Approximation and Online Algorithms for Generalized Interval Coloring Problems

Arindam Pal
arindamp@cse.iitd.ac.in

Department of Computer Science and Engineering
Indian Institute of Technology Delhi

February 11, 2013
Indian Statistical Institute Kolkata

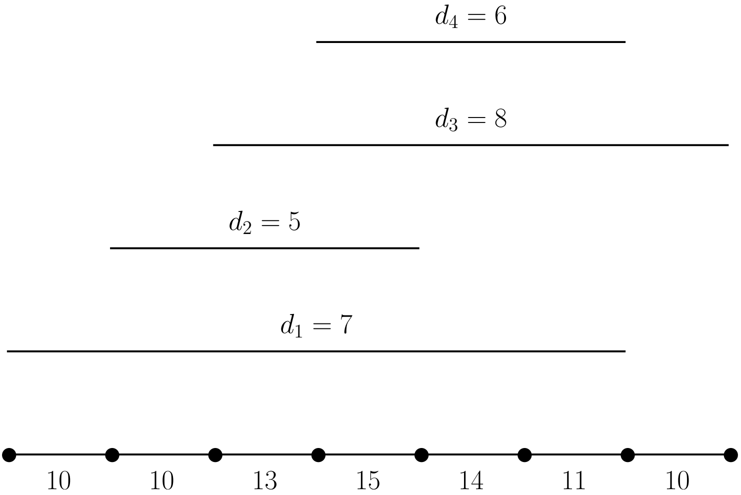
Agenda

- The INTERVAL COLORING problem and its variants
- Survey of existing results
- Our contributions
- Approximation algorithms for INTERVAL COLORING
- Online algorithms for INTERVAL COLORING
- Conclusion and future work

The Interval Coloring problem

- Given a path $P = (v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n)$ on n nodes.
- Edge e_i has capacity $c(e_i) \equiv c_i$.
- There are k intervals (requests) $I_1, \ldots, I_k$.
- $I_i = [s_i, t_i]$ and there is a demand d_i associated with it.
- A set of intervals $\mathcal{I}$ is *feasible* if the total demand of all intervals in $\mathcal{I}$ passing through any edge e does not exceed it's capacity $c(e)$.
- Goal is to partition the requests $I_1, \ldots, I_k$ into a number of sets such that each set is feasible and the total number of sets is minimized.
- We can think of this as assigning colors to intervals so that each color class is feasible and we want to minimize the number of colors.
- This can also be thought of as routing the requests in a feasible manner in a number of rounds.
- Can be studied under offline or online setting.

A sample INTERVAL COLORING instance



Motivation

- The path graph is a natural setting for many applications, where a limited resource is available and the amount of the resource varies over time.
- Many combinatorial optimization problems which are NP-HARD on general graphs remain NP-HARD on paths.
- We can represent time instants as vertices, time intervals as edges and the amount of resource available at a time interval as the capacity of the corresponding edge.
- The requirement of a resource between two time instants can be represented as a demand between the corresponding vertices with a certain profit associated with it.

Application of INTERVAL COLORING

- Consider an optical line network, where each color corresponds to a distinct frequency in which the information flows.
- Different links along the line have different capacities, which are a function of intermediate equipment along the link.
- Each request uses the same bandwidth on all links that this request contains.
- As the number of distinct available frequencies is limited, minimizing the number of colors for a given sequence of requests is a natural objective.

Related work for INTERVAL COLORING

- INTERVAL COLORING is NP-HARD for arbitrary demands since, if we take P to be a single edge, this is the BIN PACKING problem.
- If all capacities and demands are 1, this is the INTERVAL GRAPH COLORING problem, for which a greedy algorithm gives the optimum coloring with ω colors, where ω is the maximum clique size of the *interval graph*.
- For the corresponding online problem, Kierstead and Trotter gave an online algorithm which uses at most $3\omega - 2$ colors. They also gave a lower bound of $3\omega - 2$ on the number of colors required in any coloring output by any deterministic online algorithm.
- Leonardi and Vitaletti showed that no randomized algorithm for online coloring of interval graphs can achieve a competitive ratio strictly better than $3\omega - 2$.

