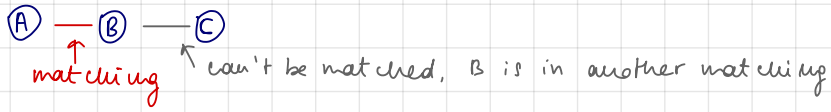
MATCHING in ONLINE ENVIRONMENTS

Question 1

- **Mathematical formulation of a matching problem**

we are given a graph (set of edges and links) and the goal is to find a matching.
a MATCHING is a subset of edges of the graph such that no pair of edges share vertices
→ every vertex can occur once in a matching

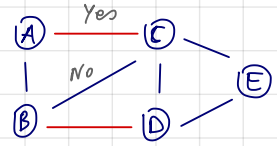


* maximal matching → matching that cannot be enlarged
* maximum matching → matching with largest cardinality (num. of edges)
* perfect matching → matching where all nodes are matched (it may no exist)

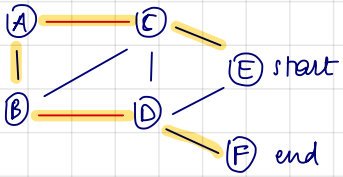
⇒ GOAL: find a maximum matching

- **Describe what an alternating and augmenting paths are**

* alternating path → path such that edges in the matching and edges of the complementary set (edges not in the matching) alternate



* augmenting path → alternating path starting and ending in unmatched vertices



- **Describe the alternating path algorithms for bipartite and arbitrary graphs**

The idea of the alternating-path algorithm is to find an augmenting path, do the complementary in order to find a new alternating path whose cardinality is strictly larger than the cardinality of the initial path
→ we find a maximum matching

- Alternating-path algorithm works iteratively until no augmenting path exists
  - look for an augmenting path
  - make the complementary

1. Consider every unmatched vertex (as root of a subtree)
2. Make a breadth-first search starting from every unmatched vertex
3. For every unexplored path which is not in the matching, make an expansion adding that path and the connected path in the matching if the vertex is matched.
4. Label vertices as blue/green according to the level of search tree (even/odd)
5. If the path in the expansion contains some visited vertices then stop the branch and take a decision according to the specific case.

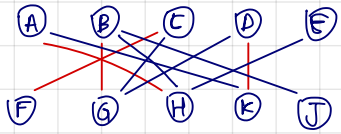
   case 1 If the expanding even node is directly connected to an even node (previously labeled) we found an augmenting path

   case 2 If the expanding even node is directly connected to an odd node (previously labeled) then stop the search

   case 3 (only for arbitrary graph) If the expanding even node is directly connected to a vertex previously labeled as even and belonging to the same subtree then we found a blossom.

Question 2 and 3

- Apply the alternating path algorithm to a bipartite graph

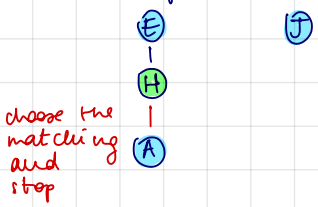


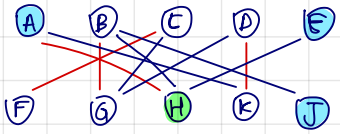
$M = \{(A,H), (C,F), (D,K), (B,G)\}$ links in the matching

