Questions Answers

Raffaello Fornasiere

July 3, 2022

Chapter 1

Price Discrimination

1.1 Matching Problem

1.1.1 Provide mathematical formulation of a matching problem

Given a graph

$$G(V,E) \quad \text{where:} \tag{1.1}$$
$$V = \text{set of nodes (i.e. set of numbers)} \tag{1.2}$$
$$E = \{\{x,y\} \mid x,y \in V \wedge x \neq y\} \tag{1.3}$$
$$\text{a matching is a set defined as:} \tag{1.4}$$
$$M = \{e_1 \in E | \forall e_2 in E \wedge e_1 = e_2 \Rightarrow \{e_{1_x}, e_{1_y}\} \cap \{e_{2_x}, e_{2_y}\} = \emptyset\} \tag{1.5}$$

In words: a matching is a subset of edges that don't have any node in common. A matching problem is the problem of finding the best matching given a graph.

1.1.2 Describe what alternating and augmenti paths are

W.r.t. the above definitions we have that

- An alternating path is a path inside the graph composed by edges that are in M and edges that are not.
- An augmenting path is an alternating path where the first and the last node are not in the matching set

1.1.3 Describe the functioning of the alternating-path algorithms for bipartite graphs

In bipartite graphs alternating-path algorithms aim to find best solutions to matching problems. They rely on finding an augmenting path that later is converted into a matching of higher cardinality of the one given.

1.1.4 Describe the functioning of the alternating-path algorithms for arbitrary graphs

Same as previous?

1.2 Alternating-path algorithms for bipartite graphs

1.2.1 Given a bipartite graph and a matching, apply alternating-path algorithm to find all shortes augmenting paths, if any

Given the graph below we want to expand all the unmatched nodes in order to find a beter matching. We aim to find shortest augmenting paths so we'll apply BFS to expand nodes in the following way:

- starting from one unmatched node we expand it through its unexpanded edges

  - if expand following the alternating policy (if we find a matched node we follow expand also through the match edge). We mark the final node as even

  - if we find a non mathced node we stop and we add it to the nodes to be expanded (should we mark it as even?)

- we mark that node as expanded
- we continue the expanding procedure
- if we find an even node of another subtree we found an augmen path and we can improve the matching
- if we find an odd node we can stop the search (nothing happens)

1.3 Alternating-path algorithms for arbitrary graphs

1.3.1 Given a arbitrary graph and a matching, apply alternating-path algorithm to find all shortes augmenting paths, if any

The procedure is the same as bipartite but we have the additional case in which we could fall into node of the same subgraph: in this case we found a blossom. At this point we can

collapse the blossom and consider it as a single node and then continue the search. At the end we'll expand again the blossom and we'll have found an augmenting path if we've found it with collapsed blossom. In other words the blossom does not affect the search of augmen paths.

1.4 Combinatorial bandits

1.4.1 Define what a combinatorial bandit problem is.

A combinatorial bandit problem is a problem where arms are correlated each other and we can pull more then one arm per turn, called super arm, and arms are regolated by a combinatorial constraint (i.e. knapsack, Independend set, matching)

1.4.2 Describe how a matching problem can be formulates as a combinatorial bandit problem

A matching problem can be seen as CMAB where arms are the nodes of a given graphs and a superarm is a matching. We wanto to maximize the matching but keeping the constraint about the matching policies.

1.4.3 Describe the Combinatorial Thompson Sampling algorithm.

In CTS we have distributions over nodes. Each round we sample the nodes to compose a super arm. We apply an optimizer to make the combinatorial problem be maximized/respected. Then we play the arm and update the distributions.

1.4.4 Define the regret in the case of combinatorial bandit problem and discuss the differences between the regret bounds of the non-combinatorial and combinatorial cases.

(How the hell is calculated the regret?)
Standard regret bound is

$$R_T \leq O\left(\sum_{a_i in A,\ a_i \neq a^*} \frac{\Lambda_i}{KL(R(a_i), R(a_*))} \cdot (\log T + \log\log T)\right) \tag{1.6}$$

In the CMAB instead we have:

$$R_T \leq O\left(|A| \frac{\log T}{\Lambda_{min}}\right) \tag{1.7}$$

If we use non exact optimization algorithm the regret increases

1.5 Online Matching

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

An online matching problem is a matching problem that is not fully observable. In particular we can observe only a subset of nodes at each time step t.

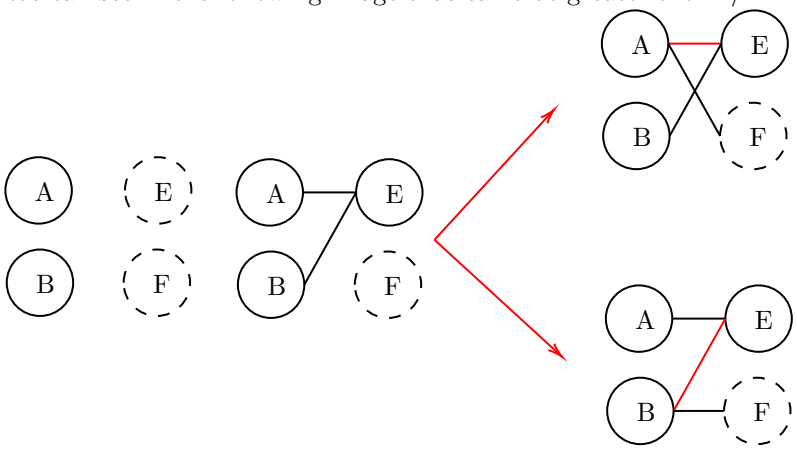
1.5.2 Define the competitive factor of an online problem.

The competitive factor is defined as:

$$C = min_P \frac{\Gamma(P)}{\Gamma_{calirvoyant}(P)} \tag{1.8}$$

1.5.3 Show that a basic online matching problem (in bipartite graphs, with the nodes of only side entering dynamically) does not admit any deterministic algorithm with competitive factor larger than 1/2.

We can see in the following image that can't be greater then 1/2

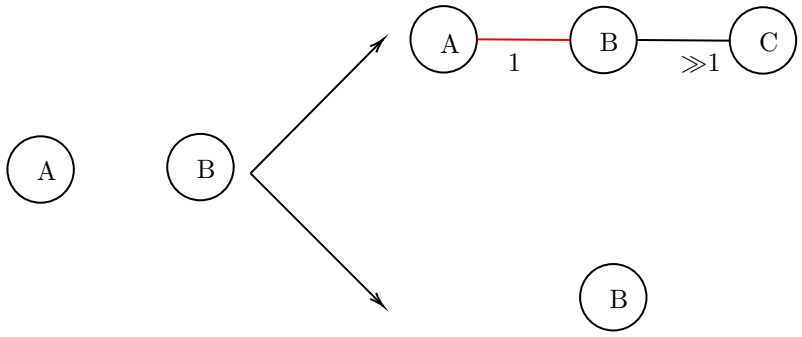


1.5.4 Describe a greedy algorithm for a basic online matching problem with competitive factor 1 - 1/e

A greedy algorithm with $1 - 1/e$ is simply an algorithm that at each timestep matches unmached nodes with the first unmached node

1.6 General online matching

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



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