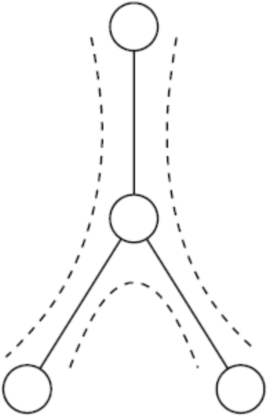
Related work for Interval Coloring ...

- The best upper bound known for the First-Fit algorithm is 8ω by Pemmaraju et al., and a lower bound of 4.4ω was shown by Chrobak and Slusarek.
- For unit capacities and arbitrary demands, Narayanaswamy gave a 10-competitive algorithm. Epstein et al. proved a lower bound of $\frac{24}{7} \approx 3.43$ for this problem.
- For arbitrary capacities and demands, Epstein et al. gave a 78-competitive algorithm, assuming that the maximum demand is at most the minimum capacity (*no-bottleneck assumption*).
- They also proved that without this assumption, there is no deterministic online algorithm for interval coloring with nonuniform capacities and demands, that can achieve a competitive ratio better than $\Omega(\log\log n)$ or $\Omega\left(\log\log\log\left(\frac{c_{\max}}{c_{\min}}\right)\right)$. Here, $c_{\max}$ and $c_{\min}$ are the maximum and minimum edge capacities of the path respectively.

Interval Coloring on trees

- It is easy to construct a set of intervals on a binary tree requiring at least $\frac{3L}{2}$ colors, where L is the maximum load on any edge.
- Raghavan and Upfal gave an algorithm to color any set of paths of maximum load L on a tree using at most $\frac{3L}{2}$ colors.
- Bartal and Leonardi gave an $O(\log n)$-competitive algorithm for the special case when $d_i = 1, 1 \leq i \leq k$ and $c_e = 1, e \in E$, *i.e.*, when all capacities and demands are one.
- They also proved that any deterministic online algorithm for trees cannot have competitive ratio better than $\Omega\left(\frac{\log n}{\log\log n}\right)$.
- Leonardi and Vitaletti showed that for trees of diameter $\Delta = O(\log n)$, no randomized algorithm for online coloring can achieve a competitive ratio better than $\Omega(\log \Delta)$.

Lower bound example on trees



Our results

For paths:

- Optimal algorithm for unit demands, arbitrary capacities.
- 3-approximation algorithm for uniform capacities, arbitrary demands.
- 24-approximation algorithm for arbitrary capacities and arbitrary demands with NBA.
- 58-competitive online algorithm with NBA.

For trees:

- 64-approximation algorithm with NBA.
- $O(\log n)$-competitive online algorithm for uniform capacities and arbitrary demands.

Our results

	Path			Tree
	Unit demand	Unit capacity	Arbitrary	
Offline	OPTIMAL	3	24	64
Online	NONE	10	58	$O(\log n)$

Preliminaries

- $F_e =$ Set of all requests passing through edge e.
- $l_e =$ Total demand of all requests passing through $e = \sum_{i:I_i \in F_e} d_i$, is the *load* on edge e.
- $r_e = \left\lceil \frac{l_e}{c_e} \right\rceil$, is the *congestion* on edge e.
- $r = \max_{e \in E} r_e$, is the maximum congestion on any edge.
- Let OPT be the minimum number of colors required for the given problem instance. Clearly, $\mathrm{OPT} \geq r$.
- If ω demands are mutually incompatible with each other, then each of them has to be assigned a different color. Hence, $\mathrm{OPT} \geq \omega$.
- The *bottleneck edge* b_i of a request I_i is the minimum capacity edge on the path from s_i to t_i. We denote the capacity of bottleneck edge also by b_i.

