1 Introduction

Graph problems constitute a rich and fundamental area in the study of online algorithms. We have previously visited the k-server problem as a graph-related problem. In the k-server problem, the underlying graph (metric) is known to online algorithms and the sequence of requests to vertices is online. In most online graph problems, however, the graph itself is revealed in an online manner. As you probably seen in other courses, there is an array of graph problems (such as variants of graph coloring, independent set, dominating set, etc.) which are NP-hard in the offline setting. Recall that in the online setting, the complexity of algorithms is not an issue. Instead, it is the online constraint that restricts the algorithm. In this section, we briefly review two graph problems in the online setting.

2 Online edge coloring

In the offline edge-coloring problem, the input is an undirected, unweighted, simple graph $G = (V,E)$, and the goal is to color edges of the graph so that any two adjacent edge (any two edge with one shared endpoints), have different colors. The objective is to minimize the number of colors used to color the edges. Figure 1 shows an example of edge coloring of a graph.

Figure 1: An example of edge coloring
Figure 2: (left) The coloring of greedy algorithm; indices of edges in the sequence are indicated with numbers. The greedy algorithm uses 5 colors. (right) The optimal algorithm uses 4 colors.

A classic theorem by Vadim Vizing in 1964 indicates that edges of a graph of maximum degree \( \Delta \) can be colored using either \( \Delta \) or \( \Delta + 1 \) colors. Recall that the maximum degree of a graph is the maximum degree of any of its degree. Note that if a vertex has degree \( \Delta \), clearly \( \Delta \) colors are required. Hence, the minimum number of colors for edge-coloring is either \( \Delta \) or \( \Delta + 1 \). Interestingly, given a graph of max-degree \( \Delta \), it is NP-hard to decide whether \( \Delta \) or \( \Delta + 1 \) colors are needed. There is a polynomial-time algorithm, however, that colors edges using \( \Delta + 1 \) colors (see, e.g., [1]).

We consider the online edge-coloring under the edge-arrival model. Under this model, the edges of the graph ‘arrive’ one by one. When an edge arrives, its endpoint vertices are also revealed to the online algorithm (in case they have not been revealed through other edges). When an edge arrives, the online algorithms has to color it. In doing so, the algorithm is oblivious to the future edges.

Consider a greedy family of algorithm which color each edge using one of the existing colors if possible. If an incoming edge is adjacent to edges with all existing colors, the greedy algorithm uses a new color for the edge. Figure 2 contrast a greedy coloring with an optimal coloring of a graph.

Theorem 1. Any greedy algorithm has a competitive ratio of at most 2.

Proof. Recall that for a graph of max-degree \( \Delta \), the cost of \( \text{Opt} \) is at least \( \Delta \). We prove that the number of colors used by the greedy algorithm is at most \( 2\Delta - 1 \). Consider the edge \( e \) which demands the last color of the greedy algorithm. Since the graph has degree at most \( \Delta \), \( e \) is adjacent to at most \( \Delta - 1 \) edges on any of its two endpoints. So, there are at most \( 2\Delta - 2 \) edges adjacent to \( e \). We know that all colors used by the greedy algorithm are used for edges adjacent to \( e \) (otherwise greedy used a missing color for \( e \)). Even if all edges adjacent to receive a separate color, at most \( 2\Delta - 2 \) colors have been used before the arrival of \( e \). Adding the color used for \( e \), there will be \( 2\Delta - 1 \) colors used by the greedy algorithm. Hence, the competitive ratio of the algorithm will be at most \( \frac{2\Delta - 1}{\Delta} < 2 \).

So, regardless of what the graph is, the competitive ratio of any greedy algorithm is at most 2. The following theorem implies that in fact, greedy algorithms are the optimal algorithms for the online edge-coloring problem.

Theorem 2. No online algorithm can have a competitive ratio better than 2.

Proof. Consider a star of degree \( \Delta - 1 \), that is, a tree with \( \Delta - 1 \) leaves and one `center’ connected to all of them. Given \( 2\Delta \) colors, there are \( K = \Delta \left( \frac{2\Delta}{\Delta} \right) \) ways to color a star.

Consider an adversarial input that starts with \( X = (\Delta + 1)K + 1 \) stars of degree \( \Delta - 1 \). Since there are \( K \) ways to color each star, sending \( X \) starts ensures that at least \( \Delta \) of these stars have the same color (pigeon-hole principle). The adversary continues the sequence by sending \( \Delta \) edges which find one of their endpoints in the center of these \( \Delta \) stars (with the same color) while sharing their
other endpoint at a new vertex $c$ (see Figure 3). Any of these $\Delta$ edges requires a different color as they share an endpoint; moreover, they cannot use any of the $\Delta - 1$ colors used for the stars they are connected to. Hence, the online algorithm requires at least $\Delta - 1$ colors (for coloring the stars) plus $\Delta$ colors (for coloring the edges at the end of the sequence). In total, the algorithm’s cost is $2\Delta - 1$. The optimal algorithm, on the other hand, uses at most $\Delta + 1$ colors by the Vizing theorem (in fact, for trees, $\Delta$ colors is sufficient). Hence, the ratio between the cost of the algorithm and $\text{OPT}$ converges to 2.

