{t-1} + q\dfrac{x_j(t-1)}{t-1}, & \text{otherwise.} \end{cases}$$

Then, we have

$$\underbrace{x_j(t) - x_j(t-1)}_{\text{discrete derivative}} = \begin{cases} 0, & \text{if } t \le j \le V; \\ \dfrac{p}{t-1} + q\dfrac{x_j(t-1)}{t-1}, & \text{otherwise.} \end{cases}$$

Let $V \to \infty$ and $\widetilde{x}_i(t) = \frac{x_i(t)}{V} \cong \frac{X_i(t)}{V}$ will be well-approximated by a differential equation

$$\frac{\mathrm{d}\widetilde{x}_j(t)}{\mathrm{d}t} = \frac{p}{t} + q\frac{\widetilde{x}_j(t)}{t} = \frac{p + q\widetilde{x}_j(t)}{t}.$$

Rearranging, we have

$$\frac{1}{p + q\widetilde{x}_j(t)} \frac{\mathrm{d}\widetilde{x}_j(t)}{\mathrm{d}t} = \frac{1}{t}.$$

Then since

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\ln(p + q\widetilde{x}_j(t))\right) = \frac{1}{p + q\widetilde{x}_j(t)} \cdot q\frac{\mathrm{d}\widetilde{x}_j(t)}{\mathrm{d}t},$$

hence we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\ln(p + q\widetilde{x}_j(t))\right) = \frac{q}{t},$$

In all, we have

$$\ln(p + q\widetilde{x}_j(t)) = q\ln(t) + qc.$$

Take exponential on both sides, we have $p + q\widetilde{x}_j(t) = t^q e^{cq}$. Solving for $\widetilde{x}_j$, we see that

$$\widetilde{x}_j(t) = \frac{t^q e^{cq} - p}{q}.$$

We now need the initial value. In this case, we have $x_j(t) = 0$, $\forall\, t \le j \le V$, which implies $\widetilde{x}_j(j) = 0$, and hence

$$\begin{aligned}
\widetilde{x}_j(j) = 0 &\Leftrightarrow j^q e^{cq} = p \\
&\Rightarrow e^{cq} = \frac{p}{j^q} \\
&\Rightarrow \widetilde{x}_j(t) = \frac{p}{q}\left(\left(\frac{t}{j}\right)^q - 1\right) \forall\, t \ge j.
\end{aligned}$$

From this, we see that the fraction of nodes with degree greater than k is

$$\frac{1}{\left(1 + \frac{q}{p}\frac{k}{V}\right)^{\frac{1}{q}}} \cong \frac{c}{k^\alpha}$$

where $\alpha = \frac{1}{q} = \frac{1}{1-p}$.

Note (Power-tail law). This is what we called *power law tail*.

Remark. This process is biasing towards incoming degrees.

Note (Formal Analysis). A more formal setup is the following. Let $\{X_j(t)\}_{j=1,\dots,V}$ be a Markov chain. Considering two aggregations:

1. For $i = 0, 1, \dots$, we determine the fraction of nodes with in-degree at least i at time t. Then

$$y_i(t) = \frac{1}{V}\sum_{k=1}^{V}\sum_{j=1}^{V} \mathbb{1}_{\{X_j(t)=k\}} = \frac{1}{V}\sum_{j=1}^{V} \mathbb{1}_{\{X_j(t)\ge i\}}.$$

2. For $s \in [0,1]$, define $t = \lfloor sV \rfloor$. Now, for $i = 0, 1, \dots$, set $\widetilde{y}_t^V(s) = y_i(\lfloor sV \rfloor)$. Then the formal comparison of $\widetilde{y}_i^V(s)$ can be made with an appropriate O.D.E. solution $\{y_i(s)\}_{i=0,1,\dots}$ for $s \in [0,1]$. This is the formal mean-field analysis that is not in the scope of the course. Hence, we give a flavor of the results using a hand-wavy argument.

Chapter 7

Game Theory

Lecture 18: Game Theory

We now turn to a new topic, game theory. The goal is to analyze situations with rational agents who want to maximize their won rewards under a particular structured game. 8 Nov. 12:30

7.1 Game

We restrict a game with the following structure.

Definition 7.1.1 (Game)**.** A *game* has players, and each player has a set of strategies and a reward function.

> **Definition 7.1.2** (Player)**.** *Players*, also called *agents*, are the participants of the game.

> **Definition 7.1.3** (Strategy)**.** *Strategies*, also called *actions*, are a set of choices a player can perform in a game.

> **Definition 7.1.4** (Reward)**.** *Rewards*, also called *utilities* or *payoffs*, is the return for each player based on the played strategies of all players (including self).

Note (Assumptions of a game)**.** We make some assumptions of a game.

- There are (**finite**, infinite) number of players.
- Each player has a (**finite**, infinite) set of strategies to choose from, homogeneous or inhomogeneous

Besides the assumptions of game structure, we also have some assumptions for the player.

Note (Assumptions of players)**.** We make some assumptions of players.

- Common knowledge: All agents know all other players' strategy set and payoffs. Basically means each player knows everything about the structure of the game.
- Rationality: All agents are fully *rational* and self utility maximizer.

Remark. As a result of common knowledge, everyone knows that all agents are rational.

By above two assumptions, players succeed in selecting optimal strategies, and our goal is to understand these optimal strategies.

7.2 Normal Form

For simplicity, we also assume the payoffs are finite, but this is nearly all the case, so we just assumed it.

Definition 7.2.1 (One-shot game)**.** A game is a *one-shot Game* if players will simultaneously and independently choose their actions, and they do so only once in this game.

We note that contrarily, there are games called dynamic game.

Definition 7.2.2 (Dynamic game)**.** A game is a *dynamic game* if actions can be played sequentially over time for players in the game.

Now, we see some examples which are all one-shot game and is in normal form, i.e., can be described in a matrix.

Definition 7.2.3 (Payoff matrix)**.** If we tabulate all combinations of players' strategies and the corresponding payoffs in a form of matrix, then we call the matrix the *payoff matrix*. We'll soon see lots of examples.

Remark. A game is in normal form simply means if we can write out its corresponding payoff matrix. But we'll see that describing a game in such a way will lose some information.

Before we see the examples, we make the following notions about characterizing strategy.

Definition 7.2.4 (Dominant strategy)**.** The *(strictly) dominant strategy* is the optimal response to every others' strategy of the other players in the sense that the payoffs of this strategy is strictly better in any cases.

Definition 7.2.5 (Weakly dominant strategy)**.** The *weakly dominant strategy* is the optimal response to every others' strategy of the other players in the sense that the payoffs of this strategy is better in any cases.

Remark. Comparison between strictly and weakly dominant strategy:

- Strictly dominant strategy: The payoff is strictly higher in all cases.
- Weakly dominant strategy: This strategy is always in the best action set. But in some cases, some other strategies will give you the same payoff.

Example (Prisoner's dilemma)**.** Assume that two people are taken for investigation of a crime. They're interrogated simultaneously and in different rooms, so they can't communicate. Each of them is offered the choice to confess or not.

- Two players.
- Actions set: $\{\text{confess(C)}, \text{not confess(NC)}\}$.
- Payoffs: There are four possibilities:
  - If you confess and partner doesn't, then you are released and partner gets 10 years in jail.
  - If you don't confess and partner confess, then you'll get 10 years and partner released.
  - If both confess, both get 4 years.
  - If both do not confess, both get charges on minor crime, which will let them get 1 year.

We can then define so-called payoff matrix,

		Player 2	
		NC	C
Player 1	NC	$(-1,-1)$	$(-10,0)$
	C	$(0,-10)$	$(-4,-4)$

Problem 7.2.1. How will you react?

Answer. We first see what action should suspect 1 plays based on rational analysis. Suspect 1's reasoning should be as follows:

- If suspect 2 confess $\to$ since $-4 > -10 \Rightarrow$ the best action is to confess.
- If suspect 2 don't confess $\to$ since $0 > -1 \Rightarrow$ the best action is to confess.

Since this problem is symmetric, namely the strategy for both suspects are the same, so both of them will confess, leading to an equilibrium. ⊛

Remark. We see that

- No hidden payoffs other than what we have in payoff matrix.
- Both not confessing is better as a pair (in the sense that if we sum the payoffs). We see that utilize sum is not maximized, which means this action is not *efficient*.
- Confessing is the best option of the other suspect's choice. In this case, confessing is a dominant strategy.

Example (Golden ball)**.** Two players have two options, one is split, and another is steal. And there are x amounts of money in total. If both players choose split, then they can both get half of the money; if one chooses split and another chooses steal, the player chooses steal can get all the money while another player get nothing. If both choose steal, then they both get nothing.

- Two players.
- Actions set: {split, steal}.
- Payoffs: Assume the total rewards is x.
  1. (split, split): $\frac{x}{2}$ for both.
  2. (split, steal): player who plays steal gets x, steal gets 0.
  3. (steal, steal): 0 for both.

  The payoff matrix is

		Player 2	
		split	steal
Player 1	split	$(x/2, x/2)$	$(0, x)$
	steal	$(x, 0)$	$(0, 0)$

The player 1's reasoning should be as follows:

- If player 2 chooses to split $\to$ since $x > \frac{x}{2}$, the best action is to steal.
- If player 2 chooses to steal $\to$ since $0 = 0$, the best action is *undetermined*.

We see that stealing is a weakly dominant strategy since it does not strictly outperform in every case.

Example (Prisoner's dilemma ver. 2)**.** Consider again the prisoner's dilemma, but with different payoff matrix.

		Player 2	
		NC	C
Player 1	NC	$(-1,-1)$	$(-3,-2)$
	C	$(-2,-3)$	$(-4,-4)$

With the same analysis, we see that the equilibrium is not confessing, namely (NC, NC) is a dominant strategy equibrilium, furthermore, it's a strictly dominant strategy.

Example (Two firms)**.** There are two firms competing with each other. They both have options to produce lower-priced products or upscale product. And we simply assume that their goal is to maximize the market share. We further assume that in the whole market, there are 60% of population will only buy low-priced products, and other 40% of population will only buy upscale products.

- Two firms.
- Actions set: {low-priced(Lp), upscales} (Uc)
- Payoffs:

		Firm 2	
		Lp	Uc
Firm 1	Lp	$(0.48, 0.12)$	$(0.6, 0.4)$
	Uc	$(0.4, 0.6)$	$(0.32, 0.08)$

We see that firm 1's reasoning should be as follows:

- Lp is a strictly dominant strategy.

We also see that firm 2's reasoning should be as follows:

- If firm 2 choose Lp $\rightarrow$ Uc
- If firm 2 choose Uc $\rightarrow$ Lp

We see that with common knowledge and rationality assumptions, firm 2 will assume that firm 2 will play its dominant strategy. As a result, firm 2 will play Uc, hence, (Lp, Uc) is the *equibrilium point*, and it's also *efficient*.

Example (Two firms ver. 2)**.** Consider the two firms game, but with three actions. The payoff matrix now becomes

		Firm 2		
		A	B	C
Firm 1	A	$(4,4)$	$(0,2)$	$(0,2)$
	B	$(0,0)$	$(1,1)$	$(0,2)$
	C	$(0,0)$	$(0,2)$	$(1,1)$

To see the best response, we see that

- For firm 1:
  - If firm 2 play $A \rightarrow$ pick A.

  - If firm 2 play $B \to$ pick B.
  - If firm 2 play $C \to$ pick C.
- For firm 2:
  - If firm 1 play $A \to$ pick A.
  - If firm 1 play $B \to$ pick C.
  - If firm 1 play $C \to$ pick B.

We see that there are no dominant strategy. But (A, A) is the *equilibrium point*, which we will call it the *Nash equilibrium*.

Note. One may notice that we are using normal form to describe one-shot games. Contrarily, we will use so-called *extensive form* to describe a dynamic game.

7.3 Nash Equilibrium

We now have seen some examples of equilibrium, and we can now formalize it.

Definition 7.3.1 (Nash equilibrium)**.** A set of players played actions is called a *Nash Equilibrium* when each player doesn't have an incitement to **unilaterally deviate**.

We further make two more useful definitions we'll later use. These notions generalize the notion of strategy.

Definition 7.3.2 (Pure strategy)**.** We say a strategy is *pure* if the player will play exactly one strategy deterministically.

Definition 7.3.3 (Mixed strategy)**.** We say a strategy is *mixed* if the player will play strategies with some probability.

Intuition. Basically Definition 7.1.3 is the same as Definition 7.3.2, and Definition 7.3.3 says that the player will play with some randomness.

We then define a game structure mathematically in the following way.

Definition 7.3.4 (Mathematical game)**.** Denote the set of player as $\mathcal{I}$, with the number of the player being $I \coloneqq |\mathcal{I}|$. Assume that each player i has a finite set $\mathcal{S}_i$ of actions to choose from with size $|\mathcal{S}_i| < \infty$. Then, we can define a strategy vector s, which denotes the action chosen by all players such that

$$s \coloneqq (s_1, \dots, s_I)$$

with dimension being I. We also define the vector of opponents' strategy s_{-i} such that

$$s_{-i} \coloneqq (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_I)$$

with dimension being $I - 1$ for player i. Finally, the utility function for player i is

$$u_i \coloneqq \prod_{j=1}^{I} \mathcal{S}_j \to \mathbb{R},$$

where $u_i(s) = u_i(s_1, \dots, s_I) = u_i(s_i, s_{-i})$.

Definition 7.3.5 (Best response)**.** Given the strategies of all other players, the subset of s_i that

maximize the payoff of player i is called the *best response*, or *best response correspondence*, denotes as $\mathrm{BR}_i(s_{-i})$.

Intuition. $\mathrm{BR}_i(s_{-i})$ can be though as follows: For player i, given every other players' strategies they will play, find the optimal response. And since there may be several strategies can achieve the optimal reward, so it's should be a set.

Remark. Note that mathematically, $\mathrm{BR}_i(s_{-i}) \subseteq \mathcal{S}_i$ such that

$$\mathrm{BR}_i(s_{-i}) = \arg\max_{s_i \in \mathcal{S}_i} u_i(s_i, s_{-i}).$$

Lecture 19: Nash Equilibrium

Besides the rationality assumptions, we also have the following. 10 Nov. 12:30

Note (Bounded rationality). The *bounded rationality* assumption says that players can only have some finite levels of rationality, i.e., not fully rational.

