CSC2421 Topics in Algorithms: Online and Other Myopic Algorithms Fall 2019

Allan Borodin

October 2, 2019

Announcements

- There is no lecture on Wednesday, October 9
- This will be the last week that I plan to give full lectures. Then starting Wednesday October 16, I would like to hold the class as a 1/2 lecture, and 1/2 reading course reporting.
- I would therefore like everyone, taking the course for credit or not for credit, to choose a chapter for their part of the reading course. As I said, we would use 1/2 of the time to give short presentations on the reading to date.
- At the end of the term, we can summarize all the readings.

Week 4 Agenda

This week we will complete some discussion of topics in chapter 4. Then we will go on to topics in chapters 5 and (maybe) 6.

More specificially, we will discuss:

- Load balancing in circuit routing.
- Input models for graphs
- One-sided input model for bipartite graphs
- Online inapproximations for various graph problems
- Online bipartite matching; the Ranking algorithm and its online optimality for adversarial inputs
- Online graph coloring
  - Arbitrary graphs
  - Bipartite graphs
  - d-inductive graphs

  Note: In reading some relevant papers, there has been some significant progress in randomized lower bounds for graph coloring. I can still see some research questions that I think are approachable.

Online graph problems

Chapter 5 studies graph problems where now the input graph is being revealed online. (This is in contrast to say the virtual circuit routing problem where the graph is known in advance.)

There are two issues to keep in mind for the study of online graph problems

1. Many graph problems are know to be NP-hard to approximate (to obtain anything much better than a trivial approximation). However, because we ignore complexity constraints, this does not imply that there is necessarily a corresponding *online hardness of approximation*.
2. In the problems we have previously studied there was a natural input representation. But for graphs there are alternatives and we need to be precise as to how graphs are represented.

Graph online input models

Let $G = (V, E)$ be a graph. In an online framework, either the vertices V or the edges E will be the online inputs. That is, we assume that the sets V and E are totally ordered and use the symbol $\prec$ to indicate the order. For example, if $\prec$ is a total order defined on V, then for two vertices $u, v \in V$ we write $u \prec v$ to indicate that the input item associated with u is revealed before the input item associated with v.

There are four input models for online graph problems.

The Edge Model (EM). Each input item is an edge given by its two endpoints. Thus, the input graph G is presented as a sequence of edges $\{u_1, v_1\}, \{u_2, v_2\}, \ldots, \{u_n, v_n\}$. The algorithm does not know V a priori and it is understood that $V = \bigcup_{i=1}^{n} \{u_i, v_i\}$.

The vertex adjacency model, past history

The Vertex Adjacency Model, Past History (VAM-PH). Each input item consists of a vertex together with a set of neighbors among the vertices that have appeared before. Suppose that input items are revealed in the order given by $\prec$ on V, then each new input item can be described as follows:

$$(v; N(v) \cap \{u : u \prec v\}).$$

We note that the model that makes the most sense for online applications is often (but not always) the VAM-PH model, especially when a graph problem requires decisions about nodes. Suppose we want to apply the theory of online algorithms to graphs that grow dynamically, for example, graphs arising out of social networks. In those real-life scenarios, consider the graph growing because of new users joining the network (as opposed to new friendships being established between existing users). When a new user joins the network, they establish connections with friends that are already using the social network.

The edge adjacency model

The Edge Adjacency Model (EAM) In this model, we associate with each edge $e = \{u, v\}$ a label ℓ_e. Note that the label does not carry the information about the endpoints of an edge e. An input item in this model consists of a vertex together with a set of labels of edges that are incident on that vertex. Formally, each input item is of the form:

$$(v; \{\ell_e : v \in e\}).$$

Thus, if a neighbor $u \sim v$ has appeared before v, i.e., $u \prec v$, then an online algorithm can recover the information that $u \sim v$, since both data items corresponding to u and v will contain the same label $\ell_{\{u,v\}}$. If a neighbor $u \sim v$ appears after v, i.e., $v \prec u$, then an online algorithm only knows that some neighbor is going to appear later in the input sequence, but it does not know the identity of u at the time of processing v.

The vertex adjacency model, full information

The Vertex Adjacency Model, Full Information (VAM-FI). Each input item consists of a vertex together with the set of all its neighbors (even the neighbors that have not appeared before); that is, an input item is of the form:

$$(v; N(v)) .$$

If in an application all nodes (and their names) are known a-priori then this is the appropriate model. But as we have said , it is not a realistic online model for many applications.