$U = \{(E,J)\}$ unmatched nodes → even

$J = \{E, J\}$ nodes to be visited

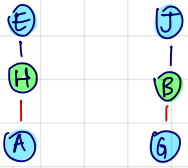
1° step : Expand E

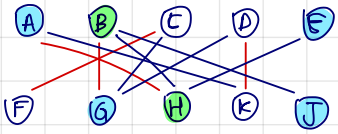




$J = \{J, A\}$

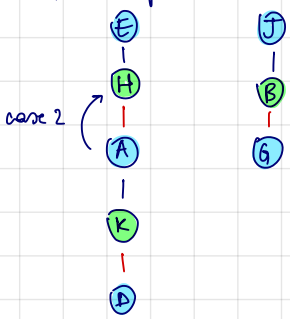
2° step : Expand J

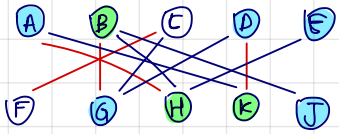




$J = \{A, G\}$

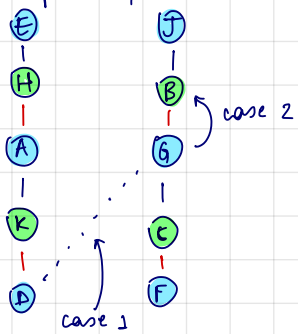
3° step : expand A

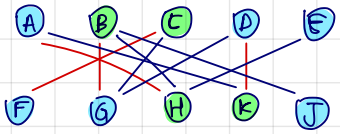




$J = \{G, D\}$

4° step : expand G





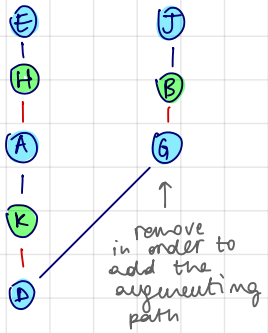
$J = \{D, F\}$

G - D augmenting path

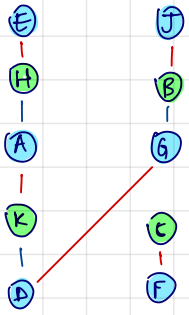
5° step : expand D D → K stop,
D → G already visited

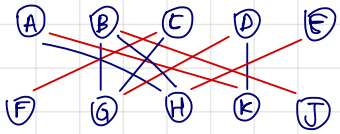
6° step : expand F F → C already visited

so we find

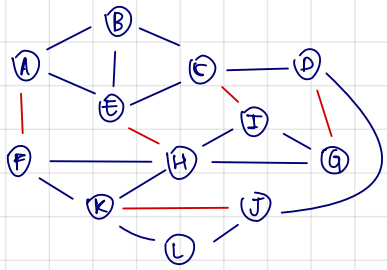


complementary





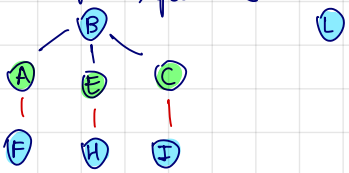
- Apply the alternating path for arbitrary graph