As previously seen. Let's review notions we introduced before.

- $\mathcal{I}$: the set of players.
- I: $|\mathcal{I}|$, the number of players.
- $\mathcal{S}_i$: player i choose strategy s_i from $\mathcal{S}_i$ such that $s_i \in \mathcal{S}_i$ with $S_i = |\mathcal{S}_i|$.
- s: $(s_1, s_2, \ldots, s_I)$, strategy chosen by all players.
- Player $i \in \mathcal{I}$, then $-i \coloneqq \mathcal{I} \setminus \{i\}$: Everyone else but i.

  Example.

  $$\begin{aligned} i = 1: &\quad -1 = \{2, 3, \ldots, I\}, \\ i = 2: &\quad -2 = \{1, 3, \ldots, I\}, \\ &\vdots \end{aligned}$$

  since

  $$s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_I).$$

- Best-response correspondence (set): For player i, give the strategies of $-i$ (s_{-i} is given), find the subset of strategies in $\mathcal{S}_i$ that give player i the highest payoff.
- Utilities: $\forall i \in \mathcal{I}$,

  $$u_i \colon \prod_{j=1}^{I} \mathcal{S}_j \to \mathbb{R}.$$

  For every vector $s = (s_1, s_2, \ldots, s_I) \in \prod_{j \in \mathcal{I}} \mathcal{S}_j$, $u_i(s)$ is a real number.

Remark. We see that s_i, s_{-i} form the entire vector of strategies every player plays, so $u_i(s_i, s_{-i})$ is **well-defined**. Moreover,

$$\max_{s_i \in \mathcal{S}_i} u_i(s_i, s_{-i})$$

finds all strategies that attain the maximum rewards.

As previously seen. We now view the Golden Balls game with what we just defined. The payoff matrix is

		Player 2	
		split	steal
Player 1	split	$(x/2, x/2)$	$(0, x)$
	steal	$(x, 0)$	$(0, 0)$

For player 1, given

1. $s_{-1} =$ steal:
   - $u_1(\text{split}, \underbrace{\text{steal}}_{s_{-1}}) = 0$
   - $u_1(\text{steal}, \underbrace{\text{steal}}_{s_{-1}}) = 0$

   So
   $$\arg\max_{s_1 \in \mathcal{S}_1} u_1(s_1, \text{steal}) = \{\text{split}, \text{steal}\},$$
   which is our best response correspondence.
2. $s_{-1} =$ split:
   - $u_1(\text{split}, \underbrace{\text{split}}_{s_{-1}}) = \frac{x}{2}$
   - $u_1(\text{steal}, \underbrace{\text{split}}_{s_{-1}}) = x$

   So
   $$\arg\max_{s_1 \in \mathcal{S}_1} u_1(s_1, \text{split}) = \{\text{steal}\},$$
   which is our best response correspondence, and moreover, since it's unique and strictly greater than any other choices, so we call it the strict best response.

Definition 7.3.6 (Strict best response). $\overline{s}_i$ is called a *strict best response* to s_{-i} if

$$u_i(\overline{s}_i, s_{-i}) > u_i(s_i, s_{-i})$$

for every $s_i \in \mathcal{S}_i \setminus \{\overline{s}_i\}$.

Note. Obviously, a strict best response will be unique. Also, note that Definition 7.3.6 is just Definition 7.2.4.

Remark. With the introduced notations, we can reformulate Definition 7.2.4 and Definition 7.2.5 as follows. s_i is *strictly dominant* if it is the strict best response for all s_{-i}, while s_i is *weakly dominant* if it is one of the best response for all s_{-i}.

Remark. We see that

- In the Golden balls example, where steal is a weakly dominant strategy.
- Rationality plus common knowledge implies that if a player with dominant strategies, then the player will play them.

We can also reformulate Nash equilibrium with the introduced notions.

As previously seen (Nash equilibrium). Let $s^* = (s_1^*, s_2^*, \dots, s_I^*)$ be the **strategy profile**, which is a pure strategy Nash equilibrium if

$$s_i^* \in \mathrm{BR}(s_{-i}^*)$$

for all $i \in \mathcal{I}$.

Note. We see that

- Every agent is playing a BR to the others.
- Utility is the incentive for agents. Setting everyone else at s_{-i}^*, there is no incentive for player i to deviate (not necessary to change action). This implies there are no *unilateral dominants* are possible.

$$\forall_{i \in \mathcal{I}} \; \forall_{s_i \in \mathcal{S}_i} \; u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*).$$

As previously seen. There are two different types of strategies:

- Pure strategy: Fixing an action for every player.
- Mixed strategy: A randomized action is played by every player.

Remark. We see that

1. Pure strategy Nash equilibrium needs not exist for every game.
2. Even if they exist, they need not be unique.
3. If there are multiple equilibrium, then which one gets played is a tough question and usually involves external quantities (outside information).

7.3.1 Pure Strategy Nash Equilibrium

We first introduce the concept of coordination games.

Definition 7.3.7 (Coordination game). *Coordination games* are games such that players get a higher payoff by working together (taking the same action).

Example (Shaking hands). There are two men want to shake their hands. They can either position their hand up (U) or down (D). The payoff matrix is

		Player 2	
		U	D
Player 1	U	$(1,1)$	$(0,0)$
	D	$(0,0)$	$(1,1)$

We see that (U,U) and (D,D) are pure strategy Nash equilibrium.

Example (Battle of the sexes). A man and a woman are going to a date. The man prefer to see football (F) while the woman prefer to go to theater (T). If their opinion are not equal, then they can't go to anywhere, so their fulfillment (payoff) will both be 0. The payoff matrix is

		Woman	
		T	F
Man	T	$(1,2)$	$(0,0)$
	F	$(0,0)$	$(2,1)$

We see that (T,T) and (F,F) are pure strategy Nash equilibrium.

Example (Stay hunt). The payoff matrix is

		Player 1	
		S	H
Player 2	S	$(4,4)$	$(0,3)$
	H	$(3,0)$	$(3,3)$

We see that (S,S) and (H,H) are pure strategy Nash equilibrium.

We also have so-called Anti-Coordination Games.

Definition 7.3.8 (Anti-coordination game). A game is an *anti-coordination game* if one player prefers to coordinate, and the other does not.

And before we see examples about anti-coordination game, we introduce two types of games.

Definition 7.3.9 (Attack-defense game). An *attack-defense game* can be simply described as follows. There are two players, one is attacker and another is defender. Attacker has two actions, they are attack with strategy A or B, respectively. Correspondingly, defender also has two actions to choose from, namely defend against A or defend against B. Clearly, if defender correctly defend attacker's attack in terms of strategy attack chooses, then defender gets a higher payoff, otherwise the attacker gets a higher payoff.

Definition 7.3.10 (Zero-sum game). A *zero-sum game* is a game such that the sum of rewards in each outcome is always zero.

Example (Rock, paper, scissors). This is an example that a game without pure Nash equilibrium. The payoff matrix is

		Player 2		
		Rock	Paper	Scissors
Player 1	Rock	$(0,0)$	$(-1,\underline{+1})$	$(\underline{+1},-1)$
	Paper	$(\underline{+1},-1)$	$(0,0)$	$(-1,\underline{+1})$
	Scissors	$(-1,\underline{+1})$	$(\underline{+1},-1)$	$(0,0)$

And it's clear that no matter what we fixed a strategy for one player first, another player will have incentive to deviate right after another player changes his strategy.

Example (Hawk-Dove). Two neighboring countries are likely going to fight. If one choose Hawk (H), then that country will fight; otherwise if choosing Dove (D), then that country will not going

to fight and will not going to fight back either.

		Country 2	
		H	D
Country 1	H	$(0,0)$	$(5,1)$
	D	$(1,5)$	$(3,3)$

Problem 7.3.1. Is (D, D) a Nash Equilibrium?

Answer. No. Since if player 2 is playing D, we see that $u_1(H, D) = 5$, $u_1(D, D) = 3$. Then player 1 has a deviation to get a higher payoff. ⊛

Problem 7.3.2. Is (H, D) a Nash Equilibrium?

Answer. True. Since

- Player 1: H is the best response.
- Player 2: Player 1 is H. Then $u_2(H, H) = 0$, and $u_2(H, D) = 1$.

We see that neither player 1 nor player 2 has an incentive to deviate. So (D, H) (also (H, D)) is a Nash Equilibrium. ⊛

Example (Matching Pennies)**.** This is a simple example for attack-defense game and also a zero-sum game. Suppose there are two people each hold a penny and simultaneously choose whether to show heads (H) or tails (T) on their penny. Player 1 loses his penny to player 2 if they match; player 1 wins player 2's penny if they don't match.

		Player 2	
		H	T
Player 1	H	$(-1,1)$	$(1,-1)$
	T	$(1,-1)$	$(-1,1)$

We see that there are no pure strategy Nash equilibrium. So we have no idea what strategy each player will play.

Remark. This is also a zero-sum game.

Remark. From matching pennies games example, we see that there are no pure strategy Nash Equilibrium. This is due to the fact that all player will play their strategy in a deterministic way, if the best response exists. And if not, then we have no idea what they will play in this case, which is hard to analyze. Now if we allow some randomness, then by John Nash, he proved that with such a setup, the Nash Equilibrium always exists (Theorem 7.3.1).

7.3.2 Mixed Strategy Nash Equilibrium

Follows the intuition of matching pennies game, we now go in-depth and analyze mixed strategy. Essentially, we are just randomizing one's strategies such that the randomization is done independently. This will bring the game structure with much more interesting properties.

As previously seen. The reason why we call it mixed strategies is initially, we only have distinct strategy to choose from. But now, with probability, we have some kind of **mixed** strategy between

choices. In matching pennies game example, we have strategies which mixes between H and T.

Example. For player i, denote $\overrightarrow{P}_i$ as a probability distribution on $\mathcal{S}_i$ (PMF) such that

$$\{s_{i1}, s_{i2}, s_{i3}, s_{i4}\} = \mathcal{S}_i,$$

and $S_i = 4$. More specifically, $\overrightarrow{P}_i$ looks like

$$(P_{i1}, P_{i2}, P_{i3}, P_{i4})$$

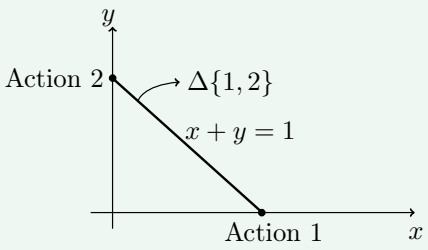
with all entries non-negative and their sum is 1. There are lots of examples for such P_i. For example,

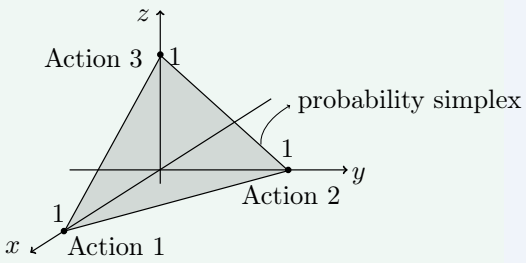
- Uniformly mixed: $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$.
- Pure strategy: $(1, 0, 0, 0)$.

We can then consider so-called probability simplex, defined as follows.

Definition 7.3.11 (Probability simplex)**.** The *probability simplex*, denotes as $\Delta(\mathcal{S}_i)$, is equal to

$$\left\{ (P_{i1}, P_{i2}, \ldots, P_{iS_i}) : P_{ij} \geq 0 \wedge \sum_{j=1}^{S_i} P_{ij} = 1 \quad \forall j = 1, \ldots, S_i \right\}.$$





Intuition. This is just a set of probability distributions over $\mathcal{S}_i$, but with a good geometry representation.

Players' actions are enhanced to picking probability distribution (adding uncertainty to your choice).

Problem 7.3.3. How to evaluate payoff?

Answer. Let

$$u_i(\overrightarrow{P}_i, \overrightarrow{P}_{-i}) \mathbb{E}_{\overrightarrow{P}_1 \times \overrightarrow{P}_2 \times \cdots \times \overrightarrow{P}_I} \left[u_i \left(\overline{S}_1, \overline{S}_2, \ldots, \overline{S}_I \right) \right],$$

such that $\overline{S}_i$ is a random variable that takes values in $\mathcal{S}_i$ with distribution $\overrightarrow{P}_i$, and they are mutually independent. This is further equal to

$$\sum_{j_1 \in \mathcal{S}_1} \sum_{j_2 \in \mathcal{S}_2} \cdots \sum_{j_I \in \mathcal{S}_I} P_{1j_1} \times P_{2j_2} \times \cdots \times P_{Ij_I} \times u_i \left(\overline{S}_{1j_1}, \overline{S}_{2j_2}, \ldots, \overline{S}_{Ij_I} \right).$$

⊛

Example (Matching Pennies revisit)**.** Recall the matching pennies game.

		Player 2 H	Player 2 T
Player 1	H	$(1,-1)$	$(-1,1)$
	T	$(-1,1)$	$(1,-1)$

with $P_1 = (\frac{3}{4}, \frac{1}{4})$ and $P_2 = (\frac{1}{3}, \frac{2}{3})$. Then

$$\begin{aligned} u_1(P_1, P_2) &= \frac{3}{4} \times \frac{1}{3} u_1(H,H) + \frac{3}{4} \times \frac{2}{3} u_1(H,T) + \frac{1}{3} \times \frac{1}{3} u_1(T,H) + \frac{1}{4} \times \frac{2}{3} u_1(T,T) \\ &= \frac{3}{12} - \frac{6}{12} - \frac{1}{12} + \frac{2}{12} \\ &= -\frac{1}{6}. \end{aligned}$$

Also, $u_2(P_1, P_2)$ is $\frac{1}{6}$ from the zero-sum setting.