The relative representation power of these input models

For two models M1 and M2 we use the notation M2$\leq$M1 to indicate that any algorithm that works in model M1 can be converted into an algorithm that works in model M2 without any deterioration in the performance, as measured by the worst-case competitive ratio. Intuitively, M2$\leq$M1 means that M1 is a harder model for online algorithms, while M2 is harder for adversaries. We shall not provide a formal definiton of the reduction used to define $\leq$ as to do so would obscure the intuitive idea that will become apparent in the following informal lemma.

Lemma (informal)

We have the following relationships

$$\mathit{VAM\text{-}FI} \leq \mathit{EAM} \leq \mathit{VAM\text{-}PH} \leq \mathit{EM}$$

Note that $\leq$ is a transitive relation.

What input model is more appropriate?

In some graph problems, it might be more appropriate to make decisions about edges rather than about nodes. For instance, suppose that a solution to a problem is a matching or a path or a tree (as in the MST problem). Then such a solution can be represented as a sequence of 0/1 decisions about edges – whether or not to include an edge in a matching or path or tree. In such scenarios, the EM input model might be more natural than the VAM-PH input model.

Often, there are graph problems where either model can be considered natural. For example, consider a maximum matching problem. A matching can either be thought of as a collection of vertex-disjoint edges, which leads to 0/1 decisions in the EM model, or it can be viewed as a map going from a vertex to its (potentially matched) neighbor, which leads to decisions labelled by names of neighbors in the VAM-PH model.

Approximation hardness for online graph problems

Not surprisingly, many NP-hard to approximate graph problems remain hard to approximate in the online model even for randomized algorithms. For $n = |V|$, we have the following results for general graphs:

- $\rho_{OBL} = \Omega(n)$ for the max independent set problem.
- $\rho_{OBL} = \Omega(n)$ for the max clique problem.
- $\rho_{OBL} = \Omega(n)$ for the longest path problem.
- $\rho_{OBL} = \infty$ for the traveling salesperson problem.
- $\rho_{OBL} = \Omega(\frac{n}{\log n})$ for the (min) coloring problem.

Perhaps less predictable, $\rho_{OBL} = \Omega(n)$ for the MST (min spanning tree) problem. But the MST result is in the EM input model.

MSTs and Steiner trees in VAM-PH model

As we have commented, EM is the weakest model (i.e. provides the least amount of information in each input item) but it is sufficient for an offline optimal greedy algorithm (e.g. Kruskal's or Prim's algorithm) for the MST problem. What is possible for an online algorithm in the VAM-PH model?

I just recalled (i.e., just before the lecture) a 1991 paper by Imase and Waxman which gives an $\log n$ competitive algoriithm (and an almost matching lower bound) for the more general dynamic Steiner tree problem.

I have not carefully looked at the precise model as to what exactly is being assumed but the paper seems to show a $\log n$ competitve ratio for the MST problem when the graph distances are a metric..

Without that assumption or some other assumptions, one can show strong negative results.

The Imase and Waxman algorithm follows on the next slide.

Imase and Waxman

I will interpret this algorithm as an MST algorithm if my understading is correct.

DYNAMIC STEINER PROBLEM

$T_0 := (\{v_0\}, \varnothing)$; $S_0 := \{v_0\}$; $i = 1$
for $i \leqq K \rightarrow$
  if r_i is an *add* request $\rightarrow$
    Choose the shortest path p_i from v_i to T_{i-1}
    $T_i := T_{i-1} \cup p_i$
    $S_i := S_{i-1} \cup \{v_i\}$
  | r_i is a *remove* request $\rightarrow$
    $S_i := S_{i-1} - \{v_i\}$
    $T_i := T_{i-1}$
    do $V(T_i) - S_i$ contains node w with degree 1 $\rightarrow$
      $T_i := T_i - w$
    od
  fi
rof

FIG. 3. *Dynamic greedy algorithm* (DGA).

A special input model for bipartite graphs

In many online applications of bipartite graphs, it is only one side of the vertex partition that arrives online while the other side is known. We describe this "one-sided" bipartite model as follows. Let $G = (U, V, E)$ be a bipartite graph.

The Bipartite Vertex Arrival Model (BVAM) In this model, vertices in U arrive online in the vertex adjacency format. We assume that "offline vertices" in V are revealed in advance. Thus, V is also called the set of "known" or "offline" vertices. U is called the set of "online" vertices [1]. When an online node $u \in U$ arrives, all of its incident offline vertices, $N(u) \subseteq V$, are revealed as well.

Lemma

For bipartite graphs we have

$$BVAM \leq VAM\text{-}PH$$

[1] There is no agreed upon conventions. In some papers, it is V that is the set of online vertices. Some papers use L (left) and R (right)

More on the input models for bipartite graphs

We can always "forget" the bipartite structure of the graph and consider G as a general graph. Thus, we can always consider G to be given in one of the input models for general graphs. We couild refer to this model as the "two-sided" online bipartite model.