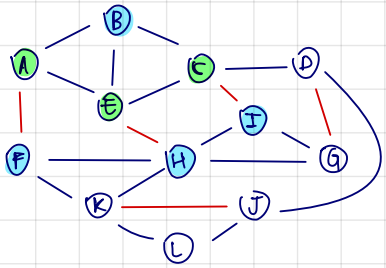


$M = \{(A,F), (E,H), (C,I), (D,G), (K,J)\}$

$U = \{(B,L)\}$ $\quad Y = \{B, L\}$

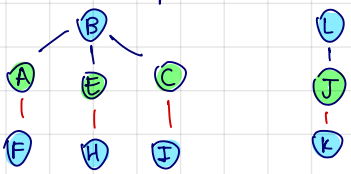
1° step: expand B

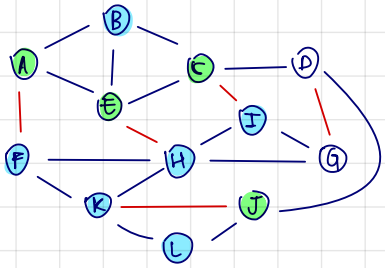




$Y = \{L, F, H, I\}$

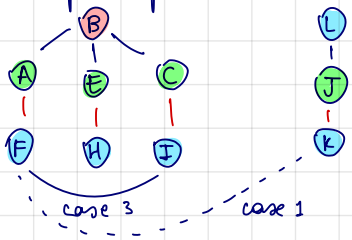
2° step: expand L





$Y = \{F, H, I, K\}$

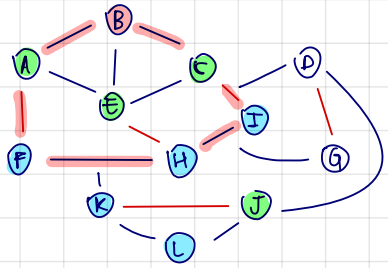
3° step: expand F



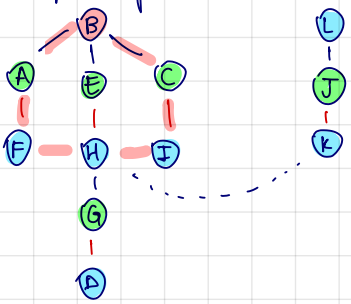
B → vertex of a blossom

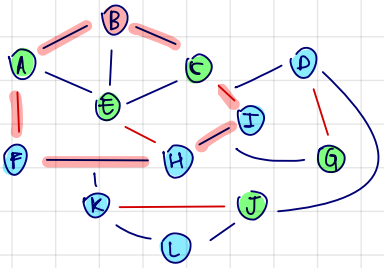
F - K → augmenting path

$Y = \{H, I, K\}$



4° step: expand H

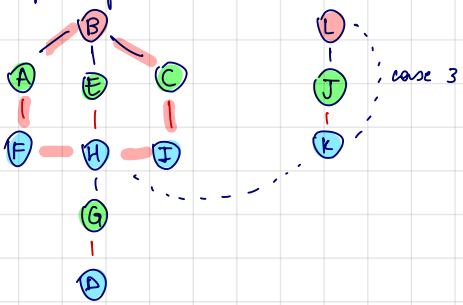




$Y = \{I, K, D\}$

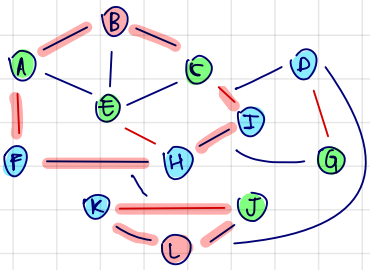
5° step: expand I → close

6° step: expand k

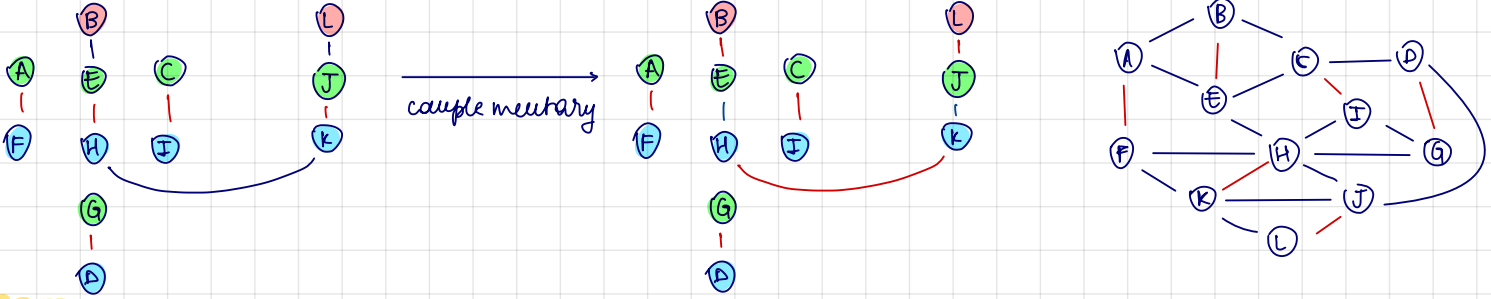


L → vertex of a blossom

$Y = \{D\}$



7° step: expand D → stop



Question 4

- **Define what a combinatorial bandit problem is**

Given a set of candidates (arms = prices) and an unknown expected reward for each arm, a combinatorial bandit problem is a problem in which we have combinatorial constraints, so a subset of feasible arms, whose arms are pulled simultaneously. These arms are the solution of a combinatorial problem.

We define: superarm → collection of candidates ($\underline{a}$)

feasible superarm → superarm satisfying the (combinatorial) constraint $C(\underline{a}) = 0$

reward of a superarm → sum of the reward included in a superarm

→ Goal: maximize the cumulative in time expected reward

- **Describe how a matching problem can be formulated as a combinatorial bandit problem**

→ In weighted matching problems the value associated to an edge is unknown, and we need to estimate it by formulating the problem as a combinatorial bandit problem

A matching problem can be formulated as a combinatorial bandit problem as follows:

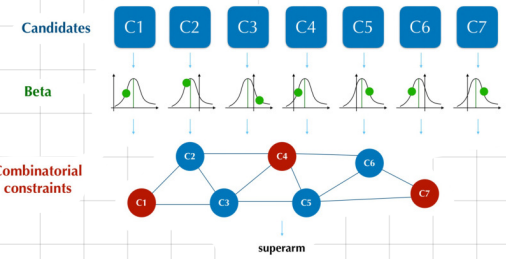
- a superarm is a matching (a set of edges)
- arms are nodes of the graph