We can now combine the probability set up with the definition of the best response, which gives us

$$\mathrm{BR}_i(\overrightarrow{P}_{-i}) = \left\{ \overrightarrow{P}_i \in \Delta(\mathcal{S}_i) \colon u_i(\overrightarrow{P}_i, \overrightarrow{P}_{-i}) = \max_{\hat{P}_i \in \Delta(\mathcal{S}_i)} u_i(\hat{P}_i, \overrightarrow{P}_{-i}) \right\}.$$

Also, for Nash Equilibrium, let P^* be the same probability distribution for all $i \in \mathcal{I}$, then

$$P_i^* \in \mathrm{BR}_i(P_{-i}^*) \Leftrightarrow \forall_{P_i \in \Delta(\mathcal{S}_i)} u_i(P_i^*, P_{-i}^*) \geq u_i(P_i, P_{-i}^*).$$

Theorem 7.3.1 (Nash's existence theorem)**.** Every finite game (finite number of actions) has at least one mixed strategy Nash Equilibrium.

Proof. A nice short proof can be found here. It's essentially proved by Brouwer fixed-point theorem. ■

Lecture 20: Mixed strategy

15 Nov. 12:30

As previously seen (Matching pennies revisit)**.** Recall the matching pennies game. Assume now that player 2 plays $(\frac{1}{2}, \frac{1}{2})$. Now, if player 1 plays

- $(1,0)$: Utility for player 1 is $u_1(H, P_2^*) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot -1 = 0$.
- $(0,1)$: Utility for player 1 is $u_1(T, P_2^*) = \frac{1}{2} \cdot -1 + \frac{1}{2} \cdot 1 = 0$.

Hence, the utility of player 1 is just

$$u_1(P_1, P_2^*) = P_1(H) u_1(H, P_2^*) + P_1(T) u_1(T, P_2^*) = 0.$$

Therefore, any mixed strategy is the best response, so we just use $(\frac{1}{2}, \frac{1}{2})$ for player 1, and the same is true for player 2. In all, we have

$$P_1^* \coloneqq \left(\frac{1}{2}, \frac{1}{2}\right), \quad P_2^* \coloneqq \left(\frac{1}{2}, \frac{1}{2}\right)$$

as the Nash Equilibrium.

Problem 7.3.4. What if player 2 was laying $(\frac{2}{3}, \frac{1}{3})$?

Answer. The best response for player 1 is to always play H. ⊛

Work this out

Problem 7.3.5. The mixed strategies are everywhere in real-life. But how can we understand mixed strategies?

Answer. We see that

- Actions are distributions or beliefs.
- In real life, people do randomize-serving in tennis, bluffing in cards game, openings in chess, etc.
- Variations in the population and lack of knowledge of how the other person will play.

⊛

Example (Football). Let player 1 be offense-side, and player 2 be defense-side. They both have two options, one is run (or to defense against run), denotes by R, another is pass (or defense against pass), denotes by P.

		Player 2	
		R	P
Player 1	R	$(0,0)$	$(5,-5)$
	P	$(10,-10)$	$(0,0)$

we then see that there is a Nash Equilibrium such that player 1 plays $(\frac{2}{3},\frac{1}{3})$, while player 2 plays $(\frac{1}{3},\frac{2}{3})$.

7.3.3 Principle of Indifference

For player i, let $P_i \in \Delta(\mathcal{S}_i)$. Now, let the support of P_i to be the actions with strictly positive probability, namely

$$\{s_i \in \mathcal{S}_i \colon P_i(s_i) > 0\} \subseteq \mathcal{S}_i.$$

Example. For example, let $\{a,b,c\} = \mathcal{S}_i$. Then

- $P_i = (1,0,0)$: the support is $\{a\}$
- $P_i = (\frac{1}{6}, \frac{5}{6} - \frac{1}{10^6}, \frac{1}{10^6})$: the support is $\{a,b,c\}$
- $P_i = (0, \frac{1}{2}, \frac{1}{2})$: the support is $\{b,c\}$

Note. In the second case, we call this a fully mixed.

Definition 7.3.12 (Fully mixed). A mixed strategy is called *fully mixed* if the support of the strategy vector is on the entire actions set.

Theorem 7.3.2 (Principle of indifference). For a Nash Equilibrium P^*, for every player i, if

$$\forall s_i^j, s_i^k \in \text{support}(P_i^*),$$

we have

$$u_i(s_i^j, P_{-i}^*) = u_i(s_i^k, P_{-i}^*) = u_i(P_i^*, P_{-i}^*).$$

Proof. Let $s_i^j, s_i^k \in \text{support}(P_i^*)$. Then $P_i^*(s_i^j) > 0$ and $P_i^*(s_i^k) > 0$. Now, denotes γ as $\gamma =$

$\min\left(P_i^*(s_i^j), P_i^*(s_i^k)\right) > 0$, and let $\epsilon := [-\gamma, \gamma]$. We can then construct a new mixed strategy $\widetilde{P}_i^\epsilon$ for player i as follows. For s_i^j and s_i^k

$$\widetilde{P}_i^\epsilon(s_i^j) = P_i^*(s_i^j) - \epsilon, \quad \widetilde{P}_i^\epsilon(s_i^k) = P_i^*(s_i^k) + \epsilon,$$

with $\widetilde{P}_i^\epsilon(s_i) = P_i^*(s_i)$ for all $s_i \in \mathcal{S}_i \setminus \left\{s_i^j, s_i^k\right\}$, which means that the other actions remain the same. Since P^* is a Nash Equilibrium, so we have

$$u_i\left(P_i^*, P_i^*\right) = \max_{P_i \in \Delta(\mathcal{S}_i)} u_i(P_i, P_{-i}^*) \geq u_i(\widetilde{P}_i^\epsilon, P_{-i}^*).$$

Furthermore, the left-hand side is

$$u_i\left(P_i^*, P_{-i}^*\right) = P_i^*(s_i^j)u_i(s_i^j, P_{-i}^*) + P_i^*(s_i^k)u_i(s_i^k, P_{-i}^*) + \sum_{s_i \in \mathcal{S}_i \setminus \left\{s_i^j, s_i^*\right\}} P_i^*(s_i)u_i(s_i, P_{-i}^*),$$

and the right-hand side is

$$u_i\left(\widetilde{P}_i^\epsilon, P_{-i}^*\right) = \widetilde{P}_i^\epsilon(s_i^j)u_i(s_i^j, P_{-i}^*) + \widetilde{P}_i^\epsilon(s_i^k)u_i(s_i^k, P_{-i}^*) + \sum_{s_i \in \mathcal{S}_i \setminus \left\{s_i^j, s_i^*\right\}} P_i^*(s_i)u_i(s_i, P_{-i}^*).$$

Then we see that the inequality is

$$P_i^*(s_i^j)u_i(s_i^j, P_{-i}^*) + P_i^*(s_i^k)u_i(s_i^k, P_{-i}^*) \geq \widetilde{P}_i^\epsilon(s_i^j)u_i(s_i^j, P_{-i}^*) + \widetilde{P}_i^\epsilon(s_i^k)u_i(s_i^k, P_{-i}^*)$$

since the summation part is the same. Rearranging, we have

$$\epsilon u_i(s_i^j, P_{-i}^*) \geq \epsilon u_i(s_i^k, P_{-i}^*).$$

Since $\epsilon \in [-\gamma, \gamma]$, we further have

$$u_i(s_i^j, P_{-i}^*) = u_i(s_i^k, P_{-i}^*) = u.$$

Hence,

$$u_i(P_i^*, P_i^*) = \sum_{s_i \in \text{support}(P_i^*)} P_i^*(s_i) \underbrace{u_i(s_i, P_{-i}^*)}_{u} = u.$$

The last equality follows since

$$\sum_{s_i \in \text{support}(P_i^*)} P_i^*(s_i) = 1.$$

■

Intuition. In other words, given the strategies of the opponents, player i is **indifferent** between the strategies in the support(P_i^*) because they yield the same payoff.

Now, we show a general method to find all Nash Equilibrium in a 2 players and 2 actions game. Let the payoff matrix be like

		Player 2	
		L	R
Player 1	U	$(a_{U,L}, b_{U,L})$	$(a_{U,R}, b_{U,R})$
	D	$(a_{D,L}, b_{D,L})$	$(a_{D,R}, b_{D,R})$

Let

- Player 1 plays strategy $(p, 1-p)$ with $p \in [0, 1]$.
- Player 1 plays strategy $(q, 1-q)$ with $q \in [0, 1]$.

We then find all the best response of each player in a parametric form. Says player 1 with $p^*(q)$, while player 2 with $q^*(p)$, where they can be a set. We see that

$$p^*(q) = \arg\max_{\widetilde{p}\in[0,1]} u_1\left((\widetilde{p}, 1-\widetilde{p}), (q, 1-q)\right).$$

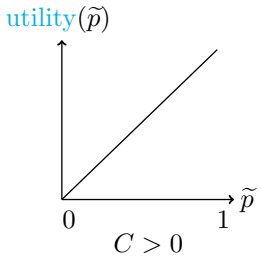
Then

$$\begin{aligned}
&u_1\left((\widetilde{p}, 1-\widetilde{p}), (q, 1-q)\right) \\
=&\widetilde{p}\left(q \cdot a_{U,L} + (1-q)a_{U,R}\right) + (1-\widetilde{p})\left(q \cdot a_{D,L} + (1-q)a_{D,R}\right) \\
=&\widetilde{p}\left([q \cdot a_{U,L} + (1-q)a_{U,R}] - [q \cdot a_{D,L} + (1-q)a_{D,R}]\right) + (q \cdot a_{D,L} + (1-q)a_{D,R})
\end{aligned}$$

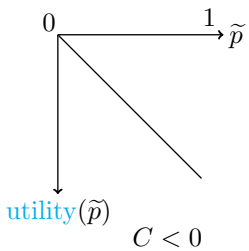
Now, denotes C as

$$C \coloneqq [q \cdot a_{U,L} + (1-q)a_{U,R}] - [q \cdot a_{D,L} + (1-q)a_{D,R}],$$

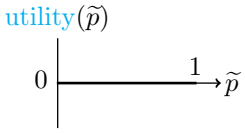
there are three cases for C:



$C > 0$



$C < 0$



$C = 0$

We see that

- $C > 0$: maximizes in $\widetilde{p} = 1$
- $C < 0$: maximizes in $\widetilde{p} = 0$
- $C = 0$: maximizes in $\widetilde{p} = [0, 1]$

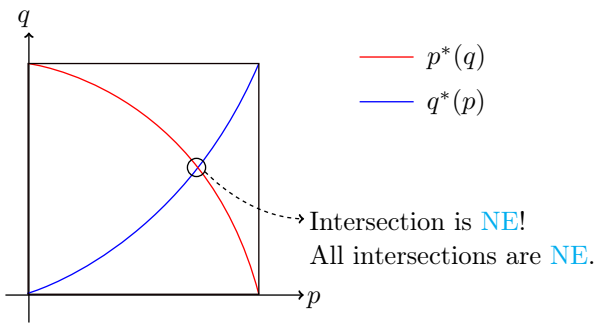
Then we denote

$$\arg\max_{\widetilde{p}\in[0,1]} u_1\left((\widetilde{p}, 1-\widetilde{p}), (q, 1-q)\right) =: p^* \coloneqq \begin{cases} 1, & \text{if } C > 0; \\ 0, & \text{if } C < 0; \\ [0,1], & \text{if } C = 0 (\text{Indifference}). \end{cases}$$

Use the similar calculations to find $q^*(p)$ using the best response. We conclude that

$$(\overline{p}, \overline{q}) \text{ is a Nash Equilibrium} \Leftrightarrow \overline{p} \in p^*(\overline{q}) \wedge \overline{q} \in q^*(\overline{p}),$$

namely they are the best response of each other.

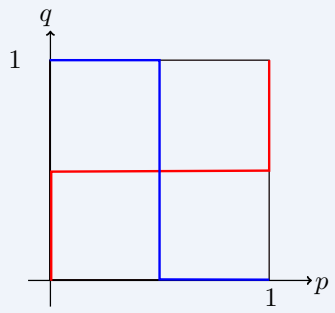


Now we see some examples.

Example (Matching pennies revisit). Coming back to the matching pennies games. We have

$$p^*(q) = \begin{cases} 1, & \text{if } q > \frac{1}{2} \\ [0,1], & \text{if } q = \frac{1}{2} \\ 0, & \text{if } q < \frac{1}{2} \end{cases}, \quad q^*(p) = \begin{cases} 1, & \text{if } p > \frac{1}{2} \\ [0,1], & \text{if } p = \frac{1}{2} \\ 0, & \text{if } p < \frac{1}{2}. \end{cases}$$

After plotting, we see that $(\frac{1}{2}, \frac{1}{2})$ is the only intersection, so it's the only Nash equilibrium.



Example (Football revisit). We also look at the football game example. We see that for $p^*(q)$, we have

- R: utility is $q(0) + (1-q)5 = 5 - 5q$.
- P: utility is $q(10) + (1-q)0 = 10q$.

Hence, the difference is $5 - 5q - 10q = 5 - 15q$, so

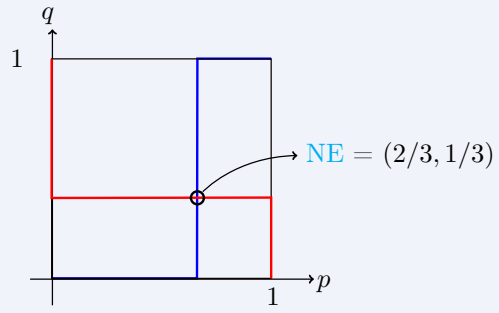
$$5 - 15q \Rightarrow \begin{cases} \text{Difference } > 0, p^*(q) = 1, & 5 > 15q \Rightarrow q < \frac{1}{3}; \\ \text{Difference } < 0, p^*(q) = 0, & 5 < 15q \Rightarrow q > \frac{1}{3}; \\ \text{Difference } = 0, p^*(q) = [0,1], & 5 = 15q \Rightarrow q = \frac{1}{3}. \end{cases}$$

As for $q^*(p)$, we have

- R: utility is $p(0) + (1-p)(-10) = -10 + 10p$.
- P: utility is $p(-5) + (1-p)0 = -5q$.