Most results for bipartite graphs are with respect to the onde-sided BVAM model, and we will assume this model unless otherwise stated. Due to the many online applications, the BVAM model will be discussed in various chapters of this text. Often, appllications will require that the graphs are weighted. The most general (and hardest to provide good algorithms) is when edges are weighted.

A simpler weighted model is when the offline vertices (i.e. the vertices in V) are weighted but not the edges. When just the online vertices are weighted, it is not clear what can be done in this model that can't be done for edge weights.

Maximum matching in bipartite graphs

In various settings, online matching in bipartite graphs is the basic problem. Due to the many applications, especially in online advertising, this problem has been a major reason for the continued research activity regarding the performance of online algorithms. Weighted and stochastic versions of bipartite matching are particularly important. See Chapters 10,11,12,14,16,17 in the text. In Chapter 5, we only consider the unweighted maximum matching problem in the adverarial setting.

Given a graph $G = (U, V, E)$ a subset of edges $M \subseteq E$ is called a matching if the edges in M do not share any vertices, i.e., for all $e, e' \in M$ we have $e \cap e' = \varnothing$. In the unweighted *maximum matching* problem, the goal is to find a matching that is as large as possible. We shall refer to a greedy online algorithm as one that will always match an online node u upon arrival if there are any unmatched neighbors of u.

Any greedy algorithm is at least $\frac{1}{2}$ competitive. No deterministic algorithm can be better.

A *maximal matching* $M \subseteq E$ is a matching that cannot be extended. The following fact is well known and easy to see:

Fact

Let M be a maximal matching and M^* a maximum matching. Then $|M| \geq \frac{1}{2}|M^*|$.

It is also easy to see that any determininistic online algorithm is at most $\frac{1}{2}$ competitive. Consider a simple 2×2 bipartite graph. Namely, let $U = \{u_1, u_2\}$, $V = \{v_1, v_2\}$ and $E = \{u_1, v_1), (u_1, v_2), (u_2, v_1)\}$. Upon seeing u_1, the simple greedy matches it to v_1, so when u_2 arrives it cannot be matched.

Can a randomized algorithm do better?

The most natural randomized greedy algorithm matches an arriving online node u to a random available neighbor. Consider the following family of graphs. The vertices are $U = \{u_1, \ldots, u_{2n}\}$ and $V = \{v_1, \ldots, v_{2n}\}$. Each node in the first half of U is connected to each node in the second half of V by an edge. In addition, each "parallel" edge, that is (u_i, v_i), is present. Formally, we have:
$E = \{(u_i, v_j) \mid i \in [n], j \in [n+1, \ldots, 2n]\} \cup \{(u_i, v_i) \mid i \in [2n]\}$. It can be shown that the natural randomized greedy algorithm achieves (asymptotic) competitive ratio $\frac{1}{2}$ on this instance. Thus, this algorithm provides no improvement over the simple greedy algorithm.

Perhaps surprisingly, there is another simple greedy strategy that, although somwehat less natural, provides a significant improvement over the competitive ratio of the natural randomized greedy. We alluded to this algorithm when we discussed the makespan problem in the restricted machines model.

The Ranking algorithm

We will present the Ranking algorithm that achieves competitive ratio $1 - \frac{1}{e} \approx .632$. Let $|V| = n$. (Recall we assume that V is known.) The Ranking algorithm initially picks a uniformly random permutation of V and fixes it for the whole duration of the online phase. In the online phase, when a vertex u arrives, match it with a vertex of best rank among the unmatched neighbors of u. The pseudocode of Ranking is as follows:

Algorithm 16 The Ranking algorithm for BMM.

procedure RANKING
  V – set of offline vertices
  Pick a ranking σ on vertices V *uniformly at random*
  $M \leftarrow \emptyset$
  $i \leftarrow 1$
  while $i \leq n$ **do**
    New online vertex u_i arrives according to $\prec$ together with $N(u_i)$
    if $N_c(u) \neq \emptyset$ **then** ▷ if there is an unmatched vertex in $N(u_i)$
      ▷ select the vertex of best rank in $N(u_i)$
      $v \leftarrow \arg\min\{\sigma(v) : v \in N(u)\}$
      $M \leftarrow M \cup \{(u_i, v)\}$ ▷ match u_i with v
    $i \leftarrow i + 1$

Ranking has competitive ratio $1 - \frac{1}{e}$

Consider first the positive result; i.e., $\rho_{\text{OBL}}(\mathit{Ranking}) \geq 1 - \frac{1}{e}$. Let $G = (U, V, E)$ be a given bipartite graph with $U = V = [n]$ (disjoint copies). We can assume G has a perfect matching without loss of generality.