- **Describe the combinatorial TS algorithm**

Assume that the random variable of every arm is a Bernoulli in $\{0, 1\}$ with unknown mean

1. At every time t and for every arm a
   $\tilde{\theta}_a \leftarrow$ sample $(\mathbb{P}(\mu_a = \theta_a))$
2. At every time t, play superarm $\underline{a}_t$ (solve the combinatorial problem)
   $\underline{a}_t \leftarrow \underset{C(\underline{a})=0}{\operatorname{argmax}} \left\{ \sum_{a \in \underline{a}} \tilde{\theta}_a \right\}$
3. Update the Beta distribution
   $(\alpha_{a_t}, \beta_{a_t}) \leftarrow (\alpha_{a_t}, \beta_{a_t}) + (X_{a_t,t};\ 1 - X_{a_t,t})$

X_a: random variable of the reward of arm a
μ_a: expected value of X_a
$x_{a,t}$: realization of X_a at time t
α, β: parameters of Beta distribution



- each arm has a beta distribution
- we drawn a sample
- we solve the associated combinatorial problem
- find the superarm $C(\underline{a}) = \{C_1, C_4, C_7\}$
- pull the arms simultaneously
- update the rewards of all these arms

- **Define the regret in case of combinatorial bandit problems and compare with non combinatorial bandit problems**

* Regret of combinatorial bandit problems $R_T(TS) \in \mathcal{O}\left(\frac{|A| \log T}{\Delta_{min}}\right)$

- increases linearly as the number of arms $|A|$ increases
- increases logarithmically as T increases

* Regret of non combinatorial bandit problems is a lower bound which

- increases linearly as the number of arms $|A|$
- increases logarithmically as T increases

⇒ combinatorial bandit problems have in general a lower regret → better!

Question 5

- **Define what an online matching problem is and how it distinguishes from the non-online case**

A matching problem is said to be "online" if the graph is discovered during time (also if it's weighted) (in bipartite graph one side of nodes is known a priori)

At each round a single node enters the problem and its edges are discovered and we have to decide to match or not the new node.

A non-online matching problem is a completely known graph a priori

- **Define the competitive factor of an online problem**

online matchings are inefficient, because the optimal choice of a possible matching depends on what happens in the future. In order to understand if the algorithm is doing the best, we compare the number of matchings with the number of matchings of the clairvoyant solution, where the clairvoyant solution is the best solution we have with complete information about the problem

$$\alpha_A = \min_P \frac{\Gamma_A(P)}{\Gamma^*(P)}$$

α_A: competitive factor of the algorithm

$\Gamma_A(P)$: value provided by the online algorithm

$\Gamma^*(P)$: value provided by the clairvoyant algorithm

it can't be larger than 1 because the clairvoyant solution is the best

- No deterministic algorithm can have a competitive factor larger than 1/2 in online bipartite matching problems
- No randomized algorithms can have an expected competitive factor larger than $\frac{e-1}{e}$ in online bipartite matching problems

- **Show that a basic online matching problem does not admit any deterministic algorithm with competitive factor larger than 1/2**

match a node with an arbitrary unmatched node → algorithm has $\alpha_A = 1/2$ (the best we can do)

The proof relies on 2 steps:

1. The cardinality of any maximal matching is at least 1/2 of the cardinality of the maximum matching

   Assume all weights $w_i = 1$ and a bipartite graph.

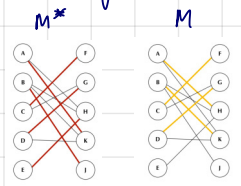
   A. Given M^* (maximum matching) and M (maximal matching)

   B. $\forall$ edge $e = (u, v)$ in M^*, either u or v is matched in M, otherwise M is not maximal (we can add e)

   C. So the number of matched nodes in M is not smaller than $|M^*|$ = # edges

   D. The number of matched nodes in M^* is $2|M^*|$

   E. The number of edges in M is at least 1/2 the number of edges in M^* : $|M| \geq 1/2 |M^*|$



2. This greedy algorithm always find a maximal matching

   A. Given M (maximal)

   B. $\forall$ edge $e = (u, v)$ not in M, consider the round in which v arrived

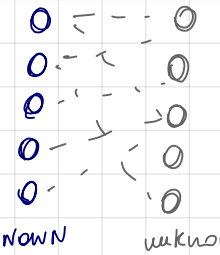
   C. { v has been matched to some other node instead of u
      { v is not matched since all other nodes are matched