Hence, the difference is $-10 + 10p - (-5p) = -10 + 15p$, so

$$-10 + 15p \Rightarrow \begin{cases} \text{Difference } > 0, q^*(p) = 1, & 5 > 15q \Rightarrow p > \frac{2}{3}; \\ \text{Difference } < 0, q^*(p) = 0, & 5 < 15q \Rightarrow p < \frac{2}{3}; \\ \text{Difference } = 0, q^*(p) = [0,1], & 5 = 15q \Rightarrow p = \frac{2}{3}. \end{cases}$$



Example (Market entering)**.** We now look at a new game. Consider the Market entering game. Let the payoff matrix being

		Player 2	
		E	D
Player 1	E	$(-1,-1)$	$(3,0)$
	D	$(0,3)$	$(0,0)$

We immediately see that (E,D) and (D,E) are pure Nash equilibrium. And since we always have odd number of Nash equilibrium, hence there are one more mixed strategy Nash equilibrium. By using the principle of indifference, we see that

$$\begin{cases} u_1(E,\overline{q}) &= \overline{q}(-1)+(1-\overline{q})3 = 3-4\overline{q} \\ u_1(D,\overline{q}) &= \overline{q}(0)+(1-\overline{q})0 = 0, \end{cases}$$

so we have

$$u_1(E,\overline{q}) = u_1(D,\overline{q})$$

by the principle of indifference. This implies $\overline{p} = \frac{3}{4}$.

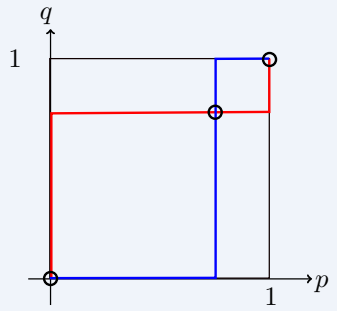


Figure 7.1: There are 3 NE.

Lecture 21: Bayesian Game

We first introduce a new notion. 17 Nov. 12:30

Definition 7.3.13 (Social welfare)**.** . The *social welfare* is the sum of payoffs among all players.

As previously seen. We look back some games and introduce another measure of Nash equilibrium.

- Coordination game. The payoff matrix is defined as

		Player 2	
		S	H
Player 1	S	$(5,5)$	$(0,3)$
	H	$(3,0)$	$(3,3)$

As we can see, there are two Nash equilibrium (S,S) and (H,H) with payoffs $(5,5)$ and $(3,3)$ respectively. If we consider social welfare, one will choose to play (S,S) with social welfare as 10 rather than 6.

- Prisoner's dilemma.

		Player 2	
		NC	C
Player 1	NC	$(-1,-1)$	$(-10,0)$
	C	$(0,-10)$	$(-4,-4)$

We see that (C, C) is the only Nash equilibrium, but (NC, NC) is the social welfare maximization, which is not in the Nash equilibrium.

Remark. We say that a Nash equilibrium *reaches social welfare* if it's a maximizer of the social welfare.

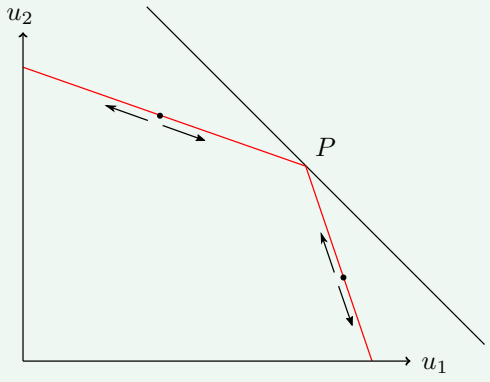
7.3.4 Operating Points

Let $i \in \mathcal{I}$ be the set of players, and P_i be the corresponding distribution of strategy on player i. Then we see the definition of the so-called operating points.

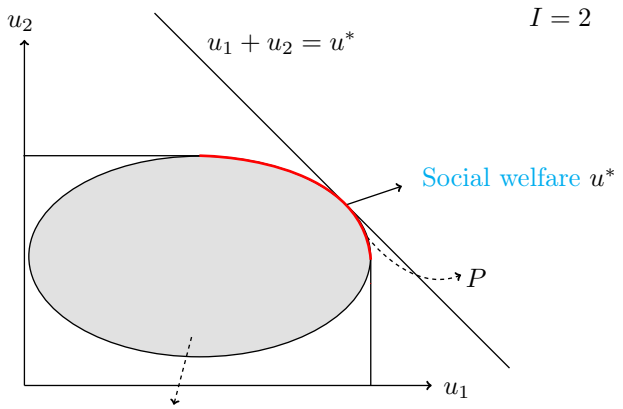
Definition 7.3.14 (Operating point)**.** An *operating point* is a vector of rewards of players such that we cannot increase the payoff of one agent without decreasing the payoff of at least one other agent.

Let P denotes the set of operating points. From here, we have the following criteria for Pareto Optimal.

Definition 7.3.15 (Pareto optimal)**.** A solution is called *Pareto optimal* if no individual or preference criterion can be better off without making at least one individual or preference criterion worse off or without any loss thereof.



$I = 2$



feasible region: All possible payoffs vectors

Note. For either social welfare maximizing or Pareto operating point,

- Nash equilibrium needs not achieve either goal.
- Social welfare maximizing points are always Pareto optimal.

Remark. We can (re)design games such that some nice objective is additive, then social welfare maximization can be achieved, i.e., the goal is that any Nash equilibrium should maximize social welfare.

Until this point, we only study the games have complete and symmetric information, i.e., common knowledge assumption. Now, we can look at some more complicated cases such that we have an incomplete and/or asymmetric information setting.

7.4 Bayesian Game

We now introduce a new type of game called Bayesian game.

Definition 7.4.1 (Bayesian game)**.** A *Bayesian game* is an asymmetric/incomplete games.

Intuition. Since we need to use Bayesian Theorem to analysis Bayesian games, so it gets its name.

Example (Auction)**.** A typical example of a Bayesian game is auctions, which we'll study in depth later.

An important difference is that in Bayesian games, nation is also a player. Further, randomness decides the state of everyone and people may know their state but not of the others.

Note. Randomness here is different from mixed strategies. This happens **at the beginning**.

7.4.1 Basic Setup

We now try to mathematically model a Bayesian game.

Definition 7.4.2 (Methematical Bayesian game)**.** Let $\mathcal{I}$ denotes the set of players, and $I = |\mathcal{I}|$. Let each player i has a type defined as follows.

Definition 7.4.3 (Type). The *type* of a player i, denotes as $\tau_i \in T_i$, is chosen with distribution π_i such that

$$\Pr(\tau_i = t_i) = \pi_i(t_i).$$

And also, we have the set of actions $\mathcal{S}_i$, and the set of feasible actions

$$c_i(t_i) \in \mathcal{S}_i.$$

Furthermore, the utility of player i is

$$u_i(t_1, s_1, t_2, s_2, \ldots, t_I, s_I),$$

where we need types and actions of all players to determine this. Also, we have

$$\vec{t} \coloneqq (t_1, \ldots, t_I), \quad \vec{s} \coloneqq (s_1, \ldots, s_I),$$

with the common notation for $-i$, namely

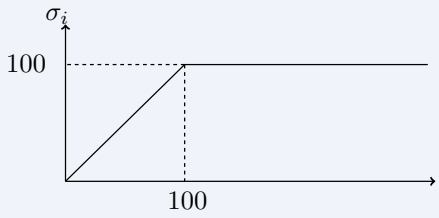
$$\vec{t}_{-i} \coloneqq (t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_I), \quad \vec{s}_{-i} \coloneqq (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_I).$$

We see that player i has to choose a strategy $\sigma_i \colon T_i \to \mathcal{S}_i$ such that

$$\forall t_i \in T_i, \text{ choose } \sigma_i(t_i) \in c_i(t_i) \subseteq \mathcal{S}_i.$$

Intuition. We are simply picking functions as strategies.

Example. Let $T_i = \mathbb{R}_+ = [0, +\infty)$ be non-negative real number. Let $\sigma_i(t_i) = \min(100, t_i)$ and $\mathcal{S}_i = \mathbb{R}_+$.



Then we can have so-called independent types model.

Definition 7.4.4 (Independent types model). An *independent types model* is a Bayesian game such that the types τ_i, $i = 1, \ldots, I$ are a set of mutually independent variables.

Remark. For an independent types model, we have

$$\Pr(\tau_1 = t_1, \ldots, \tau_I = t_I) = \prod_{i=1}^{I} \pi_i(t_i).$$

We see that we can express all games we have previously seen like follows. Let $T_i = \{i\}$, and $c_i(i) = \mathcal{S}_i$ with type being $\pi_i(i) = 1$ such that

$$u_i(1, s_1, 2, s_2, \ldots, I, s_I) = u_i(s_1, s_2, \ldots, s_I).$$

Now we define the notions of incomplete information.

Definition 7.4.5 (Ex-ante). No player has information about anyone's types but only the distribution information where strategy is chosen. Since we don't know the exact payoffs, hence we simply take

the expected value for the payoffs based on the strategy functions:

$$u_i(\sigma_1, \ldots, \sigma_I) = \mathbb{E}_{\tau_1,\ldots,\tau_I}\left[u_i(\tau_1, \sigma_1(\tau_1), \ldots, \tau_I, \sigma_I(\tau_I))\right].$$

Then the Nash equilibrium is $(\sigma_1^*, \ldots, \sigma_I^*)$ such that

$$\underset{i\in\mathcal{I}}{\forall}\ \underset{\sigma_i}{\forall}\ u_i(\sigma_i^*, \sigma_{-i}^*) \geq u_i(\sigma_i, \sigma_{-i}^*).$$

This kind of information mechanism is called *ex-ante*.

Note. The ex-ante information mechanism can be summarized as follows.

1. Given player $i, \ldots, I$.
2. $\sigma_1, \ldots, \sigma_I$ are chosen: Each player picks strategy σ_i.
3. $\tau_1, \ldots \tau_I$ are realized: Nation chooses types for each player.
4. $\underset{i}{\forall}\ s_i = \sigma_i(\tau_i)$ is chosen: Actions produced, and the game is realized.
5. $u_i(\tau_1, s_1, \ldots, \tau_I, s_I)$, namely the payoff is received.

Definition 7.4.6 (Interim). Players get to know their types since Nation play first, and they can reason based on that. So we see that player i knows τ_i but not τ_{-i}. Then $u_i(t_i, \sigma_i(t_i), \sigma_{-i})$ can be described as

$$\mathbb{E}_{\tau_{-i}}\Big[u_i(\underline{t_i, \sigma_i(t_i)}, \tau_1, \sigma_1(\tau_1), \ldots, \underline{\tau_{i-1}, \sigma_{i-1}(\tau_{i-1}),} \\ \underline{\tau_{i+1}, \sigma_{i+1}(\tau_{i+1})}, \ldots, \tau_I, \sigma_I(\tau_I))\Big].$$

This kind of information mechanism is called *interim*.

Note. The interim information mechanism can be summarized as follows.

1. Given player $i, \ldots, I$.
2. $\tau_1, \ldots \tau_I$ are realized: Nation chooses types for each player.
3. $\sigma_1, \ldots, \sigma_I$ are chosen: Each player picks strategy σ_i.
4. $\underset{i}{\forall}\ s_i = \sigma_i(\tau_i)$ is chosen: Actions produced, and the game is realized.
5. $u_i(\tau_1, s_1, \ldots, \tau_I, s_I)$, namely the payoff is received.

Remark. Think about the best response. Suppose σ_{-i} is fixed. Then for all $s_i \in \mathcal{S}_i$, $u_i(t_i, s_i, \sigma_{-i})$ is defined.

$$\arg\max_{s_i\in\mathcal{S}_i} u_i(t_i, s_i, \sigma_{-i})$$

for every t_i, $\mathrm{BR}_i(\sigma_{-i}) = \overline{\sigma}_i(\sigma_{-i})$. Then we call that $(\sigma_1^*, \ldots, \sigma_I^*)$ is a Bayes interim Nash equilibrium if

$$\underset{i\in\mathcal{I}}{\forall}\ \sigma_i^* \in \overline{\sigma}_i(\sigma_{-i}).$$

Definition 7.4.7 (Ex-post). Every player gets to know everyone's type.

$$u_i(t_1, \sigma_1(t_1), \ldots, t_I, \sigma_I(t_I)).$$

This kind of information mechanism is called *ex-post*.

We will focus on Interim setting and study auctions, which model mechanism well.

Chapter 8

Auctions

Lecture 22: Auctions

29 Nov. 12:30

We now introduce a specific kind of game, auction. Rather than the simple set up we have seen, the auction is more complex and interesting. We'll see how can we extend what we have discussed in game theory into this set up. Firstly, we see some examples.

Example (Auctions)**.** There are mainly four types of auction:

1. First-price auction (Sealed bid): Each player bids an amount for the item (how much the player is willing to pay), and the sellers selects the highest bidder and charges him the price that was bid by him and give the item. Others pay nothing, receive nothing.
2. Dutch auction (Descending price auction): Seller announces a price (high price). If no players raise hands, seller drops price until the first time some players have their hand up. Item sold to the player at that price.
3. Second-price auction (Sealed bid):
   (a) Sort the bids from the highest
   (b) Find the highest bidder
   (c) Give the good to this bidder but charge him/her the second-highest bid.
4. English auction (Ascending price auction):
   (a) Start from price 0
   (b) Keeping increasing price until only one left
   (c) Sell the good to the person remaining and at the price at which everyone else drops out.

Remark. The first two auctions basically implement the same result.

Note (Tie breaking)**.** Some form of auctions may encounter tie issues. Some common way to do ties breaking are

- Random order.
- Choose some fixed order (specified earlier).

Example (Second-price auction)**.** For example, we have

	A	B	C	D
b_i	10	2	3	7

After sorted:

	A	D	C	B
b_i	10	7	3	2

We then sell the good to A with price 7.

Example (English auction). For example, we start with 0, and everyone wants the good. Then we slightly increase the price, until:

1. Price is 2 or $(2+\epsilon)$: A, D and C want the good, B drops out.
2. Price is 3 or $(3+\epsilon)$: A and D want the good, C drops out too.
3. Price is 7 or $(7+\epsilon)$: A want the good, D drops out too.

Then A gets the good with price being 7.

We relate auctions with game as we discussed before. Given the types for player $i \in \mathcal{I}$ being i.i.d. such that

$$t_1, t_2, \ldots, t_I,$$

and also for the valuations such that

$$v_1, v_2, \ldots, v_I.$$

Note. We see that

- For simplicity, without specifying, we assume they are distributed in
$$\text{uniform}([0,1]).$$
- The valuation v_i is taken as the satisfaction in dollar amount from buying the good.

