Competitive Analysis and the Online Paging Problem

Algorithms in Uncertain Scenarios (097280)

Yuval Emek

Spring 2024/5

Rent or buy?

- Going skiing for unknown number of days
  - Call your boss every evening to ask if may stay for another day
- Ski gear costs
  - Renting: 1 EUR per day
  - Buying: $b > 1$ EUR
- Will you buy or rent?
- Exact rules:
  - Every day, input is "continue skiing" or "go home"
  - If input is "continue skiing" and still didn't buy ski gear, then
    - rent for one more day and pay 1 EUR; or
    - buy and pay b EUR
  - Objective: minimize costs in current vacation
  - No prior knowledge on vacation length
    - Aim for worst case scenario
- *Online algorithm* decides on today's action based on input seen so far

Quality of online algorithms

- How do we measure the quality of an online algorithm?
- Failed attempt: overall cost in worst case
  - Best approach in ski rental is always buy and pay b?
  - Infeasible if overall cost is unbounded
    - Example soon...
- Solution: compare online algorithm to optimal offline algorithm
  - Runs on same input
  - Knows input in advance
- We wish to design online algorithm Alg so that

$$R(\sigma) = \texttt{Alg}(\sigma)/\texttt{Opt}(\sigma)$$

  is small for all input instances σ
  - Opt = optimal offline algorithm
- Convenient to assume:
  σ picked by adversary whose aim is to maximize $R(\sigma)$
  - Knows Alg
  - Collaborates with Opt

Ski rental algorithm

- Ski rental instance σ defined by #skiing days
  - Denote by (unknown) variable D
- $\texttt{Opt} = \min\{b, D\}$
- Online $\texttt{Alg}$ fully characterized by parameter t:
  - Rent up to (including) day $t - 1$
  - Buy on day t if vacation did not end beforehand
  - t determined by algorithm designer
- $\texttt{Alg}(\sigma) = \begin{cases} D, & D < t \\ t - 1 + b, & D \geq t \end{cases}$
- Observation: no point for adversary to extend vacation beyond day t
  - May increase $\texttt{Opt}(\sigma)$, but not $\texttt{Alg}(\sigma)$
- Adversary knows $t \implies$ w.l.o.g. $D \leq t$

Adversary's considerations

- Case 1: $D < t$
  - $R(\sigma) = \dfrac{D}{\min\{D, b\}}$
  - Monotone non-decreasing function of D, so adversary sets $D = t - 1$
  - Yields $R(\sigma) = \dfrac{t-1}{\min\{t-1, b\}} \leq \dfrac{t}{\min\{t, b\}}$
- Case 2: $D = t$
  - $R(\sigma) = \dfrac{t - 1 + b}{\min\{t, b\}}$
- $b > 1 \implies$ adversary prefers case 2

Algorithm designer's considerations

- Designer of `Alg` wishes to minimize $\dfrac{t-1+b}{\min\{t,b\}}$
- If we set $t = b$, then

$$R(\sigma) \leq \frac{2b-1}{b} = 2 - 1/b$$

- Analysis shows that this upper bound is tight

Comments

- Rare example for establishing upper bound together with lower bound
  - Toy problem
  - `Alg`'s structure is very simple (single parameter)
- Does analysis hold if algorithm is randomized?
  - If adversary is oblivious to `Alg`'s coin tosses, then no: randomized `Alg` can ensure $R(\sigma) \leq \frac{e}{e-1} \approx 1.58$
  - Where does analysis fail?

Managing multiple memory layers

- Modern computer memories organized in multiple layers
  - smaller layers — faster access
- Paging problem: 2 layer memory
  - Main memory with *N pages*
  - *Cache* that stores $k < N$ pages at any given time
- Program can access page only if stored in cache
- *Page fault (PF)* = requested page not in cache
  - Should be copied to cache, evicting some other page

Online paging — formal definition

- Input: sequence $\sigma \in [N]^*$ of *requests*
  - Each request specifies index $i \in [N]$ of some page
- Algorithm maintains cache configuration $C \subseteq [N]$ of size $|C| = k$
- On request $i \in [N]$, algorithm should update $C \to C' \ni i$
  - *Cost* of this operation $= |C' - C|$
- Online computation: algorithm doesn't know future requests
- Do we have to consider arbitrary C'?

Lazy paging algorithm

- Algorithm `Alg` is *lazy* if:
  - Evicts exactly one page on PF
  - Does nothing if no PF
- Observation: every paging algorithm can be turned into lazy algorithm without affecting cost
  - Delay non-urgent actions until they become necessary (if at all)
- Concentrate hereafter on lazy algorithms
- Lazy `Alg` pays 1 cost unit per PF (and nothing else)
- Policy of lazy `Alg` boils down to:
  which page to evict from cache on PF?