   D. In both cases (u, v) cannot be added to M, not being feasible

- **Describe a greedy algorithm for a basic online matching problem with $\alpha_A = \frac{e-1}{e}$**

1. generate randomly an ordering over the nodes that are initially available
2. match a node with the first unmatched node in the ordering that we generated initially

   → competitive factor = $\frac{e-1}{e} > 1/2$



~ Randomized algorithm (Ranking) ~

Question 7

- Show when both sides of a bipartite matching problem enter dynamically, there not online algorithm with strictly positive competitive factor

There are some istances where the competitive factor is 0, so there's no deterministic algorithm returning any strictly competitive factor

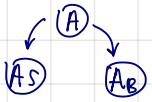
1. A enters the problem (A)
2. B enters the problem : do we match A and B ?
   - ↳ match: (A) —1— (B) ···>>1··· (E) $\quad \frac{APT}{OPT} = \frac{1}{>>1} \to 0$ $\quad \frac{\text{match A-B}}{\text{best solution B-E}}$
   - ↳ don't match (B) $\quad \frac{APT}{OPT} = \frac{0}{1} = 0$ $\quad \frac{\text{don't match}}{\text{best solution A-B}}$

- Describe the functioning of the Postponed Dynamic Deferred Acceptance and report an example

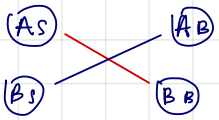
There's a randomized algorithm with an expected competitive factor of 1/4 when the waiting time of every node is the same (k)

→ every time a node arrives, it stays the same number of rounds and it leaves (when it's critical or at its deadline)

1. When the node arrives: the algorithm generates the node as seller and buyer



2. If a node has to leave:
   a. we match it
   b. it leaves the problem



A. Select A as seller ($P = 1/2$)
the matched B is labelled as buyer
match A_S with B_B because it's the best matching
remove virtual nodes that are different from labelling

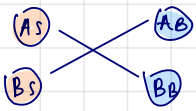
B. Select A as buyer ($P = 1/2$)
the matched B is labelled as seller
since A_B was optimally matched with another node, then A leaves the problem without being matched

→ worst case example (A) —1— (B) —>>1— (E)

1. A enters the problem:
2. B enters the problem:

3. A reached its deadline → we have to match it with B or it leaves the problem

A. A is seller → B is buyer (AS —1— BB)
C enters the problem but it's unmatched



B. A is buyer → B is seller (AB BS ⇒ BS ~~BB~~)
A is not matched with B and it leaves the problem → matching of value 1

↳ 3. C enters the problem
B is seller → C is buyer (BS —>>1— CB) → matching of value >>1

→ competitive factor is 1/2 in both cases

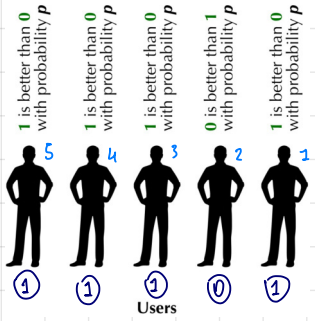
SOCIAL INFLUENCE MAXIMIZATION

Question 1

- **Provide the model of information cascade**

We are given a set of users and every user has to take an action among a given set of actions.

1. At the beginning all users don't know which action is the best
2. Each user at the beginning receives a signal saying which action is the best with a probability $p > 1/2$
3. Apart from the first user, each user builds his belief upon what previous users chose.



user 1: his belief depends on the received signal
→ he chooses action 1

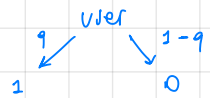
user 2: he sees user 1 chose 1, he received also the signal telling that action 0 is the best. The two beliefs nullify: ties are broken in favour of the received signal
→ he chooses action 0

⇒ users' belief depends on the difference between the number of observed 1 and 0.

- **Show that asymptotically the probability that an information cascade happens is 1.**

If two consecutive users select the same action, then they generate an information cascade because all other users will select the same action regardless their signal.