Then, the utility function is determined based on specific forms of auction.

8.1 Order Statistics

For all kinds of auctions we mentioned, we see that to analyze auctions, it'll involve some kind of order of bidding. In the content of auctions, we often refer this as *order statistics*.

Let $X_1, X_2, \ldots, X_n$ be i.i.d. and non-negative, continuous valued random variables. Let $F_X(\cdot)$ be the CDF, and $f_X(\cdot)$ be the probability density function.

Example. Consider

1. $X = \text{uniform}([0,1])$:
$$F_X(x) = \begin{cases} x, & \text{if } x \in [0,1]; \\ \min(x,1) & \text{if } x > 1, \end{cases}$$
with
$$f_X(x) = \begin{cases} 1, & \text{if } x \in [0,1]; \\ 0, & \text{if } x > 1. \end{cases}$$

2. Exp(λ):
$$F_X(x) = 1 - e^{-\lambda x}, \quad x \geq 0,$$
with
$$f_X(x) = \lambda e^{-\lambda x}, \quad x \geq 0.$$

The idea is simple, we first sort $x_1, x_2, \ldots, x_n$ in decreasing order and denotes

$$y_1, y_2, \ldots, y_n$$

such that

$$\begin{aligned} y_1 &= \max\{x_1, \ldots, x_n\} \\ y_2 &= \max\{x_1, \ldots, x_n\} \setminus \{y_1\} \\ y_3 &= \max\{x_1, \ldots, x_n\} \setminus \{y_2, y_3\} \\ &\vdots \end{aligned}$$

Then we have
$$\max\{x_1, \ldots, x_n\} = y_1 \geq y_2 \geq \cdots \geq y_n = \min\{x_1, \ldots, x_n\}.$$

Now, we want to find out $\mathbb{E}[Y_1]$, which is just

$$\mathbb{E}[Y_1] = \int_0^\infty y f_{Y_1}(y)\,\mathrm{d}y$$

where f_{Y_1} is the probability density function of Y_1. Since $f_{Y_1}(y) = \frac{\mathrm{d}}{\mathrm{d}y}(F_{Y_1}(y))$, where

$$\begin{aligned} F_{Y_1}(y) &= \Pr(Y_1 \leq y) = \Pr(\max\{X_1, \ldots, X_n\} \leq y) \\ &= \Pr(X_1 \leq y, X_2 \leq y, \ldots, X_n \leq y) \\ &= \prod_{i=1}^n \Pr(X_i \leq y) = \prod_{i=1}^n F_{X_i}(y) = (F_X(y))^n, \end{aligned}$$

where we have used the properties of the i.i.d. assumption.

Further, since we now know

$$f_{Y_1}(y) = \frac{\mathrm{d}}{\mathrm{d}y}\left((F_X(y))^n\right) = n\,(F_X(y))^{n-1}\,\frac{\mathrm{d}}{\mathrm{d}y}F_X(y) = n\,(F_X(y))^{n-1}\,f_X(y),$$

we have

$$\mathbb{E}[Y_1] = \int_0^\infty ny\,(F_X(y))^{n-1}\,f_X(y)\,\mathrm{d}y.$$

Example. Let $F_X(\cdot) \sim \text{uniform}([0,1])$. Then

$$\mathbb{E}[Y_1] = \int_0^1 nyy^{n-1}\,\mathrm{d}y = n\int_0^1 y^n\,\mathrm{d}y = \left.\frac{ny^{n+1}}{n+1}\right|_0^1 = \frac{n}{n+1}.$$

Hence, $\mathbb{E}[X_1] = \frac{1}{2}$.

Remark. The expected value of the k^{th} element is exactly

$$\frac{k}{n+1}$$

as one can check.

8.2 Second-Price Auction

We first analyze second-price auction, since it's easier to analyze. Recall the interim setting, namely each agent know its valuation but only the distribution of the valuation of others. Now, let

$$\sigma_i \colon [0,1] \to [0,1] \qquad v_i \mapsto \sigma_i(v_i)$$

where σ_i is just the strategy of bidding. Then the utility is

$$\mathbb{E}_{-i}\left[u_i(\underbrace{v_i, \sigma_i(v_i)}_{\text{known for } i}, \underbrace{v_{-i}, \sigma_{-i}(v_{-i})}_{\text{random}})\right].$$

The goal is to choose $\sigma_i(\cdot)$ such that expected utility is maximized for each v_i assuming σ_{-i} is fixed. Then,

$$\begin{aligned} u_i(v_i, \sigma_i(v_i), v_{-i}, \sigma_{-i}(v_{-i})) &= \left(v_i - \max_{j \in I \setminus \{i\}} \sigma_j(v_j)\right) \mathbb{1}_{\{i \text{ wins the auction}\}} \\ &= \left(v_i - \max_{j \in -i} \sigma_j(v_j)\right) \mathbb{1}_{\{\sigma_i(v_i) \geq \sigma_j(v_j) \forall j \in -i\}}. \end{aligned}$$

We now aim to maximize

$$\mathbb{E}_{-i}\left[u_i(v_i, \sigma_i(v_i), v_{-i}, \sigma_{-i}(v_{-i}))\right].$$

8.2.1 Reason of Response

We need to reason how agents will bid strategically.

- If $b_i = \sigma_i(v_i) > v_i$.
  - (a) $b_i < \widetilde{P}_i = \max_{j \in -i} \sigma_j(v_j) = b_j$, then we see that $\max(b_i, \widetilde{P}_i) = \widetilde{P}_i$, so i never wins the auction. Continue to do so even if he reduces bid to v_i.
  - (b) $b_i = \widetilde{P}_i$, then i can win the auction. Since $v_i - \widetilde{P}_i < 0$, and the second-highest bid is $\widetilde{P}_i$. So He better to bid v_i and lost the auction.
  - (c) $b_i > \widetilde{P}_i$, then i definitely wins the auction.
    - $v_i < \widetilde{P}_i$: utility is $v_i - \widetilde{P}_i < 0$, not a viable option.
    - $v_i \geq \widetilde{P}_i$: utility is $v_i - \widetilde{P}_i \geq 0$, but same result would also hold if b_i is exactly v_i.
- If $b_i = \sigma_i(v_i) < v_i$.
  - (a) $\widetilde{P}_i \leq b_i < v_i$. Agent i wins with utility being $v_i - \widetilde{P}_i \geq 0$ and no lose in increasing bid to v_i.
  - (b) $b_i < v_i \leq \widetilde{P}_i$. Don't win the auction. No lose in increasing bid to v_i.
  - (c) $b_i \leq \widetilde{P}_i \leq v_i$. Lose the auction. However, bidding v_i the agent could have won the auction and gotten non-negative utility.

> **Remark** (Incentive compatible)**.** Bidding v_i is a (weakly) dominant strategy for agent i. This further implies that for second-price auction, agents bid their true value. This is essentially *incentive compatibility*, or *truth-telling desirable*.

> **Remark.** We see that second-price auction is truth-telling in dominant strategies. Hence, irrespective of how the other players bid, truthfully bidding is optimal, namely
>
> $$\sigma_i(v_i) = v_i.$$

8.2.2 Revenue of the Auctioneer

Given $v_1, v_2, \ldots, v_I$, defined $\widetilde{v}_i$ as before, namely

$$\max\{v_1, \ldots, v_I\} = \widetilde{v}_I \geq \widetilde{v}_{I-1} \geq \cdots \geq \widetilde{v}_1 = \min\{v_1, \ldots, v_I\}.$$

And since it's a second-price auction, so the revenue is $\widetilde{v}_{I-1}$.

> **Note.** The expected revenue of $\widetilde{v}_{I-1}$ is
>
> $$\mathbb{E}\left[\widetilde{v}_{I-1}\right] = \frac{I-1}{I+1}$$
>
> if we assume uniform valuation.

Hence, we see that the winner gets $\widetilde{v}_I - \widetilde{v}_{I-1}$, and the expected utility is

$$\mathbb{E}\left[\widetilde{v}_I\right] - \mathbb{E}\left[\widetilde{v}_{I-1}\right].$$

In the uniform case, the above further equals to

$$\frac{I}{I+1} - \frac{I-1}{I+1} = \frac{1}{I+1}.$$

With winner and auctioneer together, their utility is

$$\widetilde{v}_I - \widetilde{v}_{I-1} + \widetilde{v}_{I-1} = \widetilde{v}_I.$$

> **Remark.** We see that the good always goes to the highest valuation.

Sums of utilities of all players in auctioneer is social welfare - max value possible of $\widetilde{v}_I$.

> **Remark.** Second-price auction is social welfare maximizing.

Now, if agent i bids v_i and wins, then he gets

$$v_i = \max_{j \in -i} v_j \geq 0,$$

and if lose then get 0. Hence, the expected utility y is always non-negative if you attend the auction, while staying out will get you 0, hence we see that it's better to participate in the auction.

> **Remark** (Voluntary participation)**.** This phenomenon is called *voluntary participation*, or *individual rationality*.

And since the probability of winning is $\frac{1}{I}$, hence the expected utility is

$$\frac{1}{I} \times \frac{1}{I+1} = \frac{1}{I(I+1)}$$

for every agent.

> **Note.** Even if we are using ex-post setting, it's still optimal to bid truthfully in a second-price auction.

8.3 First-Price Auction

We'll see that the first price auction is not as **simple**. There are no dominant strategy equilibrium, only so-called Bayes-Nash equilibrium in interim.

Consider $I = 2$, and v_i choose $b_i = \sigma_i(v_i)$. The utility is

$$u_i(v_i, b_i, v_{-i}, b_{-i}) = \left(v_i - \max_{j \in \mathcal{I}} b_i\right) \mathbb{1}_{\{i \text{ is the winner}\}} = (v_i - b_i)\mathbb{1}_{\{b_i \geq b_{-i}\}}$$

If $I > 2$, then the utility is now

$$(v_i - b_i)\mathbb{1}_{\{\sigma_i(v_i)\geq\sigma_j(v_j)\forall j\in -i\}} = (v_i - b_i)\prod_{j\in -i}\mathbb{1}_{\{\sigma_i(v_i)\geq\sigma_j(v_j)\}}$$

Then, since we're in the interim setup, the expected value of the utility is

$$\begin{aligned}\mathbb{E}_{-i}\left[u_i(v_i,\sigma_i(v_i),v_{-i},\sigma_{-i}(v_{-i}))\right] =&\mathbb{E}_{-i}\left[(v_i-\sigma_i(v_i))\prod_{j\in -i}\mathbb{1}_{\{\sigma_i(v_i)\geq\sigma_j(v_j)\}}\right]\\ =&(v_i-\sigma_i(v_i))\prod_{j\in -i}\Pr(\sigma_j(v_j)\leq\sigma_i(v_i)),\end{aligned}$$

where the last equality comes from independence.

Note. There are two facts to note.

- Identical valuation implies $\sigma_i \equiv \sigma$ for all $i \in \mathcal{I}$. Same bidding function so it's symmetric.
- $\sigma(\cdot)$ is monotonically increasing. Then

  $$\Pr(\sigma_j(v_j)\leq\sigma_i(v_i)) = \Pr(\sigma(v_i)\leq\sigma(v_i)) = \Pr(v_j\leq v_i) = F(v_i) = v_i,$$

  where F is the CDF, and the last equality comes from the fact that we assume the valuation is uniform($[0,1]$).

  This is a fairly reasonable assumption, since it'll rule out the situations like if two buyers' true value are different, but they somehow decide to bid the same value.

With the second fact of the note above, the expected utility is

$$(v_i - \sigma(v_i))F^{I-1}(v_i).$$

If we assume that this is uniform valuation, then above further equals to

$$(v_i - \sigma(v_i))v_i^{I-1}.$$

Lecture 23: Bayesian Nash Equilibrium

01 Dec. 12:30

Now we look at the expected utility of agent i over other agents' valuation and bids. From the monotonically increasing property of $\sigma(\cdot)$, the expected utility of i is

$$\begin{aligned}\mathbb{E}_{-i}\left[u_i(v_i,\sigma_i(v_i),v_{-i},\sigma_{-i}(v_{-i}))\right] =&\mathbb{E}_{-i}\left[(v_i-\sigma_i(v_i))\prod_{j\in -i}\mathbb{1}_{\{\sigma_i(v_i)\geq\sigma_j(v_j)\}}\right]\\ =&(v_i-\sigma_i(v_i))\prod_{j\in -i}\Pr(\sigma_j(v_j)\leq\sigma_i(v_i))\\ =&(v_i-\sigma_i(v_i))\prod_{j\in -i}\Pr(v_j\leq v_i)\\ =&(v_i-\sigma_i(v_i))F^{I-1}(v_i)\\ =&(v_i-\sigma_i(v_i))v_i^{I-1}.\end{aligned}$$

Now the question is how should an agent bid? Namely, how to find $\sigma_i(\cdot)$?

8.3.1 Reason of Response

To reason about the bidding strategy, we first see an important principle.

Theorem 8.3.1 (Revelation Principle)**.** Let σ be the equilibrium bidding function. Then any deviation can be decomposed into two steps:

(a) $v_i \to v$: Map value to same **false** value.

(b) Apply σ to the **false** value, so that the bid is $\sigma(v)$.

Intuition. Theorem 8.3.1 simply tells us that we don't need to perturb σ itself to analyze incentive of deviation, we can simply perturb v to achieve the same result.

Remark (Equilibrium bidding)**.** The *equilibrium bidding* is

- Step 1: $v_i \to v_i$: identical mapping.
- Step 2: The same, so bid is $\sigma(v_i)$.