Approximation algorithms for Interval Coloring with arbitrary capacities and demands

- Separate the requests based on whether $d_i > \frac{1}{4}b_i$ (large demands) or $d_i \leq \frac{1}{4}b_i$ (small demands), where b_i is the bottleneck edge capacity.
- We sort the small demands based on their left endpoints and then assign a demand to the first color, where the total load on the bottleneck edge e (excluding this demand) is at most $\frac{c_e}{16}$.
- It can be proven that this requires at most $16r$ colors and the coloring is feasible.

- For large demands, round down capacity of every edge to the nearest multiple of $c_{\min}$.
- This will increase the congestion r by a factor of 2.
- Round up every demand to $c_{\min}$. Note that for any large demand, $d_i > \frac{1}{4} b_i \geq \frac{1}{4} c_{\min}$.
- Moreover, $d_i \leq c_{\min}$ because of NBA.
- This will increase the congestion r by a factor of 4.
- The resulting instance has uniform demands, which can be colored with r colors. So, large demands require $8r$ colors.
- In total, we require at most $24r \leq 24 \cdot \mathrm{OPT}$ colors.

Online Interval Coloring with arbitrary capacities and demands

- We scale down all capacities and demands by a factor of $c_{\min}$, so that the new $c_{\min} = 1$ and the new $d_{\max} \leq 1$.
- Then, we round down all edge capacities to the nearest power of 2, so that if $c(e) \in [2^k, 2^{k+1})$ then the new $c(e) = 2^k$.
- The *class* of a demand d_i is defined as $\ell_i = \log_2 c(b_i)$.
- For a demand d_i in class $j \geq 1$, we call it a small demand if $d_i \leq \min(1, 2^{j-3})$.
- For a demand d_i in class 0, we call it a small demand if $d_i \leq \frac{1}{4}$.
- Note that large demands can exist only in classes 0, 1 and 2.

Schematic representation of classes of demands

Class	Small	Large	Bottleneck capacity	Allocated capacity
0	$(0, \frac{1}{4}]$	$(\frac{1}{4}, 1]$	1	1
1	$(0, \frac{1}{4}]$	$(\frac{1}{4}, 1]$	2	1
2	$(0, \frac{1}{2}]$	$(\frac{1}{2}, 1]$	4	2
3	$(0, 1]$	NONE	8	4
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
j	$(0, 1]$	NONE	2^j	2^{j-1}

Handling small demands

- Small demands are $\frac{1}{4}$-small.
- The resulting instance has uniform capacity.
- 4-competitive algorithm for this.
- Additional loss of a factor of 8 due to rounding and allocating only 2^{j-1} capacity instead of 2^j.
- So this is 32-competitive.

Algorithm for small demands and uniform capacity

- Our algorithm partitions intervals into disjoint sets and colors each set independently with separate colors.
- $S = \{S_1, S_2, \ldots\}$ is the family of sets containing already processed requests.
- S_i is the set of requests at *level* i.
- For each new request R, we look for a set with the lowest possible index k such that the total load of all the demands in $\left(\bigcup_{i=1}^{k} S_i\right) \cup \{R\}$ on any edge e of R does not exceed $\frac{1}{4}kc$.
- If on any edge e this inequality is violated, we call e a *critical edge* of R on that level.
- Note that e is the edge which prevented R to be put on level k.