This is a seminal result and the result is due to Karp, Vazirani and Vazirani (STOC 1990) and is sometimes called the KVV algorithm. KVV also showed that this is an optimal online algorithm; that is, no randomized algorithm can have an asymptotic ratio better than $1 - \frac{1}{e}$. It turns out that there was a subtle issue with their analysis which was later discovered and by now there are several proofs of the KVV algorithm competitive ratio. The first completely rigorous proof s due to Goel and Mehta (2008). We will see a proof based on game theory in Chapter 9. Here, we present the Birnbaum and Mathieu proof.

The Birnbaum Matheiu proof

We note that a vertex of rank t is a random variable. Let p_t denote the probability over σ that the vertex of rank t in V is matched by Ranking. We are interested in computing the expected size of the matching returned by Ranking, which is given by $\sum_{t=1}^{n} p_t$. The analysis is centered around establishing the following lemma:

Lemma

For all $t \in [n]$ we have $1 - p_t \leq (1/n) \sum_{s=1}^{t} p_s$.

We first assume the Lemma to prove that Ranking has competitive ratio $1 - \frac{1}{e}$ as follows. Observe that $p_1 = 1$, since G has a perfect matching. By induction and the lemma, it follows that $p_t \geq (1 - 1/n)(n/(n+1))^{t-1}$ for all $t \geq 2$.

Ranking analysis continued

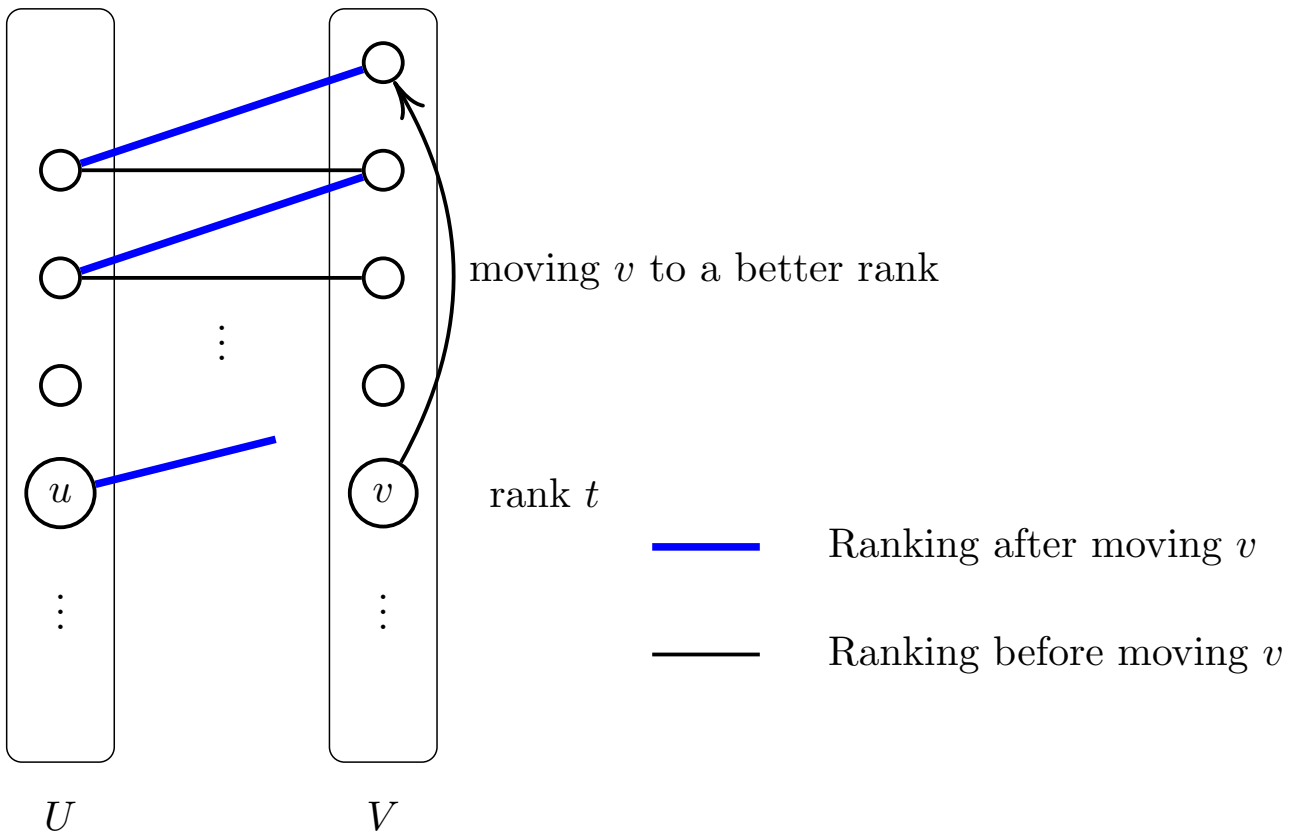
Using the lemma, we have:

$$\begin{aligned}\sum_{t=1}^{n} p_t = p_1 + \sum_{t=2}^{n} p_t &\geq \frac{1}{n} + \left(1 - \frac{1}{n}\right) \sum_{t=1}^{n} \left(\frac{n}{n+1}\right)^{t-1} \\ &\geq \frac{1}{n} + \left(1 - \frac{1}{n}\right) \frac{1 - \left(\frac{n}{n+1}\right)^{n}}{1 - \left(\frac{n}{n+1}\right)} \\ &= \frac{1}{n} + \left(n - \frac{1}{n}\right) \left[1 - \left(\frac{n}{n+1}\right)^{n}\right] \geq n \left[1 - \left(1 - \frac{1}{n+1}\right)^{n}\right],\end{aligned}$$

where the first inequality follows by representing $p_1 = 1/n + (1 - 1/n)$ and absorbing the second $(1 - \frac{1}{n})$ term into the sum, and the last inequality follows since $1/n \geq 1/n(1 - (n/(n+1))^n)$. Lastly, to conclude the proof of the Theorem, we observe that $1 - \left(1 - \frac{1}{n+1}\right)^{n} \to 1 - 1/e$ as $n \to \infty$.

Completing the proof of the Ranking ratio

It remains to prove the lemma. Let A_t denote the set of permutations such that a vertex of rank t is matched by Ranking. Let $S_{[n]}$ denote the set of all permutations $V \to V$ and define $B_t = S_{[n]} \setminus A_t$; that is, B_t is the set of permutations such that a vertex of rank t is not matched by Ranking. We shall construct an injection of the form $[n] \times B_t \to \bigcup_{i=1}^{t} A_i$. This will prove the lemma. The following figure helps in understanding the injection.



The online optimality of the Ranking algorithm

KVV also proved that no randomized algorithm can do better (asymptotically) than the Ranking algorithm. To prove the negative result, it suffices to exhibit a distribution on inputs such that the best deterministic algorithm does not perform very well with respect to that distribution. Next, we define the distribution. We fix the vertex sets to be $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. For a permutation $\pi : [n] \to [n]$ we define a graph $G_\pi := (U \times V, E_\pi)$, where $E_\pi = \{(u_i, v_j) \,|\, \pi^{-1}(j) \geq i\}$. The distribution $\mathcal{P}$ is the distribution of G_π when π is chosen uniformly at random among all possible permutations. Observe that when π is the identity permutation, G_π is the so-called triangular graph, because the biadjacency matrix where rows are indexed by U and columns are indexed by V is the upper-triangular matrix. Any other graph from the support of $\mathcal{P}$ is obtained by permuting the columns of the upper-triangular matrix. Observe that G_π has a perfect matching given by matching u_i with $v_{\pi(i)}$.

The optimality of KVV continued

Fix a deterministic algorithm A. We can assume that A is greedy. The intuition behind the negative result is that by choosing π randomly we hide the "true" identities of offline nodes. By "true" identities of offline nodes we mean their ranks in the triangular graph.

The algorithm sees an offline node v_i, but the true identity of the node is given by $v_{\pi^{-1}(i)}$. By induction, when an online node u_i arrives with k available neighbors, the true identities of those neighbors form a uniformly random set of size k from among $\{v_i, \ldots, v_n\}$. It follows that the performance of A on $\mathcal{P}$ is exactly equal to the performance of the natural randomized greedy algorithm on the triangular graph.

Online graph coloring

Given a graph $G = (V, E)$, a function $c : V \to [k]$ is called a valid k-coloring of G if for every edge $\{u, v\} \in E$ we have $c(u) \neq c(v)$. The value $c(v)$ is called a color of the vertex v. A graph is called k-colorable if there exists a valid k-coloring.

Trivially, with the exception of the complete graph which requires n colors, every graph on $n = |V|$ vertices is $(n-1)$-colorable. However, many graphs can be colored with many fewer colors. Constructing a valid coloring online in the VAM-PH input model is difficult for general graphs. More precisely, we have the following result:

Theorem

For any deterministic online algorithm ALG for coloring in the VAM-PH input model, there exists a $\log n$*-colorable graph* G *such that ALG uses at least* $2n/(\log n)$ *colors. In other words, we have* $\rho(ALG) \geq \frac{2n}{\log^2(n)}$.