Assume a deviation of $v_i \to v$ and the bid is $\sigma(v)$. Then the utility becomes

$$\begin{aligned} u_i &= (v_i - \sigma(v)) \Pr(\text{Winning with bid } \sigma(v)) \\ &= (v_i - \sigma(v)) \prod_{j \in -i} \Pr(\sigma(v_j) \le \sigma(v)) \\ &= (v_i - \sigma(v)) F^{I-1}(v) \\ &= (v_i - \sigma(v)) v^{I-1} \\ &=: f_\sigma(v; v_i). \end{aligned}$$

Then the best response calculation for i is as follows. Given a σ, maximize utility by properly choosing v for each v_i. We simply differentiate the function and set the value be zero

$$\frac{\mathrm{d}}{\mathrm{d}v} f_\sigma(v; v_i) = v_i(I-1)v^{I-2} - (I-1)v^{I-2}\sigma(v) - \sigma^1(v)v^{I-1} := 0$$

for the first order condition. It's equivalent to solve the following O.D.E.:

$$v_i(I-1)v^{I-2} = \frac{\mathrm{d}}{\mathrm{d}v}(\sigma(v)v^{I-1}).$$

We see that $\sigma(v) \le v_i$ is needed as you never bid higher than your valuation in a first-price auction.

For the equilibrium bidding $v_i \to v_i$, the maximizer should be v_i itself. For the equilibrium bidding function σ^*,

$$\frac{\mathrm{d}}{\mathrm{d}v} f_{\sigma^*}(v; v_i)\Big|_{v=v_i} = 0.$$

Expand it out, we have

$$\frac{\mathrm{d}}{\mathrm{d}v} f_{\sigma^*}(v; v_i) = v_i(I-1)v^{I-2} - \left((I-1)v^{I-2}\sigma^*(v) + \frac{\mathrm{d}\sigma^*(v)}{\mathrm{d}v} v^{I-1}\right).$$

By setting $v = v_i$, we should get zero!

$$(I-1)v_i^{I-1} = (I-1)v_i^{I-2}\sigma^*(v_i) + \frac{\mathrm{d}\sigma^*(v)}{\mathrm{d}v}\Big|_{v=v_i} {v_i}^{I-1}.$$

This equation has to hold for every v_i in $[0, 1]$.

Now, for a $\overline{v}$, $\sigma^*(\overline{v})$ should be

$$(I-1)v_i^{I-1} = \frac{\mathrm{d}}{\mathrm{d}v_i}\left(v_i^{I-1}\sigma^*(v_i)\right).$$

Integrate on both sides, we have

$$\int_0^{\overline{v}} (I-1)v_i^{I-1}\,\mathrm{d}v_i = v_i^{I-1}\sigma^*(v_i)\Big|_0^{\overline{v}}.$$

Solve for the left-hand side, we have

$$\frac{I-1}{I}\overline{v}^I = \overline{v}^{I-1}\sigma^*(\overline{v}).$$

Hence,

$$\sigma^*(\overline{v}) = \frac{I-1}{I}\overline{v} = \left(1-\frac{1}{I}\right)\overline{v},$$

which means that agent bids $\frac{I-1}{I}$ times the value who estimate!

Note. We see that

- Agents will shade their bid, hence it's not truthful.
- The good goes to the agent with the highest valuation, hence it's a social-welfare maximizer.

Remark. If an agent whose valuation is zero, then from the function, we see that he would bid zero, hence the utility is zero as well.

8.3.2 Revenue of the Auctioneer

The expected revenue for the auctioneer is equal to

$$\frac{I-1}{I}\mathbb{E}\left[\widetilde{v}_I\right] = \frac{I-1}{I}\frac{I}{I+1} = \frac{I-1}{I+1}.$$

Note. Recall that the expected revenue for second-price auction for the auctioneer is also

$$\mathbb{E}\left[\widetilde{v}_{I-2}\right] = \frac{I-1}{I+1}.$$

We see that both auctions give the same expected revenue.

Remark (Revenue equivalence). If two auctions implement the same outcome in Bayesian Nash equilibrium, and agents with value 0 will bid 0, then the expected revenue will be the same. This is so-called *revenue equivalence*. And this equivalence relation holds for more general kinds of auctions, namely under some conditions of auctions, the revenue for auctioneer is always the same. We'll come back to this relation soon.

8.4 Mechanism Design of Auctions

After seeing two kinds of auctions, we now try to give a general framework of auctions and try to design a mechanism of such games. Assume that the auctioneer doesn't know values of agents.

Problem 8.4.1. How the auction will be conducted?

Answer. Given $b_1, \ldots, b_I$, we need to answer the following two questions:

(a) Who wins the auction?

(b) What that person pays?

The mechanism design of auctions can be though as answering these two questions.

Example (All-pay auction). We first see an example called *all-pay auction*. For bids $b_1, \dots, b_I$,

- The highest bidder wins.
- Everyone pays their bid.

⊛

For a general auction, the bids are just messages. Auctioneer is specifying two things.

- Allocation rule: $(x_1, \dots, x_I)$, where x_i is the amount of good goes to agent i.
- Payment rule: $(p_1, \dots, p_I)$, where p_i to be paid by agent i.

This suggests the following definition.

Definition 8.4.1 (Direct mechanism). In the specification, if the messages are in the same space as values, then we call such a mechanism as a *direct mechanism*.

Remark. From revelation principle, direct mechanisms are sufficient. Since bids are messages, which can be anything, hence it suffices to consider messages to be in the same space as the values.

Hence, we assume that the message space is the same as value space, namely we are designing a direct mechanism. For every message vector, specifies the allocation rule and the payment rule. Furthermore, we let

$$\text{utility} = \text{Value} - \text{Payment},$$

which turns out to be so-called quasilinear utility.

Note. There are things that a mechanism designer may want.

- Incentive compatible (IC).
- Individual rationality (IR).

Also, a mechanism designer may have a goal in mind.

- Social welfare maximization (Efficiency).
- Revenue maximization.
- Budget-Balance: Payments should sum to 0.

Remark. All can't be met simultaneously.

Assume every agent i values $v_1, \dots, v_I$, and bids $b_1, \dots, b_I$ respectively. With the specification being

$$\begin{aligned} \text{Allocations} &: x_1, \dots, x_I; \\ \text{Payments} &: p_1, \dots, p_I, \end{aligned}$$

where all are functions of b_i. Then the utility is

$$u_i(v_i, n_i(\vec{v})) - p_i(\vec{v}).$$

With interim setup, we average across all other agents. Now, assume that $x_i(\vec{v}) \in \{0, 1\}$ and will allow multiple agents to get

$$x_i(\vec{v}) = 1.$$

If $x_i(\vec{v}) = 1$, then utility is v_i, else 0. We then see that

$$u_i(v_i, x_i(\vec{v})) = v_i x_i(\vec{v}) - p_i(\vec{v}).$$

Then the expected value is

$$\begin{aligned}\mathbb{E}_{-i}\left[u_i(v_i, x_i(\vec{v}))\right] &= v_i\mathbb{E}_{-i}\left[x_i(\vec{v})\right] - \mathbb{E}_{-i}\left[p_i(\vec{v})\right] \\ &= v_i\mathbb{E}_{-i}\left[x_i(v_i, v_{-i})\right] - \mathbb{E}_{-i}\left[p_i(v_i, v_{-i})\right] \\ &= v_i\overline{x}_i(v_i) - \overline{p}_i(v_i).\end{aligned}$$

Now we see a theorem.

Theorem 8.4.1 (Bayesian Nash equilibrium characterization). When the types are drawn from a continuous joint distribution $\vec{F}$ with independent types, strategies

$$\vec{\sigma}^* = (\sigma_1^*, \sigma_2^*, \ldots, \sigma_I^*) \qquad (b_i = \sigma_i^*(v_i))$$

are in a Bayesian Nash equilibrium if and only if

- Monotonicity. $\overline{x}_i(v_i)$ is monotonically non-decreasing.
- Payment identity.
$$\overline{p}_i(v_i) = v_i\overline{x}_i(v_i) - \int_0^{v_i} \overline{x}_i(z)\,\mathrm{d}z + \overline{p}_i(0),$$
where often $\overline{p}_i(0) = 0$.

If the strategy profile is onto, the converse is true as well.

Lecture 24: Matching Market

8.5 Bayesian Nash Equilibrium

06 Dec. 12:30

As previously seen. For a valuation $\vec{v}$, the bid $\sigma^*(\vec{v})$ is in the same space as $\vec{v}$ by revelation principle. With allocation being specified as

$$\vec{x}_i(v_i)$$

which is monotonically increasing, and the payment being specified as

$$\overline{p}_i(v_i) = v_i\overline{x}_i(v_i) - \int_0^{v_i} \overline{x}_i(z)\,\mathrm{d}z + \overline{p}_i(0),$$

then we can analyze a general auction with Theorem 8.4.1.

8.5.1 Revenue Equivalence

We start with the expected revenue. The expected revenue is

$$\mathbb{E}_v\left[\sum_i \overline{p}_i(v_i)\right] = \mathbb{E}_v\left[F(\vec{v})\right] + \sum_{i=1}^{I} \overline{p}_i(v).$$

(a) If two mechanisms have the same allocation rule in equilibrium, then

- $\overline{x}_i(v_i)$ will be the same
- $\overline{p}_i(v_i)$ will be the same except for the $\overline{p}_i(0)$ term.

(b) If agent with valuation zero, pay zero, then

$$\overline{p}_i(0) = 0$$

in both mechanisms. This implies that the revenue will be the same.

Note. Similar characterization holds for dominant strategy equilibrium but those ex-post.

- Allocation rule property.
- Payment rule property.

If we now assume independence and identical valuation setting, together with the valuation being uniform($[0,1]$). Then the second-price auction and first-price auction are revenue equivalent.

By symmetry, the payment rules will be identical in each auction, hence the revenue is

$$I \times \text{payment rule for each agent.}$$

- For the second price auction (SP),

$$\overline{p}_1^{\mathrm{SP}}(v_1) = \mathbb{E}\left[\left(\max_{j\in -1} v_j\right)\mathbb{1}_{\{\max_{j\in -1} v_j \le v_1\}}\right].$$

Denote $\widetilde{p}_1 = \max_{j\in -1} v_j$, then

$$F_{\widetilde{p}_i^{\mathrm{SP}}}(x) = \Pr(\widetilde{p}_1 \le x) = \Pr(v_2 \le x, v_3 \le x, \dots, v_I \le x) = x^{I-1},$$

where $x \in [0,1]$. Then,

$$f_{\widetilde{p}^{\mathrm{SP}}}(x) = (I-1)x^{I-2}.$$

We see that

$$\begin{aligned}\widetilde{p}_1^{\mathrm{SP}} &= \mathbb{E}\left[\widetilde{p}_1 \mathbb{1}_{\{\widetilde{p}_1 \le v_1\}}\right] = \int_0^{v_1} x f_{\widetilde{p}_1}(x)\,\mathrm{d}x \\ &= (I-1)\int_0^{v_1} x\cdot x^{I-2}\,\mathrm{d}x = (I-1)\int_0^{v_1} x^{I-1}\,\mathrm{d}x = \frac{I-1}{I}v_1^I.\end{aligned}$$

- For the first price auction (FP),

$$\overline{p}_1^{\mathrm{FP}}(v_1) = \sigma^{\mathrm{FP}}(v_1)\Pr(\text{Agent 1 has the highest value } v_1) = \sigma^{\mathrm{FP}}(v_1)v_1^{I-1},$$

where every other agent's value is less or equal to v_1.

From revenue equivalence,

$$\widetilde{p}^{\mathrm{SP}}(v_1) = \widetilde{p}^{\mathrm{FP}}(v_1),$$

which implies

$$\frac{I-1}{I}v_1^I = \sigma^{\mathrm{FP}}(v_1)v_1^{I-1} \Rightarrow \sigma^{\mathrm{FP}} = \frac{I-1}{I}v_1,$$

as we discussed before.

Chapter 9

Matching Market

Beyond the current setup of auctions, we can see a more general setup. Specifically, we introduce so-called **Matching Market**.

> **Note.** The comparison between auctions and matching market is as follows.
>
> - One good to many buyers: Auctions.
> - Many goods to many buyers: Matching Market.

9.1 Ads Auctions

We first see an example of matching market in terms of a general form of auction, namely *ad-auctions*. Specifically, consider a particular keyword searching result, the ads related to which may show up in the following order:

- AD1 with r_1
- AD2 with r_2
- AD3 with r_3
- AD4 with r_4
- $\vdots$

where r_i is so-called the click-through rate for ad location i. We see that the location of ads matter.

> **Note.** We see that
>
> - Click-through rate is location dependent. Person needs to click on ad and then buy the product the ad is about.
> - Often, $r_i < r_j$ for $i > j$. (i.e., people prefer to click on ads on the top)

We see that there is a bipartite structure. On the one side of the graph contains buyers, and the other side contains sellers (Ad positions). Now, assuming for each buyer's good i, the cost is b_i. Then the expected revenue to advertiser i is

$$b_i r_j$$

if one purchase the location j. Now, let $v_{ij} \coloneqq b_i r_i$, where i denotes the buyers, and j denotes the sellers. Then we have the so-called valuation matrix V defined as follows.

> **Definition 9.1.1** (Valuation matrix)**.** The *valuation matrix* V is matrix with element V_{ij} being the amount that bidder i values good j.

We see that in our case, i.e., ads-auctions, our valuation matrix can be defined as

$$V \coloneqq \vec{r}^{\top} \vec{b}.$$

If $V_{ij} \geq 0$ and beyond rank-one imply good allocation here. Note that we allow the sellers to set prices to goods.

Remark. There are different types of prices.

- Posted. Item comes with price.
- Anonymous. The price is the same for any buyers.

Definition 9.1.2 (Balance)**.** The market is called *balance* if the number of sellers and buyers are the same.

Without loss of generality, we assume that the market is balanced by letting

$$V \in \mathbb{R}^{N \times N},$$

which implies there are a same number (N) of buyers and sellers.

Remark (Balance the market)**.** This can be achieved by adding dummy buyers or sellers.

- Dummy buyers: Value every good as 0.
- Dummy sellers: Valued as 0 for every buyer.

Consider each seller/item coming with a posted and anonymous price, namely item j has price $p_j \geq 0$.

Problem 9.1.1. How does a buyer i react to this? Assuming i knows his valuation, and each buyer can only purchase one good.

Answer. Suppose the valuation for buyer i is $V_{i1}, V_{i2}, \ldots, V_{iN}$, and the prices for each good is $p_1, p_2, \ldots, p_N$. Then it's clear that buyer i will

(a) only consider item whose valuation exceeds the price,

(b) focus only on goods whose payoff is maximum (valuation − price).