An online algorithm for $\frac{1}{4}$-small demands

$k \leftarrow 1$;
while *there are still requests in the input* **do**
    let R be the next request;
    while *for any edge* $e \in R$, $l_e\left(\left(\bigcup_{i=1}^{k} S_i\right) \cup \{R\}\right) > \frac{1}{4}kc$ **do**
        `//` e `is called a critical edge of` R `on level` k.
        $k \leftarrow k + 1$;
    end
    $S_k \leftarrow S_k \cup \{R\}$;
    give R the lowest numbered color not used in any sets $S_1, \ldots, S_{k-1}$ and consistent with S_k;
end

Competitive ratio

- Small demands require at most $32 \cdot \mathrm{OPT}$ colors.
- Large demands in classes 0, 1 and 2 require at most $26 \cdot \mathrm{OPT}$ colors.
- Total number of colors required is at most $58 \cdot \mathrm{OPT}$.
- Hence, this algorithm is 58-competitive.

Online Interval Coloring for trees

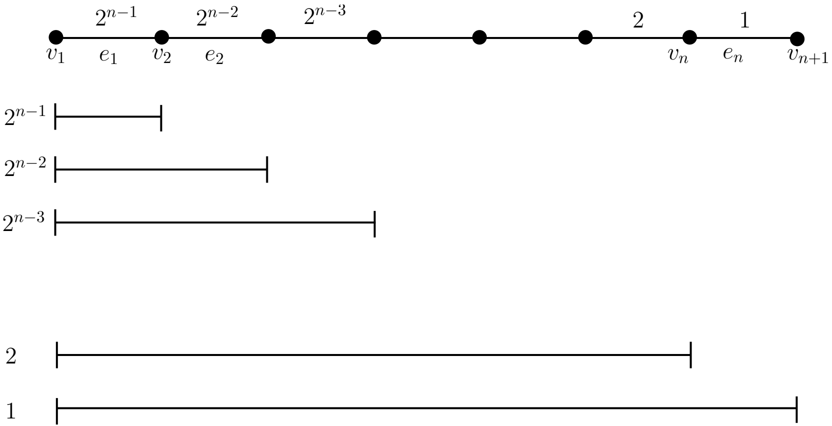
- Given a tree with n vertices, we can find a vertex r, whose removal partitions the vertices into disconnected components, each of which has size at most $\frac{n}{2}$.
- We call such a vertex a *vertex separator*.
- We can divide each of these components further in a similar manner recursively.
- The vertex set V will thus be partitioned into classes $V_1 = \{r\}, V_2, \ldots, V_{\log n}$.
- The vertices in V_i are called *level* i *vertex separators*.
- Request R is called a *level* i *request*, if i is the minimum level of any vertex in the interval I of R.

- Alternatively, we can also classify the requests based on the *least common ancestor* of the endpoints of a request R, $\mathrm{LCA}(s,t)$.
- A (balanced) binary tree has height $O(\log n)$.
- A request is on *level* i, if $\mathrm{LCA}(s,t)$ is on *level* i.
- Note that a request can be on only one level.

Algorithm

- We allocate separate colors for requests on different levels.
- When a request on any level comes, we use $\textsc{First-Fit}$ to assign it to the lowest available color, while maintaining feasibility.
- For requests on a particular level, $\textsc{First-Fit}$ is 2-competitive.
- For binary trees with n vertices, our algorithm is $(2 \log n)$-competitive.
- For b-ary trees, this will give a $(b \log n)$-competitive algorithm.

How bad the congestion bound can be?



$$\mathrm{OPT} = n, r = 2, \omega = n.$$

Conclusion

- In this talk, we presented several algorithms for solving various instances of the INTERVAL COLORING problem.
- We saw that some special cases of this problem can have much better algorithms.
- We gave a constant factor competitive algorithm for paths and an $O(\log n)$-competitive algorithm for trees for the ONLINE INTERVAL COLORING problem.

Future work

- Is there a unified algorithm for INTERVAL COLORING for all cases?
- Can we improve the approximation factor of the INTERVAL COLORING problem on paths and trees?
- What is the approximability of these problems without the *no-bottleneck assumption*?
- Is there a better constant factor competitive algorithm for the ONLINE INTERVAL COLORING problem on paths?
- What is the hardness of approximation of these problems?
- What is the lower bound on the competitive ratio of online algorithms?

Questions?