The probability to have a cascade (p) is larger than the probability to have 3 consecutive 1-signals or 0-signals ($\bar{p}$)



$$p > \bar{p} = q^3 + (1-q)^3$$

prob. to have 3 1-signal — prob. to have 3 0-signal

if the time horizon $T \to \infty$, then we have the certainty to reach a cascade

$$\boxed{1 - (1-\bar{p})^{T/3} \longrightarrow 1}$$

- **Discuss the mathematics behind information cascade**

PRIOR: $\mathbb{P}(1 \succ 0) = \mathbb{P}(0 \succ 1) = 1/2$ probability that action 1 is better than action 0

SIGNAL: $\mathbb{P}(\{1\} \mid 1 \succ 0) = \mathbb{P}(\{0\} \mid 0 \succ 1) = p > 1/2$ probability that user 2 receives signal 1 when action 1 is better than action 0

$\mathbb{P}(\{0\} \mid 1 \succ 0) = \mathbb{P}(\{1\} \mid 0 \succ 1) = 1-p < 1/2$ probability that user 2 receives signal 0 when action 0 is better than action 1

Bayesian Rule: $\mathbb{P}(1 \succ 0 \mid \{1\}) = \dfrac{\mathbb{P}(\{1\} \mid 1 \succ 0)\,\mathbb{P}(1 \succ 0)}{\mathbb{P}(\{1\} \mid 1 \succ 0)\,\mathbb{P}(1 \succ 0) + \mathbb{P}(\{1\} \mid 0 \succ 1)\,\mathbb{P}(0 \succ 1)} = p$ update

Question 2

- **Belief update in information cascades**

$$\mathbb{P}(1 \succ 0 \mid (n_0, n_1)) = \frac{\mathbb{P}((n_0, n_1) \mid 1 \succ 0)\,\mathbb{P}(1 \succ 0)}{\mathbb{P}((n_0, n_1) \mid 1 \succ 0)\,\mathbb{P}(1 \succ 0) + \mathbb{P}((n_0, n_1) \mid 0 \succ 1)\,\mathbb{P}(0 \succ 1)}$$

$$= \frac{p^{n_1}(1-p)^{n_0}\,1/2}{p^{n_1}(1-p)^{n_0}\,1/2 + p^{n_0}(1-p)^{n_1}\,1/2} =$$

$$= \frac{p^{m_1}(1-p)^{n_0}}{p^{m_1}(1-p)^{n_0}\left(1 + p^{n_0-m_1}(1-p)^{m_1-n_0}\right)} = \frac{1}{1 + \frac{(1-p)^{\Delta}}{p^{\Delta}}} > 1/2$$

and if $\Delta = m_1 - n_0 \to +\infty \Rightarrow \mathbb{P} \to 1$ many users select 1 $\Rightarrow$ cascade

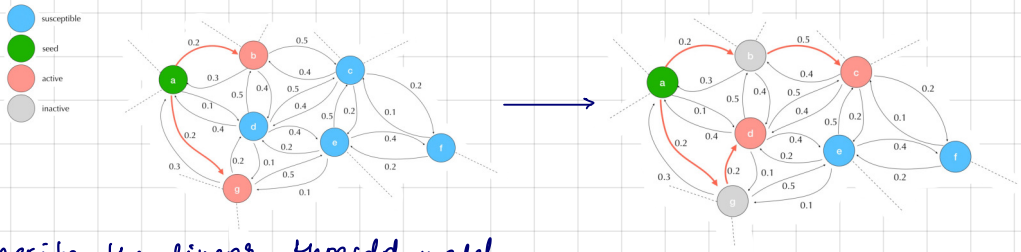
Question 3

- **Describe the independent cascade model**

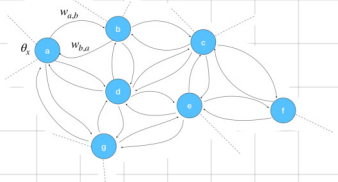
We are given a set of nodes and a weighted direct graph. Each weight is $p_{a,b} \in [0,1]$ and corresponds to the probability that node a effects node b.