Relative quality measure

- No natural bound on request sequence length $\Longrightarrow$ no natural bound on cost of $\texttt{Alg}$ and $\texttt{Opt}$
  - In contrast to ski rental
- Becomes clear that $\texttt{Alg}(\sigma)$ should be bounded relatively to $\texttt{Opt}(\sigma)$
- What about initial cache configuration?
  - Has bounded effect on both $\texttt{Alg}(\sigma)$ and $\texttt{Opt}(\sigma)$
  - Vanishes in long run as $\texttt{Alg}(\sigma)$ and $\texttt{Opt}(\sigma)$ increase

The quality measure

Definition

Consider some problem P and a (deterministic) online algorithm $\texttt{Alg}$. We say that $\texttt{Alg}$ is *c-competitive* for P if there exists some $\alpha \in \mathbb{R}_{\geq 0}$ such that for any instance σ of P, it holds that

$$\texttt{Alg}(\sigma) \leq c \cdot \texttt{Opt}(\sigma) + \alpha$$

if P is a minimization problem; and

$$\texttt{Alg}(\sigma) \geq \frac{1}{c} \cdot \texttt{Opt}(\sigma) - \alpha$$

if P is a maximization problem.

- c = *competitive ratio (CR)*
- α = *additive constant*
- Main algorithmic challenge: design online algorithms with small CR

Role of the additive constant

- Why do we need the additive constant?
- Ignore phenomena that have bounded effect on cost of `Alg` and `Opt`
  - E.g., initial configuration in paging
- Concentrate on asymptotic behavior of `Alg` relative to `Opt`
- Often makes analysis simpler

Phases

- Family of online paging algorithms known as *marking algorithms*
- Key component:
  partition request sequence σ into *phases* defined inductively:
  - Phase 0 is an empty subsequence (ends at time 0)
  - Phase $i \geq 1$ is the maximal subsequence following phase $i - 1$ that contains $\leq k$ distinct page requests
- Phase $i + 1$ (if exists) begins on request for $(k + 1)$th distinct page since beginning of phase i

The marking rule

- Each page associated with a *mark* bit
  - Page can be *marked* or *unmarked*
- Based on partition into phases, impose marking rules:
  - Mark a page every time it is requested
  - At the end of a phase, unmark all pages
- Marking invariant maintained throughout each phase:
  #marked pages = #distinct requested pages
  - By phase definition, $\leq k$ marked pages at any moment
- Partition of σ into phases and marking scheme do not depend on `Alg`

The marking algorithms family

Definition

Algorithm `Alg` is a *marking algorithm* if it never evicts a marked page from the cache.

- Feasible due to aforementioned marking invariant
- A family of algorithms (rather than one algorithm)
  - May exist multiple unmarked pages to choose from
- Examples:
  - Least recently used (LRU)
  - Clock replacement
    - Approximating (implicit) clock of LRU by 1 bit
  - Flush when full (FWF)
    - Strictly speaking, doesn't fit our definition of paging

The Competitive Ratio of Marking Algorithms

Theorem

Let `Alg` *be any online marking algorithm for the paging problem. Then* `Alg` *is k-competitive.*

- Consider request sequence σ
- Claim: `Alg` may suffer $\leq k$ PFs during any phase
  - Each page is marked when 1st requested and remains marked until phase ends
  - `Alg` does not evict marked pages
  - $\Longrightarrow$ `Alg` cannot fault twice on same page during phase
  - Claim follows since k distinct pages are requested during phase
    - Maybe less if last phase

Proof continues

- Fix some phase i which is not last
- p_i = first requested page in phase i
- t_i = time of request p_i
- Immediately after time t_i, page p_i must be in $\texttt{Opt}$'s cache
  - Leaving room for $k-1$ other pages
- By phase definition, until (including) time t_{i+1}, $\texttt{Opt}$ receives requests for k distinct pages other than p_i
- $\implies$ Must suffer ≥ 1 PF
- Summing up: every phase other than last contributes $\leq k$ to $\texttt{Alg}(\sigma)$ (by Claim) and ≥ 1 to $\texttt{Opt}(\sigma)$
- Last phase contributes $\leq k$ to $\texttt{Alg}(\sigma)$ (by Claim)
- Therefore $\texttt{Alg}(\sigma) \leq k \cdot \texttt{Opt}(\sigma) + k$ ∎
- Demonstrates convenience in introducing additive constant