The above results show that Greedy algorithms are the optimal deterministic edge-coloring algorithms. In fact, Greedy is the best strategy even if you allow randomization, that is, if the adversary generates the input and the online algorithm uses the randomization, we cannot achieve a competitive ratio better than 2 [2]. If edges arrive in a random order (as opposed to adversarial order), however, it is possible to achieve a competitive ratio of 1.43 [3].

3 Online bipartite matching

A bipartite graph is a graph whose vertices can be partitioned into two independent sets. In other words, you can draw graph with a subset of vertices on the ‘left’ and the remaining vertices on the right such that all edges find endpoints on the two sides. A matching in a graph is a subset of edges such that no two edge are adjacent (share an endpoint). In the matching problem, the goal is to find a matching of maximum size. It is well-known that finding a maximum matching in a bipartite graph can be done in polynomial time (see, e.g., [4]).

In the online setting, the vertices on the left are given and vertices on the right arrive one by one. Upon arrival of a vertex, all edges adjacent to that vertex are also revealed. After a vertex is arrived, the online algorithm has the option to match the vertex with of its neighbors on the left (assuming they are not connected to another vertex). Decisions of the algorithm are ‘online’ in the sense that the algorithm has no a priori information on how the graph will evolve, and it cannot change its previous decisions. The objective is to maximize the number of matched pairs of vertices. Note that this problem is a maximization problem. To be consistent with previous minimization problems that we saw throughout the course, we define the competitive ratio of a maximization problem as the maximum ratio between the benefit of the optimal algorithm and that of the online algorithm.
algorithm for the same input. This way, similar to minimization problems, the ratio is always larger than or equal to 1, and the goal of an online algorithm is to minimize the competitive ratio.

Consider a family of greedy algorithms that, upon arrival of a vertex, matches with any of the candidate neighbors on the left. Note that greedy algorithms achieve ‘maximal matchings’ in which at least one endpoint of any edge in the graph is involved in a matching; this is because if two endpoints of an edge are unmatched, the greedy algorithm would have matched them on the arrival of the vertex on the right. In what follows, we prove an upper bound for competitive ratio of a the greedy algorithms:

**Theorem 3.** Competitive ratio of any greedy algorithm is at most 2.

**Proof.** Let \( n \) denote the number of vertices matched by \( \text{Opt} \) on the left, that is, the benefit of \( \text{Opt} \) is \( n \). These \( n \) vertices have \( n \) ‘partners’ on the right where by partner we mean the matched vertex on the right in the optimal matching. Assume among the \( n \) vertices on left that \( \text{Opt} \) matches, \( X \) of them are also matched by the greedy algorithm and the remaining \( n - X \) vertices are unmatched. Since the matching of the greedy algorithm is maximal, the partners of these \( n - X \) vertices are matched by the greedy algorithm. In summary, greedy matches \( X \) vertices on the left and \( n - X \) vertices on the right. So, the benefit of the greedy algorithm is \( \max\{X, n - X\} \geq n/2 \). In summary, \( \text{Opt} \) has benefit of \( n \) and the greedy algorithm has benefit of at least \( n/2 \).

Next theorem show that the greedy algorithm is the optimal deterministic algorithm for bipartite matching.

**Theorem 4.** No deterministic online algorithm can have a competitive ratio better than 2.

**Proof.** Assume vertices on the left are labelled as \( L_1, \ldots, L_n \). Similarly, vertices on the right arrive as \( \langle R_1, R_2, \ldots, R_n \rangle \). Adversary creates the input as follows. When \( R_1 \) arrives, it is only connected to \( L_1 \) and \( L_2 \). Now, if the online algorithm matches \( R_1 \) with \( L_1 \), adversary ensures \( R_2 \) is only connected to \( L_1 \); in this case, the optimal algorithm’s matching is \((L_1, R_2), (L_2, R_1)\), while algorithm has only \((L_1, R_1)\). Similarly, if the online algorithm matches \( R_1 \) with \( L_2 \), then \( R_2 \) will only be connected to \( L_2 \); in this case, the matching of \( \text{Opt} \) will be \((L_1, R_1), (L_2, R_2)\) while the algorithm only has \((L_1, R_2)\). Repeating this argument for other vertices, the adversary ensures that for any edge that the algorithm matches, the optimal algorithm matches two edges.

The above results show that the competitive ratio of the best existing deterministic algorithm is 2 (which is given by greedy algorithms). In the case of bipartite matching, however, randomization can help to achieve better competitive ratios.

Consider a randomized algorithm naked Rank which works as follows. Initially, vertices on the left are randomly permuted and each is given a random, unique index between 1 and \( n \). Now, upon arrival of a vertex \( v \) on the right, Rank matches \( v \) with the vertex with minimum index among its unmatched neighbors on the left (see Figure 4. Note that Rank is still a greedy algorithm. But it uses randomization to avoid the worst-case scenarios. In a famous result, Karp et al. [?] proved that Rank has a competitive ratio of \( \frac{e}{e-1} \approx 1.58 \), and in fact it is the best that a randomized algorithm can achieve. So, the status of the problem is closed in the general case for both deterministic and randomized setting. However, there are still many variants of the problem which are being actively studied (e.g., what is the best competitive ratio that one can achieve when vertices arrive in random order?).
Figure 4: The matching of the Rank algorithm. Number on the left indicate the indices in the initial random permutation.

References