Coloring k-colorable graphs

The theorem shows that the class of $\log n$-colorable graphs does not admit online algorithms that are significantly better than the trivial algorithm. For the case of constant $k \geq 3$ colors, the competitive and approximation ratios remain an open question. Blum and Karger's (1997) $\tilde{O}(n^{3/14})$ approximation remains the best known offline approximation for 3-colorable graphs. (Coloring 3-colorable graphs is an NP-hard problem.)

The current state of the art for the online competitive ratio is $O(n^{1-\frac{1}{k!}})$ for coloring k-colorable graphs (and, additionally an improved $\tilde{O}(n^{2/3})$ for 3-coloroable graphs) due to Kierstead (1998).

For the case of $k = 2$ (i.e., bipartite graphs), tight bounds are known. We first note that trees are a special case of 2-colorable graphs. The following result shows that for every deterministic coloring algorithm ALG, there is a tree T with 2^{k-1} nodes such ALG uses k colors in coloring T. This will show that the competitive ratio $\rho \geq \frac{\log n}{2}$. A somewhat more involved argument shows that $\rho \geq \log n$ for which there is a corresponding positive result.

A lower bound for coloring trees

Theorem

Let ALG be a deterministic online algorithm for Graph Coloring problem restricted to bipartite graphs in the VAM-PH input model. Then there is a tree T with $n-1$ nodes such ALG uses atv least $\log n$ *colors when coloring T. That is, $\rho(ALG) \geq \frac{\log n}{2}$.*

Proof.

We prove the following statement by induction on k: given an arbitrary sequence of input items $I_1, \ldots, I_m$, the adversary can extend the sequence with disjoint trees $T_1, T_2, T_3, \ldots$ such that ALG colors roots of the trees with k different colors and the combined size of the trees is $\leq 2^k - 1$. See text.

□

First Fit is an optimal coloring algorithm for trees

The natural greedy coloring algorithm is First Fit. That is, for each new node, color it with the smallest non-conflicting color.

Theorem

For online coloring of trees, First Fit achieves the optimal $\rho = \log n$ *competitive ratio.*

Coloring bipartite graphs

Since First Fit is an optimal online coloring algorithm for trees, it is natural to ask how well First Fit performs on the more general class of bipartite graphs. It is easy to see that First Fit can do very poorly when coloring some bipartite graphs.

Fact

$\rho(FirstFit) \geq n/4$.

Proof.

Let $G_n = (U, V, E)$ be the $2n$ node bipartite graph where $|U| = |V| = n$ and $E = \{(u_i, v_j) | i \neq j\}$. That is G_n is a complete bipartite graph minus the edges $\{u_i, v_i\}$. The adversary presents the vertices in the following order: $u_1, v_1, u_2, v_2, \ldots u_n, v_n\}$. First Fit wll use n colors on this input sequence compared to the optimal 2 colors.

An optimal online coloring algorithm for bipartite graphs

While First Fit performs poorly on bipartite graphs, there is an online algorithm that will color every bipartite graph with n colors. Rather than greedily coloring each node v, the algorithm just avoids using the same color in the connected component in which v initially occurs. Here is the algorithm:

When a vertex v arrives, CBIP computes the connected component C_v (so far) to which v belongs. Since the entire graph is bipartite, C_v is also bipartite. CBIP computes a partition of C_v into two blocks: S_v that contains v and $\widetilde{S}_v$ that does not contain v. In other words, $C_v = S_v \cup \widetilde{S}_v$. Note that neighbors of v are only among $\widetilde{S}_v$. Let i denote the smallest color that does not appear in $\widetilde{S}_v$. CBIP colors v with color i.

Theorem

For coloring bipartite graphs, we have

$$\rho(CBIP) \leq \log n.$$

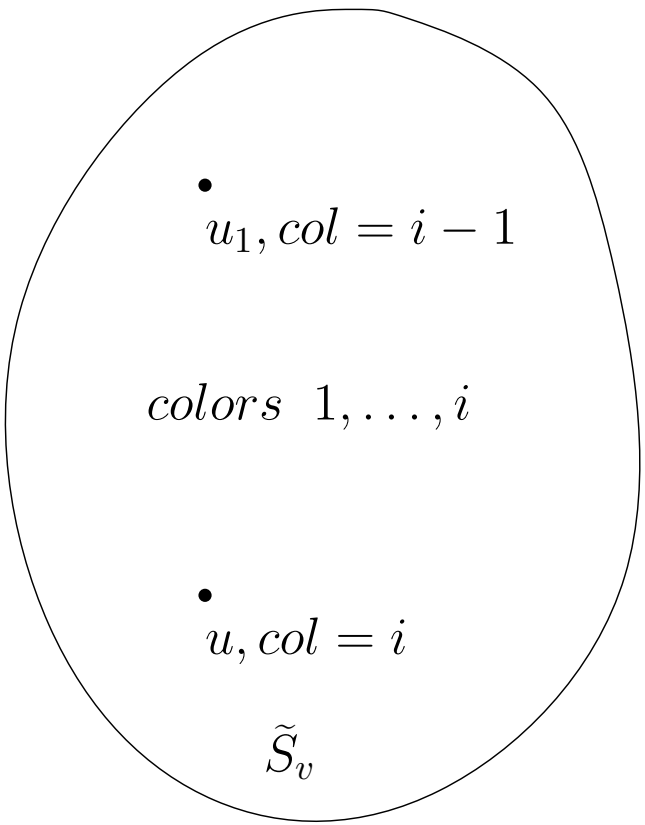
Proving the $\log n$ competitive ratio for coloring bipartite graphs