Up-bounding cost of Opt

- Can we do better than that?
  - Recall that marking family is very wide
  - Better non-marking algorithms?
- Establish lower bound on CR of deterministic paging when $N = k + 1$
- Offline marking algorithm `OffMark`:
  on PF, evict unique page not requested during current phase
  - Well defined since $N = k + 1$
  - Not an online algorithm!
- Observation: `OffMark` makes ≤ 1 PFs per phase
  - Evicted page will not be requested until next phase begins
  - All other $(N - 1 = k)$ pages are in cache
- $\Longrightarrow$ Total cost of `Opt` $\leq$ #phases
  - `Opt` at least as good as `OffMark`

Bad request sequence

Theorem

Let `Alg` *be any (deterministic) online paging algorithm. Then, the CR of* `Alg` *is at least* k.

- Fix $N = k + 1$
- Construct arbitrarily long request sequence σ by always requesting the (unique) page missing from `Alg`'s cache
  - $\texttt{Alg}(\sigma) = |\sigma|$
- Each (non-last) phase contains $\geq k$ requests $\Longrightarrow$ cost of `Alg` in a phase $\geq k$.
- Assertion follows since $\texttt{Opt}(\sigma) \leq \#\text{phases}$ ■
- How does proof addresses additive constant?
  - By making σ arbitrarily long
- Does lower bound hold if $N > k + 1$?
  - Adversary can always ignore some pages
- Does proof hold for randomized paging algorithms?

Competitive ratio of randomized online algorithms

Definition

Consider some problem P and a randomized online algorithm $\texttt{Alg}$. We say that $\texttt{Alg}$ is *c-competitive* for P against an *oblivious adversary* if there exists some $\alpha \in \mathbb{R}_{\geq 0}$ such that for any instance σ of P, it holds that

$$\mathbb{E}[\texttt{Alg}(\sigma)] \leq c \cdot \texttt{Opt}(\sigma) + \alpha$$

if P is a minimization problem; and

$$\mathbb{E}[\texttt{Alg}(\sigma)] \geq \frac{1}{c} \cdot \texttt{Opt}(\sigma) - \alpha$$

if P is a maximization problem.

- Expectation over random decisions of online algorithm
  - Do not make any assumptions on input!
- Adversary knows $\texttt{Alg}$, but oblivious to its coin tosses

Harmonic numbers

- The nth *harmonic number* is

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

- Well known (and easy to verify):

$$\ln n < H_n \leq 1 + \ln n$$

  - $H_n = \Theta(\log n)$

Algorithm RandMark

- Online marking algorithm RandMark:
  on PF, evict unmarked page chosen u.a.r.
  - Yet another member of marking algorithms family

Theorem

Algorithm RandMark *is* $(2H_k)$*-competitive.*

- Consider request sequence σ and fix some phase i
  - C_i = cache configuration immediately before beginning of phase
- Page $p \in [N]$ is
  - stale if $p \in C_i$ and p requested during phase i
  - fresh if $p \notin C_i$ and p requested during phase i
  - fading if $p \in C_i$ and p not requested during phase i
- m_i = #distinct fresh pages requested during phase i
  - $k - m_i$ distinct stale pages requested during phase i

Proof continues

- RandMark suffers PF on 1st request for (each) fresh page
- May also suffer PF on 1st request for (each) stale page
  - Depending on coin tosses
- Doesn't suffer PF on any subsequent request for fresh or stale page
- Probabilities for PFs on stale pages maximized if 1st requests for m_i fresh pages arrive before 1st requests for $k - m_i$ stale pages
  - Worst possible arrival order from perspective of RandMark
  - Assume hereafter

Proof continues

- p_j = jth requested stale page
  - $1 \leq j \leq k - m_i$
- Upon 1st request for p_j, exactly $k - (j-1)$ unmarked stale and fading pages
  - $k - (j-1) - m_i$ of them are in cache
- `RandMark` suffers PF on p_j w.p.

$$1 - \frac{k-(j-1)-m_i}{k-(j-1)} = \frac{m_i}{k-(j-1)}$$

- Expected #PF during phase i is

$$\begin{aligned} m_i + \sum_{j=1}^{k-m_i} \frac{m_i}{k-(j-1)} &= m_i + m_i \sum_{\ell=m_i+1}^{k} \frac{1}{\ell} \\ &= m_i \left(1 + H_k - H_{m_i}\right) \leq m_i H_k \end{aligned}$$

Proof continues

- #distinct pages requested during phases $i-1$ and i is $\geq k + m_i$
  - `RandMark` is marking algorithm, hence none of fresh pages was requested in phase $i-1$
- $\Longrightarrow$ #PFs suffered by `Opt` during phases $i-1$ and i is $\geq m_i$
  - For each $i > 1$
- Summing up:

$$\texttt{RandMark}(\sigma) \leq \sum_{i \geq 1} m_i H_k = H_k \sum_{i \geq 1} m_i$$

$$\texttt{Opt}(\sigma) \geq \frac{1}{2} \sum_{i > 1} m_i$$

- Charge PFs suffered by `RandMark` in phase 1 to additive constant ■
- Can we do better than that?