1. every node belongs to one of these states:
   - susceptible → the node can be activated by the influence of some neighbourhood
   - active → the node is activated by a neighbourhood
   - inactive → the node was previously activated
2. Time is discrete
3. If a node becomes "active" at time t, then at time $t+1$ it can activate a neighbour with a probability $p_{a,b}$
4. If node b is not activated by node a, then node a cannot activate node b in the future
5. If a node activates at time t, at time $t+1$ it becomes inactive

And so we define a set of seeds in order to generate a cascade



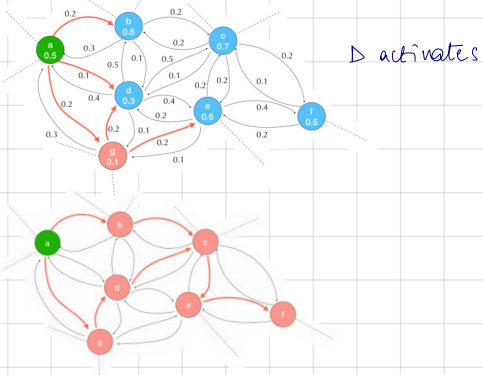
- **Describe the linear threshold model**



We are given a weighted direct graph where
$\vartheta_x \in [0,1]$ is the threshold
$w_{ab} \in [0,1]$
$\sum_a w_{ab} \leq 1$

The activation goes as follows:

1. once a node is active, it remains active in the future
2. A node activates if $\sum_{y:\, y \text{ is active}} w_{yx} \geq \vartheta_x$



- **Discuss what a live-edge graph is**

Given a direct weighted graph and a seed, a LIVE EDGE GRAPH is the subgraph composed by all nodes that can be reached by the seed (so the ones which can be active in the future)

Question 4

- **Provide a formal model for the influence maximization problem**

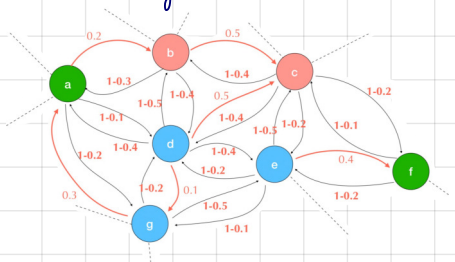
Considering the independence cascade scenario, the influence maximization problem can be described as follows:

- Input: network, influence probabilities, budget → in terms of number of nodes we can buy
- Actions: select simultaneously a subset of nodes (seeds) with cardinality not larger than the budget

→ GOAL: find the best subset of nodes subject to budget constraints to maximize the expected number of nodes which have been activated during the cascade

- **Describe an exact algorithm to compute the expected number of nodes influenced by a set of seeds**

1. Call E the set of edges $(e_1 \dots e_m)$
2. Select a subset of edges $x = (x_1 \dots x_m)$ where $x_i = 1$ if edge e_i is present, 0 otherwise
3. Enumerate every x (they're 2^m) — x : live-edge graph
   ↳ the enumeration can be done by using a tree with branching factor 2
4. $\forall x$ find the set of nodes containing nodes which have been activated during the cascade
5. $\forall x$ compute the corresponding probability (probability to reach all nodes in the set built at step 4) which corresponds to the product of all above probabilities
6. The activation probability of a node is the sum of the probabilities of x, with a path connecting the node with source seed



step 4: {b, c}
step 5: {b: 0.2, c: 0.2·0.5}
step 6: consider all live-edges graphs and sum all the probabilities

- **Describe an approximate algorithm to compute the expected number of nodes influenced by a set of seeds**

Monte Carlo sampling algorithm:

1. For each node assign $z_i = 0$ (z_i corresponds to the number of times node i is activated)
2. Generate randomly a live-edge graph according to the probability of each edge
3. For each node which is active → $z_i = z_i + 1$
   [checking whether a node is active can be done by using depth-first tree excluding nodes previously visited]
   → repeat 2. and 3. for k times
4. For each node compute z_i/k → frequency of a node which has been activated in the simulations

- **Provide a theoretical bound of the approximate algorithm**

By using Monte Carlo sampling, we may be asked how many repetitions (k) can we make_ we provide the following upper bound_ with a probability of at least $1-\delta$, the estimated activation probability of each node is subject to an additive error of $\pm \varepsilon n$ when the number of repetitions is

$$R = \vartheta\left(\frac{1}{\varepsilon^2}\log(|S|)\log\left(\frac{1}{\delta}\right)\right)$$