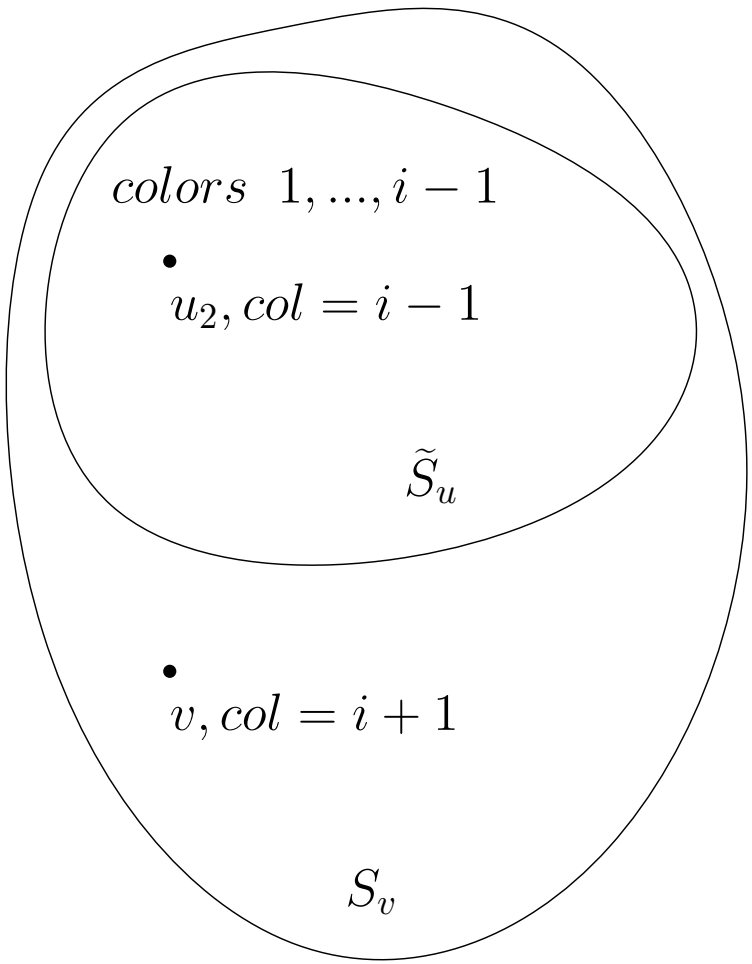
Let $n(i)$ denote the minimum number of nodes that have to be presented to CBIP in order to force it to use color i for the first time. We want to show that $n(i) \geq \lceil 2^{i/2} \rceil$ by induction on i.
Base cases: clearly we have $n(1) = 1$ and $n(2) = 2$.

Inductive step: let v be the first vertex that is colored with color $i+1$ by CBIP. Consider C_v, S_v, and $\widetilde{S}_v$ as defined.In particular, *all colors* $1, 2, \ldots, i$ appear among $\widetilde{S}_v$. Let u be a vertex in $\widetilde{S}_v$ that is colored i. Let $C_u, S_u, \widetilde{S}_u$ be defined for the vertex u at the time that it appeared. Since u was assigned color i, then all colors $1, 2, \ldots, i-1$ appeared in $\widetilde{S}_u$. Observe that $\widetilde{S}_u \subseteq S_v$. Therefore, there exists vertex $u_1 \in \widetilde{S}_v$ colored $i-1$ and there exists vertex $u_2 \in S_v$ colored $i-1$, as well.
Without loss of generality assume that $u_1 \prec u_2$. At the time that u_2 was colored, the connected component of u_2 had to be disjoint from the connected component of u_1, for otherwise u_2 would not have been colored with the same color as u_1.