RandMark is asymptotically optimal

Theorem

Let Alg be any randomized online algorithm for the paging problem with $N = k + 1$ pages. The competitive ratio of Alg is at least H_k.

- Adversary maintains $p(j)$ = probability that page j not in cache
  - For $j \in [N]$
  - Feasible because adversary knows Alg
- At any time, $\sum_{j=1}^{N} p(j) = 1$
  - Exactly 1 page not in cache, so events are disjoint
- Construct bad σ request-by-request based on $(p(1), \ldots, p(N))$
  - Procedural manner
- Procedure maintains partition of σ to phases and marking bits
  - Recall: independent of online algorithm
- Explain how procedure constructs 1 phase
- σ consists of arbitrarily long sequence of phases

Proof continues

- Phase excluding 1st request consists of k subphases
  - When subphase i begins, $k + 1 - i$ unmarked pages
- Each subphase consists of:
  - Prefix: 0 or more requests for marked pages
  - Suffix: 1 request for unmarked page
    - Becomes marked
- Last request of kth subphase = 1st request of next phase
- Show that expected cost of `Alg` on ith subphase is $\frac{1}{k+1-i}$
  - Sums up to expected cost of entire phase =

$$\sum_{i=1}^{k} \frac{1}{k+1-i} = \sum_{\ell=1}^{k} \frac{1}{\ell} = H_k$$

  - Completes the proof since $\texttt{Opt}(\sigma) \leq \#\text{phases}$

Proof continues

- M, U = sets of marked and unmarked pages, respectively, at beginning of ith subphase
  - By definition, $u = |U| = k + 1 - i$
  - So, $|M| = i$
- χ = variable that counts total expected cost of `Alg` in ith subphase
- Goal: ensure that $\chi \geq 1/u$ when subphase ends
- $\mu = \sum_{j \in M} p(j)$
- If $\mu = 0$, then $\sum_{j \in U} p(j) = 1$
  - Pigeonhole principle: there exists $j \in U$ such that $p(j) \geq 1/u$
  - Adversary ends subphase by requesting page j which sets $\chi \geq 1/u$
- Note: marking algorithms always satisfy $\mu = 0$
  - Analysis ends here for this family

Proof continues

- Assume $\mu > 0$
- There exists page $\ell \in M$ such that $p(\ell) = \epsilon > 0$
- 1st request in subphase is for page ℓ
- Then, adversary keeps adding requests by executing loop:
  - While $\chi < 1/u$ and $\mu > \epsilon$, add to σ request for page $j \in M$ such that $p(j) > \epsilon/|M|$
  - Loop is feasible:
    if $\mu > \epsilon$, then there exists $j \in M$ such that $p(j) > \epsilon/|M|$
    - By pigeonhole principle
  - Loop must terminate: χ increases by $\geq \epsilon/|M|$ in each iteration
- When loop terminates, if $\chi \geq 1/u$, then adversary ends subphase by adding request for arbitrary page in U
- Assume $\mu \leq \epsilon$ when loop terminates
- There exists $j \in U$ such that $p(j) \geq \frac{1-\mu}{u} \geq \frac{1-\epsilon}{u}$
  - By pigeonhole principle
- Adversary ends subphase by adding request for page j
- Combined with ℓ's contribution, we have $\chi \geq \epsilon + \frac{1-\epsilon}{u} \geq \frac{1}{u}$ ∎

Constructive proof

- Proof is constructive:
  procedure that constructs bad request sequence for given `Alg`
    - Not necessarily efficient (definitely tedious)
- Some lower bound proofs merely establish existence of bad instance
    - Without explicitly specifying it
    - Less desirable in some cases, but still serves purpose of lower bound

Metrical Task Systems

Algorithms in Uncertain Scenarios (097280)