n = # nodes
s = # seeds

Question 5

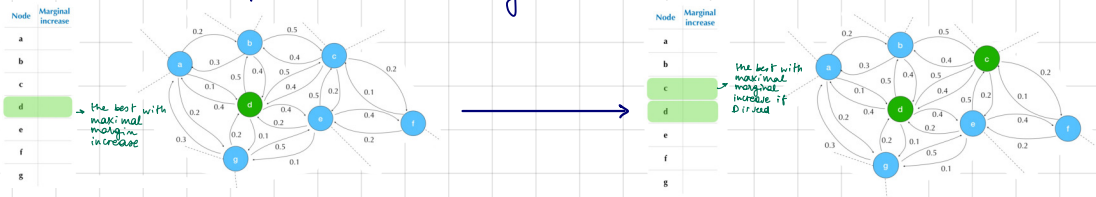
- **Describe the exact algorithm to maximize the social influence given a budget constraint**

The IM problem cannot be solved exactly in polynomial time, even using Monte Carlo to approximate the influence.
Given a budget, the network and influence probabilities, the basic exact algorithm requires to enumerate all possible subsets of k nodes (seeds) and for each subset we evaluate the expected number of active nodes and then choose the one which is the maximum

- **Describe an approximation algorithm to maximize the social influence given a budget constraint**

A simple greedy algorithm provides an approximation of $1 - 1/e \simeq 0.63$.

1. For each node that is not a seed, we need to evaluate the marginal increase in the objective function if that node is not a seed
2. Select the node for which the marginal increase is maximized and add it to the set of seeds



repeat until we have budget

The greedy algorithm works incrementally (adding seed by seed) and it decides the set of seeds, it doesn't simulate the nodes become seeds incrementally

- **Describe the theoretical guarantee of this approximation algorithm**

The approximation bound follows from the sub-modularity property

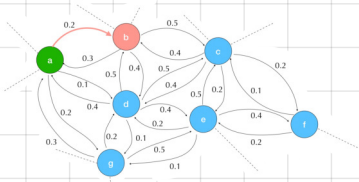
$\forall X \subseteq Y,\ x \notin Y : f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$

$\Rightarrow$ adding a seed to a set of seeds gives a marginal increase in the utility function larger than adding the same node to a strictly larger set of seeds

Question 6

- **Discuss what the main learning issues are in influence maximization and provide a learning algorithm for each case**

we are given a network and weights are unknown



- 1° scenario: FULLY OBSERVABLE EDGES

For each edge in the network we can observe whether the edge activated or not (we can see the node which has been activated and the node which activated)

→ it's a standard combinatorial MAB problem

- every edge ~ Be(p) with p unknown
- For each edge we can draw a sample from a Beta (with TS) or use an upper confidence bound (UCB1)
- Once we have a value for each edge, we use a greedy algorithm to find the optimal solution

- 2° scenario: PARTIALLY OBSERVABLE EDGES

For each edge in the network we can observe the activation of a small portion of edges. since collecting a huge number of samples is impossible, we want to exploit the information on the observable edges to infer the probabilities of the non observable edges (by reusing the information collected in some parts of the network for other parts)

This idea is built on some structure about the probabilities of the edges:

- features $\mathcal{F} = \{1, 2 \dots l\}$
- values of the features: $x_{ij} \in [0, 1]$ i = edge, j = feature [$x_{ij} = 1$: feature j is important for edge i]
- parameters of the features $\vartheta_j \in [0, 1]$ j = feature

$\Rightarrow p_i = \sum_{j=1}^{l} \vartheta_j x_{ij}$ probability of edge i

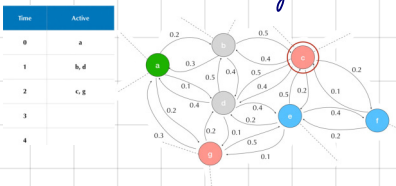
Example: if the features are the interest of users, a value of 1 for a feature means that both users share that interest

→ algorithm:

1. parameters of features are estimated by using linear regression (features' value is known)
2. Given the values of the probabilities, a term is added to obtain an upper confidence bound (UCB1 idea)
3. The optimization is performed by using a greedy algorithm

- 3° scenario: NON OBSERVABLE EDGES

For each edge in the network we can observe only the activation of nodes without observing the activation of the edges



→ In order to solve the problem we need to observe the evolution of the cascade by providing additional information by taking trace of the active nodes

→ When a node could have been activated by multiple nodes, Maximum likelihood techniques are used to estimate which edge is the most which activated that node or alternative frequentist techniques should be used