Finishing the proof for online bipartite graph coloring

Thus, we have $C_{u_1} \cap C_{u_2} = \varnothing$. Furthermore, we can apply the inductive assumption to each of C_{u_1} and C_{u_2} to get that $|C_{u_1}|, |C_{u_2}| \geq \lceil 2^{(i-1)/2} \rceil$. Hence the number of vertices that have been presented prior to v is at least $|C_{u_1}| + |C_{u_2}| \geq 2\lceil 2^{(i-1)/2} \rceil \geq \lceil 2^{(i+1)/2} \rceil$.





What other classes of graphs have small competitive ratios for coloring? A tour of some graph classes

Since trees are such a constrained class of graphs, there are many generalizations. One of the most studied generalizations is that of *bounded tree width graphs*. Tree width graphs are further generalized by the class of (degree) d-inductive graphs.

A graph G is called *d-inductive* (also called d-degenerate) if there is an ordering of vertices such that every node u has at most d neighbors appearing after u in the ordering. When a d-inductive graph G is given in the VAM-PH input model, the order in which the nodes are presented is adversarial and can be different from an inductive ordering.

Continued tour of some graph classes and their online coloring

If the nodes of G are presented in the reverse order of an inductive order then the FirstFit algorithm uses at most $d + 1$ colors. This implies that every d-inductive graph is $(d + 1)$-colorable. We show in the text that when a d-inductive graph is given in an adversarial VAM-PH model, the FirstFit algorithm uses at most $O(d \log n)$ colors. In particular, this generalizes the fact that FirstFit achieves competitive ratio $\log n$ on trees, noting that trees are 1-inductive.

Theorem

FirstFit uses at most $O(d \log n)$ colors on any d-inductive graph G under the VAM-PH input model.

Continued tour of graph classes

An interval graph is the graph induced by the intersection of intervals on the line. Interval graphs are a special case of *chordal graphs* which in turn are *perfect graphs*. There is a characterization of chordal graphs using the concept of a *perfect elimination ordering*. For a graph G, an order $v_1, v_2, \ldots, v_n$ is a prefect elimination order if for all i, $Nbhd(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ is a clique. Equivalently $Nbhd(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ does not have 2 independent (i.e. non adjacent) nodes.

For any graph G, we have $\chi(G) \geq \omega(G)$ where $\chi(G)$ is the chromatic number of a graph and $\omega(G)$ is the clique number of a graph. A perfect graph G is one that satisfies $\chi(G) = \omega(G)$.

We can generalize chordal graphs to the class of d-inductively independent graphs which have a vertex ordering $v_1, v_2, \ldots, v_n$ such that $Nbhd(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ does not have $d+1$ independent nodes.

Clearly Every d-inductive graph is d-inductively independent.

Online coloring of interval graphs

For interval graphs, it is known that First Fit has a constant (but not optimal) competitive ratio. However, we have the following:

Theorem

For every deterministic online coloring algorithm ALG for interval graphs, there is an interval graph such that ALG uses $3\omega - 2$ colours. There is a deterministic online coloring algorithm ALG^* that colors every interval graph G using at most $3\omega(G) - 2$ colors. Therefore, $\rho(ALG^*) = 3$.

Albers and Schraink (2017) provide randomized online coloring lower bounds for many graphs classes including trees, chordal graphs, and d-inductive graphs. The tree lower bound also implies that planar and bipartite graphs also have an $\Omega(\log n)$ lower bound on the randomized competitive ratio.

Sketch of the interval coloring algorithm

The algorithm might be called *online recursive greedy*
For a sequence of initial nodes in the interval graph, the congestion number (i.e., clique number) is the maximum number of edge intersections. For each clique number k, we can recursively define an algorithm $RECG_k$:

The base case trivially colors all nodes witn color 1.

Induction step assuming $RECG_k$ has been defined. Consider the sequence $\sigma = v_1, \ldots v_i$ which we assume has clique number at most $k+1$. Let $A = \{v_{j_1}, \ldots, v_{j_r}\}$ be a minimal set of nodes (i.e., intervals) such that $\sigma' = \sigma \setminus A$ can be colored with k colors.
If $\sigma' v_{i+1}$ has cliques number k, then color v_{i+1} using $RECG_k$. Otherwise use color $k+1$ for v_{i+1}.

The online interval coloring algorithm

The online algorithm uses $RECG_k$ until clique number $k+1$ is created by some new vertex v_i and then we start using $RECG_{k+1}$.

The proof of the $3\omega - 2$ competitive ratio relies on showing that Av_i can be colored using only 3 new colors.

The ends our discussion of Chapter 5.

Chapter 6 concerns two perhaps seemingly unrelated problems:

1. Maximizing a non-monotone submodular function (without any constraints)
2. Max-Sat