Mathematically, we have

$$\arg\max_{j\colon V_{ij} \geq p_j} V_{ij} - p_j.$$

For convenience, we define a function $[\,\cdot\,]_+$ such that

$$[x]_+ \coloneqq \max(x, 0) = \begin{cases} x, & \text{if } x \geq 0; \\ 0, & \text{if } x \leq 0. \end{cases}$$

Then the utility of buyer i at price vector $\vec{p}$ is

$$u_i = \left[\max_{j \in \{1,\ldots,N\}} V_{ij} - p_j\right]_+ = \max_{j \in \{1,\ldots,N\}} \left[V_{ij} - p_j\right]_+.$$

We then defined so-called preferred sellers list $s(i)$ for buyer i as follows.

Definition 9.1.3 (Preferred sellers list)**.** The *preferred sellers list*, denotes as $s(i)$ for buyer i, is defined as

$$s(i) \coloneqq \begin{cases} \varnothing, & \text{if } \forall j \ V_{ij} < p_j; \\ \arg\max_j \left(V_{ij} - p_j\right), & \text{otherwise.} \end{cases}$$

Note that $s(i)$ contains goods that buyer i is interested in.

Remark. If $s(i) \neq \varnothing$, then there is at last one seller that buyer i is interested in.

We then use the preferred sellers lists to construct an undirected bipartite graph between buyers and sellers. ⊛

Example. Let $N = 3$, and

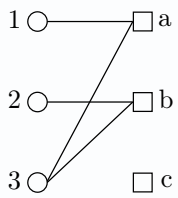
$$V \coloneqq \begin{pmatrix} 5 & 3 & 4 \\ 3 & 8 & 3 \\ 5 & 4 & 7 \end{pmatrix}, \quad p \coloneqq \begin{pmatrix} 4 \\ 3 \\ 7 \end{pmatrix},$$

the goal is to fulfill the market as much as possible.

Proof. Then, we have

1. Buyer 1. $V_{11} - p_1 = 1$, $V_{12} - p_2 = 0$, $V_{13} - p_3 = -3$. Hence,
$$s(1) = \{1\}.$$
2. Buyer 2. $V_{21} - p_1 = -1$, $V_{22} - p_2 = 5$, $V_{23} - p_3 = -4$. Hence,
$$s(2) = \{2\}.$$
3. Buyer 3. $V_{31} - p_1 = 1$, $V_{32} - p_2 = 1$, $V_{33} - p_3 = 0$. Hence,
$$s(3) = \{1, 2\}.$$

We see that c is not desired by any buyers at current price.

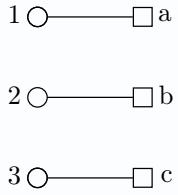


⊛

Example. Consider the same valuation matrix V as above while change p into

$$p = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Proof. Then we see that satisfaction is possible.

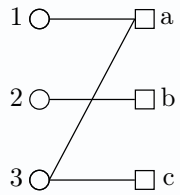


⊛

Example. Consider the same valuation matrix V as above while change p into

$$p = \begin{pmatrix} 5 \\ 8 \\ 7 \end{pmatrix}.$$

Proof. Then we see that satisfaction is possible as well.



⊛

Remark. We see that the satisfaction match is not always possible.

9.2 Perfect Matching

We now discuss a general matching problem.

Remark. Note that the discussion about ad-auctions is indeed general enough to be considered as a general market matching problem since if we neglect the meaning of V under the content of ads-auctions, instead just interpret V_{ij} as in the original definition, then it's just a general matching problem.

The task is that for any given price and the valuation matrix V, find the preferred sellers bipartite graph and perform matching. Furthermore, we try to maximize the matching in terms of size. In other words, we are trying to satisfy as many buyers as possible. This suggests the following definition.

Definition 9.2.1 (Perfect matching)**.** We call a matching is a *perfect matching* if the size of the maximum matching is N.

To discuss perfect matching, we first defined some notations. Let

$$S \subseteq \{1, \dots, N\}$$

be the subset of sellers, and

$$B \subseteq \{1, \dots, N\}$$

be the subset of buyers. We further denote $N(S)$ be the union of all buyers that like goods in S, and $N(B)$ be the union of preferred sellers of all the buyers in B.

Definition 9.2.2 (Constricted set)**.** Given the definition of $N(S)$, we say S is a *constricted set* if $|S| > |N(S)|$.

Note. Similarly, if $|B| > |N(B)|$ then B is constricted.

Theorem 9.2.1 (Kőnig-Hall maximize theorem)**.** A bipartite graph has a perfect matching if and only if there are no constricted sets.

Note. This is a necessary and sufficient conditions for a perfect matching to exist given a bipartite graph.

Remark. With naive approach, we need to check every non-empty subset of sellers. Hence, there are $2^N - 1$ checks, but polynomial time checks exists. Hence, We can also find a perfect matching in polynomial time.

> **Note** (Market clearing prices). The prices for which a perfect matching exists are called *market clearing prices*.

Lecture 25: Perfect Matching

We now want to solve the following questions. 08 Dec. 12:30

Problem 9.2.1. Given a bipartite graph, can we check if there is a perfect matching?

Answer. Yes, by Hopcroft-Karp algorithm, which is an asymmetric BFS algorithm. ⊛

Problem 9.2.2 (Market clearing prices). Given a market, do the market clearing prices exist? If they do, can we find them?

Answer. Yes. We can find it in polynomial time by Hungarian algorithm. Also, by using VCG principle, we can obtain specific market clearing prices. ⊛

Problem 9.2.3 (Social welfare). What's the connection between market clearing prices and social welfare maximizing.

9.3 Hopcroft-Karp Algorithm

To answer the first question, since we already have the powerful Theorem 9.2.1 which characterize the existence of a perfect matching, a simple way to idea is to check whether a constricted set exists or not. But a brute-force approach will require we to check for all subsets of $\{1, 2, \ldots, N\}$, which is exponential.

Hence, another simple idea is to find a maximum matching and check whether it's perfect, i.e., whether a maximum matching has size N. Note that we'll have to use Theorem 9.2.1. We first see some definitions which is important to understand the algorithm.

Definition 9.3.1 (Augmenting path). Let M be a bipartite matching in $\mathcal{G} = (\mathcal{V} = L \bigsqcup R, \mathcal{E})$. Then we say that a simple path P in $\mathcal{G}$ is an *augmenting path* w.r.t. M if it starts at an unmatched vertex in L, ends at an unmatched vertex in R with its edges belong alternately to M and $E - M$.

Note. Definition 9.3.1 s related to, but different from, an augmenting path in a flow network algorithm like Ford-Fulkerson algorithm.

Intuition. To find an augmenting path, we utilize the idea of breadth-first search since it'll build up a level graphs in its nature, and in the case of a bipartite graph, with some checking, we can build up a level graph which the end points of edges alternating between matched and unmatched vertices.

Then, we'll see that in the algorithm we're going to develop, we'll need to find the shortest augmenting paths, and this is done by using depth-first search.

Algorithm 9.1: Hopcroft-Karp Algorithm

Data: A bipartite graph $\mathcal{G} = (\mathcal{V} = L \bigsqcup R, \mathcal{E})$
Result: A maximum matching M

1 $M \leftarrow \varnothing$ `// Initialize`
2 **do**
3 $\quad \mathcal{P} \leftarrow \{P_i\}_{i=1}^{k}$ `// Maximal set of disjoint shortest augmenting paths w.r.t.` M
4 $\quad M = M \oplus (\bigcup_{i=1}^{k} P_i)$ // $A \oplus B \coloneqq (A - B) \cup (B - A)$
5 **while** $\mathcal{P} \neq \varnothing$
6 **return** M

Intuition. We see that

- If there is an augmenting path, we can augment the total size of the matching by 1 by altering this path. Specifically, if an augmenting path exists, then flip all the labels of the edges, and label the free buyers and sellers to matched. This will increase the size of the matching by 1.
- This is a polynomial time algorithm, since we will only explore every node once.

Remark (Finding maximal set of disjoing shortest augmenting paths w.r.t. M)**.** We see that it's not entirely clear how should we find P_i in line 3 at the first glance. It's basically just a combination of BFS and DFS as we'll see.

Proof. For each unmatched vertex in L, we perform a modified BFS to find the length of the shortest path to an unmatched vertex in R. Modify the BFS to ensure that we only traverse an edge if it causes the path to alternate between an edge in M and an edge in $E \setminus M$. The first time an unmatched vertex in R is reached we know the length k of a shortest augmenting path.

We can use this to stop our search early if at any point we have traversed more than that number of edges. To find disjoint paths, start at the vertices of R which were found at distance k in the BFS. Run a DFS backwards from these, which maintains the property that the next vertex we pick has distance one fewer, and the edges alternate between being in M and $E \setminus M$. As we build up a path, mark the vertices as used so that we never traverse them again. ⊛

Remark. We see that line 3 takes $O(|\mathcal{E}|)$ to complete in the worst case.

Now we analyze the Hopcroft-Karp algorithm rigorously. We first prove that the correctness of Hopcroft-Karp algorithm.

9.3.1 Correctness Analysis

We first see an important lemma which characterizes the behavior of line 3 and line 4.

Lemma 9.3.1. If M is a matching and P is an augmenting path w.r.t. M, then the symmetric difference $M \oplus P$ is a matching and $|M \oplus P| = |M| + 1$. Furthermore, if $P_1, P_2, \ldots, P_k$ are vertex-disjoint augmenting paths w.r.t. M, then the symmetric difference

$$M \oplus (P_1 \cup P_2 \cup \cdots \cup P_k)$$

is a matching with cardinality $|M| + k$.

Proof. Suppose M is a matching and P is an augmenting path w.r.t. M, then P consists of k edges in M and $k + 1$ edges not in M by the definition.[a]

This implies that

$$|M \oplus P| = |M| + |P| - 2k = |M| + 2k + 1 - 2k = |M| + 1,$$

since we must remove each edge of M within is in P from both M and P. Now suppose $P_1, P_2, \ldots, P_k$

are vertex-disjoint augmenting paths w.r.t. M. Let k_i be the number of edges in P_i which are in M, so that $|P_i| = 2k + i + 1$. Then we have

$$M \oplus (P_1 \cup P_2 \cup \cdots \cup P_k) = |M| + |P_1| + \cdots + |P_k| - 2k_1 - 2k_2 - \cdots - 2k_k = |M| + k.$$

To see that we in fact get a matching, suppose that there was some vertex v which had at least 2 incident edges e and e'. They cannot both come from M since M is a matching. They cannot both come from P since P is simple and every other edge of P is removed. Thus, $e \in M$ and $e' \in P \setminus M$. However, if $e \in M$, then $e \in P$, so $e \notin M \oplus P$, a contradiction. A similar argument gives the case of $M \oplus (P_1 \cup \cdots \cup P_k)$. ■

[a] Specifically, the first edge of P touches an unmatched vertex in L, so it cannot be in M Similarly, the last edge in P touches an unmatched vertex in R, so the last edge cannot be in M. Since the edges alternate being in or not in M, there must be exactly one more edge not in M than in M.

We see that Lemma 9.3.1 ensures that M is always a matching, also, M will increase k at each iteration. We now analyze why whenever there are no augmenting paths, i.e., $\mathcal{P} = \varnothing$, M is optimal.[1]

Theorem 9.3.1. If a matching M is not optimal, then there exists at least one augmenting path.

Proof. Suppose that a matching M is not optimal, and let P be the symmetric difference $M \oplus M^*$ where M^* is an optimal matching. Since M and M^* are both matchings, every vertex has degree at most 2 in P. So P must form a collection of disjoint cycles, of paths with an equal number of matched and unmatched edges in M of augmenting paths for M, and of augmenting paths for M^*. But the latter is impossible because M^* is optimal.

Now, the cycles and the paths with equal numbers of matched and unmatched vertices do not contribute to the difference in size between M and M^*, so this difference is equal to the number of augmenting paths for M in P. Thus, whenever there exists a matching M^* larger than the current matching M, there must also exist an augmenting path. ■

Hence, with Theorem 9.3.1, we see that Hopcroft-Karp algorithm terminate with the desired result. We now want to know its time complexity.

Remark (Finite termination). Before doing a formal time complexity analysis, we note that just from Lemma 9.3.1 and Theorem 9.3.1, we already have finite termination guarantee for Hopcroft-Karp algorithm. It's because that A maximum matching M^* is always finite since we only consider a finite bipartite graph, and from Lemma 9.3.1, at each iteration before termination, we must find at least one augmenting path P, and from Lemma 9.3.1, the corresponding new matching $M \oplus P$ has cardinality increased at least by 1, hence we see that we're guaranteed for a finite termination.

9.3.2 Time Complexity Analysis

In this section, we're going to prove that the time complexity of Hopcroft-Karp algorithm is $O(\sqrt{|\mathcal{V}|}\,|\mathcal{E}|)$, hence it's indeed a polynomial time algorithm. We first see a lemma.

Lemma 9.3.2. Given two matchings M and M^* in $\mathcal{G}$, then every vertex in the graph $\mathcal{G}' = (\mathcal{V}, M \oplus M^*)$ has degree at most 2. Hence, G' is a disjoint union of simple paths or cycles. Furthermore, the edges in each such simple path or cycle belong alternately to M or M^*. And if $|M| \leq |M^*|$, then $M \oplus M^*$ contains at least $|M^*| - |M|$ vertex-disjoint augmenting paths w.r.t. M.

Proof. Suppose some vertex in G' has degree at least 3. Since the edges of G' come from $M \oplus M^*$, at least 2 of these edges come from the same matching. However, a matching never contains two edges with the same endpoint, so this is impossible. Thus, every vertex has degree at most 2, so G' is a disjoint union of simple paths and cycles.

If edge (u, v) is followed by edge (z, w) in a simple path or cycle, then we must have $v = z$. Since two edges with the same endpoint cannot appear in a matching, they must belong alternately to M and M^*. Since edges alternate, every cycle has the same number of edges in each matching and every path has at most one more edge in one matching than in the other. Thus, if $|M| \leq |M^*|$,

[1] Note that optimal doesn't mean perfect, just maximum.

there must be at least $|M^*| - |M|$ vertex-disjoint augmenting paths w.r.t. M. ■

Now, let ℓ be the length of a shortest augmenting path w.r.t. a matching M, and let $P_1, P_2, \ldots, P_k$ be a maximal set of vertex-disjoint augmenting paths of length ℓ w.r.t. M. Let $M' \coloneqq M \oplus (P_1 \cup \ldots P_k)$, and suppose that P is a shortest augmenting path w.r.t. M'.

With the above notation, we see that following.

Lemma 9.3.3. If P is a vertex-disjoint path disjoint from $P_1, P_2, \ldots, P_k$, then P has more than ℓ edges.

Proof. Every vertex matched by M must be incident with some edge in M'. Since P is augmenting w.r.t. M', the left endpoint of the first edge of P isn't incident to a vertex touched by an edge in M'. In particular, P starts at a vertex in L which is unmatched by M since every vertex of M is incident with an edge in M'. Since P is vertex-disjoint from $P_1, \ldots, P_k$, any edge of P which is in M' must in fact be in M and any edge of P which is not in M' cannot be in M. Since P has edges alternately in M' and $E \setminus M'$, P must in fact have edges alternately in M and $E \setminus M$. Finally, the last edge of P must be incident to a vertex in R which is unmatched by M'. Any vertex unmatched by M' is also unmatched by M, so P is an augmenting path for M. P must have length at least ℓ since ℓ is the length of the shortest augmenting path w.r.t. M. If P had length exactly ℓ, then this would contradict the fact that $P_1 \cup P_2 \cup \cdots \cup P_k$ is a maximal set of vertex-disjoint paths of length ℓ because we would add P to the set. Thus, P has more than ℓ edges. ■

Lemma 9.3.4. Suppose P is not vertex-disjoint from $P_1, P_2, \ldots, P_k$. Let A be the set of edges $(M \oplus M') \oplus P$. Then $A = (P_1 \cup P_2 \cup \cdots \cup P_k) \oplus P$ and

$$|A| \geq (k+1)\ell,$$

hence P has more than ℓ edges.

Proof. Any edge in $M \oplus M'$ is in exactly one of M or M'. Thus, the only possible contributing edges from M' are from $P_1 \cup \cdots \cup P_k$. An edge from M contribute if and only if it is not in exactly one of M and $P_1 \cup \cdots \cup P_k$, which means it must be in both. Thus, the edges from M are redundant so $M \oplus M' = (P_1 \cup \cdots \cup P_k)$ which implies

$$A = (P_1 \cup \cdots \cup P_k) \oplus P.$$

Now we'll show that P is edge disjoint from each P_i. Suppose that an edge e of P is also an edge of P_i for some i. Since P is an augmenting path w.r.t. M', either $e \in M'$ or $e \in \mathcal{E} \setminus M'$.

- $e \in M'$: Since P is also augmenting w.r.t. M, we must have $e \in M$. However, if $e \in M$ and also $e \in M'$, then e cannot be in any of the P_i's by the definition of M'.
- $e \in \mathcal{E} \setminus M'$: Then $e \in \mathcal{E} \setminus M$ since P is augmenting w.r.t. M. Since e is an edge of P_i, $e \in \mathcal{E} \setminus M'$ implies $e \in M$, a contradiction.

Since P has edges alternately in M' and $\mathcal{E} \setminus M'$ and is edge disjoint from $P_1 \cup \cdots \cup P_k$, P is also an augmenting path for M, which implies $|P| \geq \ell$. Since every edge in A is disjoint, we conclude that

$$|A| \geq (k+1)\ell.$$

■

Lemma 9.3.5. If a shortest augmenting path w.r.t. M has ℓ edges, the size of the maximum matching is at most

$$|M| + |\mathcal{V}| / (\ell + 1).$$

Proof. Suppose M^* is a matching with strictly more than $|M| + |\mathcal{V}| / (\ell + 1)$ edges. Then by Lemma 9.3.2, there are strictly more than $|\mathcal{V}| / (\ell + 1)$ vertex-disjoint augmenting paths w.r.t. M. Each one of these contains at least ℓ edges, so it is incident on $\ell + 1$ vertices. Since the paths are

vertex disjoint, there are strictly more than

$$|\mathcal{V}|\,(\ell+1)/(\ell+1)$$

distinct vertices incident with these paths, a contradiction. Thus, the size of the maximum matching is at most

$$|M| + |\mathcal{V}|\,/(\ell+1).$$

■

Theorem 9.3.2. The number of the iteration in Hopcroft-Karp algorithm is at most $2\sqrt{|\mathcal{V}|}$, and the time complexity of Hopcroft-Karp algorithm is $O(\sqrt{|\mathcal{V}|}\,|\mathcal{E}|)$.

Proof. Consider what happens after iteration number $\sqrt{|\mathcal{V}|}$. Let M^* be a maximal matching in $\mathcal{G}$, then $|M^*| \geq |M|$, hence from Lemma 9.3.2, we have that $M \oplus M^*$ contains at least $|M^*| - |M|$ vertex-disjoint augmenting paths w.r.t. M. Then from Lemma 9.3.3, each of these is also an augmenting path for M. Since each has length $\sqrt{|\mathcal{V}|}$, there can be at most $\sqrt{|\mathcal{V}|}$ such paths, so

$$|M^*| - |M| \leq \sqrt{|\mathcal{V}|}.$$

Hence, only $\sqrt{|\mathcal{V}|}$ additional iterations of the repeated loop can occur, hence there are at most $2\sqrt{|\mathcal{V}|}$ iteration in total for Hopcroft-Karp algorithm.

Together with the remark we have seen, one iteration needs $O(|\mathcal{E}|)$, hence we conclude that the total runtime is $O(\sqrt{|\mathcal{V}|}\,|\mathcal{E}|)$. ■

9.3.3 An Alternated Approach

Another way to answer whether a given market has perfect matching is as follows. We first define a new label called `exhausted`[2], which means that for a free buyer node, no augmenting paths are found. Then, with the notion of constricted set, we state a fundamental theorem about the **only** constraint when constructing a perfect matching, without proving it.

Theorem 9.3.3. If there are no exhausted buyer nodes, then we have a perfect matching. Otherwise, if there are exhausted buyer nodes, then we can find some constricted sets.

Remark. Theorem 9.3.3 essentially tells us that the only obstacle one will meet when constructing a perfect matching is the existence of constricted set. Compare to Theorem 9.2.1, this is an even stronger result in the sense that it gives us a useful characterization of constricted set when we want to actually check the condition given in Theorem 9.2.1 algorithmically.

Now, we simply run a modified version of Hopcroft-Karp algorithm with the special label `exhausted` and utilizing Theorem 9.3.3. But it's essentially just the same thing, hence we omit it here.

9.4 Market Clearing Prices

We now come back to market clearing prices problem.

Problem 9.4.1. Given valuation matrix V and price vector p, is it always the case that a market clearing price exists?

Answer. Yes, and we can find it in polynomial time. (Hungarian algorithm do it in $O(n^3)$, which is strongly polynomial time) ⊛

Without loss of generality, as assume that $V_1 > V_2 > V_3 \geq \cdots \geq V_N$. Then we easily see that if

- $p = (V_1, 0, \ldots, 0)$, then a perfect matching exists.

[2]Note that we only have `matched` or `free` before.

> **Note.** This is essentially a first price auction.

- $p = (V_2, 0, \ldots, 0)$, then a perfect matching exists as well.

  > **Note.** This is essentially a second price auction.

  Furthermore, if

  - $p_1 > V_1$, then no perfect matching exists.
  - $p_1 < V_2$, then no perfect matching exists as well.

We claim that market clearing prices always exists, and will follow so-called *Lattice structure*. Says if p^1 and p^2 are both market clearing prices, then

$$p^1 \vee p^2 := \text{element-wise max}$$
$$p^1 \wedge p^2 := \text{element-wise min}$$

are also market clearing prices, where

- Max market clearing price is by taking the maximum in all components.
- Min market clearing price is by taking the minimum in all components.

> **Remark.** There is a nice property for market clearing prices: It's always social welfare maximizing, respect to the utility of all buyers and the sum of prices (seller made).

We can actually formulate the above question as a linear programming. Define $x_{ij} \in [0, 1]$ as the part of good j in assigned to buyer i, namely we allow fractional assignment. Then we model the above question as

$$\begin{aligned} V^* := \max \quad & \sum_{i,j} V_{ij} x_{ij} \\ & \sum_j x_{ij} = 1 \ \forall i \\ & \sum_i x_{ij} = 1 \ \forall j \\ & x_{ij} \in [0, 1]. \end{aligned}$$

> **Intuition.** We see that
>
> - The first constraint says that buyer want one good.
> - The second constraint says that seller has one good for type j.

If we further restrict the fractional assignment property, then we have the integer programming version of above, namely

$$\begin{aligned} V^* := \max \quad & \sum_{i,j} V_{ij} x_{ij} \\ & \sum_j x_{ij} = 1 \ \forall i \\ & \sum_i x_{ij} = 1 \ \forall j \\ & x_{ij} \in \{0, 1\}. \end{aligned}$$

> **Remark.** We see that
>
> - Perfect matching is the only possible solution.
> - Solving the linear programming leads to an integer solution.

- The market clearing price induced by perfect matching will achieve V^* in the integer programming, which is social welfare maximizing.

9.5 Demange-Gale-Sotomayor Algorithm

Given an English auction for multiple goods, with valuations and prices being all in integers, we can run the so-called Demange-Gale-Sotomayor (DGS) algorithm and get a set of market clearing prices.

Algorithm 9.2: Demange-Gale-Sotomayor Algorithm

Data: valuation matrix $V_{N\times N}$
Result: A perfect matching M, Final price p

for $j = 1, \dots, N$ **do** // Initialize
  $p_j \leftarrow 0$

do
  $\mathcal{G} \leftarrow$ `PreferredSellerGraph`(p) // Get the preferred sellers bipartite graph
  $M \leftarrow$ `Hopcroft-Karp`$(\mathcal{G})$ // Get a maximum matching of $\mathcal{G}$
  if $|M| = N$ **then** // If M is perfect
    return M, p
  else
    $N \leftarrow$ `FindConstrictedSet`$(\mathcal{G}, M)$ // Find $B' \subseteq B$ s.t. $|N(B')| < |B'|$
    $S \leftarrow$`randSubset`(N) // Pick any $S \subseteq N(B)$ s.t. $S \neq \varnothing$
    for $i = 1, \dots, N$ **do**
      $p_i \leftarrow p_i + 1$
while *True*

Intuition. If there is more demand and less supply, then we increase the price.

Remark (Polynomial time). This algorithm will terminate in finite-time, further, it's actually a polynomial time algorithm since finding a constricted set in this case is not exponential anymore with a maximum matching M given, and the fact that Hopcroft-Karp algorithm is a polynomial time algorithm, we have the desired result.

If one carefully enough for choosing S, he will always terminate in the minimum market clearing prices.

Note. Note that it's easy to extend to the case that the market is not balance, since Hopcroft-Karp algorithm can deal with unbalance market.

9.6 Vickrey-Clarke-Groves Principle

Consider a second price auction. There is only 1 item to be sold with N buyers with valuation $V_1, V_2, \dots, V_N$. We further let

$$V_1 > V_2 > V_3 \geq \dots V_N.$$

In this case, we see that agent 1 gets the good with price V_2. We denote the utility as $\Delta \coloneqq V_1 - V_2 > 0$.

Problem 9.6.1. What's the utility of other players?

Answer. 0. ⊛

We now conduct a thought experiment. If we remove agent 1 and replace him with a dummy agent, says $\tilde{1}$, then we see that agent 2 will get the good with price V_3. Further, we see that the sum of the values obtained by agents other than $\tilde{1}$ is V_2, since now agent 2 gets the good and his valuation is V_2.

We see that by the appearance of agent 1, other agents collectively lost V_2 values. We call such collectively lost values as *externality* agent 1 imposes.

With this view point to view a second price auction, we see the following principle.

Theorem 9.6.1 (Vickrey-Clarke-Groves principle)**.** Charge a price to an agent as the externality agent imposes on others.

Example. We see that for

- Agent 1 should be charged with price V_2
- Agent 2 should be charged with:

  In the original auction, the value of all other agents are V_1. Now, replace agent 2 by a dummy agent $\tilde{2}$, then the value of all agents other than $\tilde{2}$ is V_1. Hence, the difference is

  $$\Delta_2 = V_1 - V_1 = 0,$$

  namely by VCG principle, we charge agent 2 with price 0.

To apply VCG principle in a matching market problem mathematically, we first denote the total set of sellers as S, and the total set of buyers as B. Recall the definition of V^*, we further specify it by adding the superscript and subscript as

$$V_B^{*,S}$$

for considering S and B in the programming. Then, for $i \in B$, denote $j^*(i)$ as the matched seller respect to i in a perfect matching that maximize social welfare. Now, consider the *reduced problem*, namely calculate

$$V_{B-i}^{*,S-j^*(i)},$$

which stands for the value of everyone else when agent i is not present.

Lastly, we consider the alternate problem, where we need to calculate the value that everyone else gets if i is not present, namely we calculate

$$V_{\widetilde{B}}^{*,S}$$

where $\widetilde{B} \coloneqq B - i + \varnothing$, where $\varnothing$ stands for a dummy agent $\widetilde{i}$.

Combining the reduced problem and the alternate problem, together with the VCG principle, we see that the auctioneer should charge buyer i

$$p_i^{\mathrm{VCG}} = V_{\widetilde{B}}^{*,S} - V_{B-i}^{*,S-j^*(i)}.$$

We now simply let

$$p_{j^*(i)} = p_i^{\mathrm{VCG}},$$

namely we make the price as posted price.

Remark. We finally note that

- It's a market clearing price, furthermore, it's actually the minimum market clearing price.
- There is a conceptual difference between the VCG approach and the DGS algorithm approach. One is posted, one is not. We see that in VCG approach, we determine the prices for each agent by their valuation, while this is not the case in DGS algorithm approach.

Appendix

This note is completed in LaTeXwith Inkscape, in case of anyone is interested, please check out this blog[3] together with my configuration[4].

[3] https://castel.dev/
[4] https://github.com/sleepymalc/VSCode-LaTeX-Inkscape

Bibliography

[EK10] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010. ISBN: 9781139490306. URL: https://books.google.com/books?id=atfCl2agdi8C.