Yuval Emek

Spring 2024/5

1. The Online Metrical Task System Problem

2. Deterministic algorithms
   - Traversal algorithms
   - General Metric Spaces

3. A Randomized Algorithm for Uniform Metric Spaces

4. Algorithm Design via Probabilistic Embedding
   - Metric embedding
   - Probabilistic embedding

Metric spaces

Definition

A *metric space (MS)* $\mathcal{M}$ is a pair $\mathcal{M} = (V, \delta)$, where V is a set of *points* and $\delta : V \times V \to \mathbb{R}_{\geq 0}$ is a *distance function* that satisfies the following three conditions for every $x, y, z \in V$:

- $\delta(x, y) = 0$ if and only if $x = y$ (reflexivity, positivity);
- $\delta(x, y) = \delta(y, x)$ (symmetry); and
- $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$ (triangle inequality).

$\mathcal{M}$ is said to be *finite* if V is finite.

- Fundamental mathematical structure, plays key role in many fields
- Examples:
  - Real vector space with Euclidean norm (infinite)
  - More generally, real vector space with ℓ_p norm, $1 \leq p \leq \infty$ (infinite)
  - Vertices of finite positively weighted connected undirected graph with pairwise distances (finite)

Serving tasks

- Finite MS $\mathcal{M} = (V, \delta)$
  - Points = system's states
  - Distance function = cost of moving from one state to another
  - For simplicity: $V = [n]$
- *Task* = vector $r \in (\mathbb{R}_{\geq 0} \cup \{\infty\})^n$
  - $r(i)$ = cost of serving (a.k.a. processing) task r in state $i \in [n]$
    - Forbidden to serve task r in state i if $r(i) = \infty$

Metrical task system

- Instance of *metrical task system (MTS)* problem consists of:
  - Finite MS $\mathcal{M}$
    - Known in advance (part of problem's definition)
  - Request sequence $\sigma = (r_1, \ldots, r_{|\sigma|})$
    - r_j = task over $\mathcal{M}$, reported at time $1 \leq j \leq |\sigma|$
    - Tasks often taken from (known) finite domain $\subset (\mathbb{R}_{\geq 0} \cup \infty)^n$
- Algorithm $\texttt{Alg}$ determines state sequence $\texttt{Alg}[1], \ldots, \texttt{Alg}[|\sigma|] \in [n]$
  - Task r_j served in state $\texttt{Alg}[j]$
  - Online decisions: $\texttt{Alg}[j]$ determined at time j (in response to r_j)
- Alg's cost components:
  - *Processing cost* $= \sum_{j=1}^{|\sigma|} r_j(\texttt{Alg}[j])$
  - *Transition cost* $= \sum_{j=1}^{|\sigma|} \delta(\texttt{Alg}[j-1], \texttt{Alg}[j])$
    - Initial state $\texttt{Alg}[0]$ pre-specified
- Objective: minimize (total) processing cost + (total) transition cost
- Assume hereafter: $\min_{x \neq y \in [n]} \delta(x, y) = 1$
  - WLOG as we can scale distances and processing costs

General framework

- MTS generalizes many online problems
- Paging with N pages and cache size k:
  - States = possible cache configurations
  - Transition cost from cache configuration $C \subseteq [N]$ to cache configuration $C' \subseteq [N]$, $|C| = |C'| = k$, is

$$\delta(C, C') = |C - C'|$$

  - Request for page $i \in [N]$ encoded by task

$$r(C) = \begin{cases} 0, & i \in C \\ \infty, & i \notin C \end{cases}$$

- Not necessarily a "compact representation"
  - In paging, $|V| = \binom{N}{k}$

Switching states along continuous time axis

- Actions of $\texttt{Alg}$ captured by function $\texttt{Alg}[] : \{1, \ldots, |\sigma|\} \rightarrow [n]$
  - Determines state of $\texttt{Alg}$ at each integral time $1 \leq j \leq |\sigma|$
- Convenient to allow $\texttt{Alg}$ to switch states along continuous time axis
- Replace aforementioned function by $\texttt{Alg}[] : [1, |\sigma| + 1) \rightarrow [n]$
  - Determines state of $\texttt{Alg}$ at each real time $t \in [1, |\sigma| + 1)$
- Interpretation for integer $1 \leq j \leq |\sigma|$ and real $t \in [j, j + 1)$:
  $\texttt{Alg}$ serves task r_j in state $\texttt{Alg}[t]$ during time interval $[t, t + dt)$
  - $r_j(\texttt{Alg}[t])\, dt$ = accumulated processing cost during $[t, t + dt)$
- Redefine cost of $\texttt{Alg}$ on σ:
  - Processing cost $= \sum_{j=1}^{|\sigma|} \int_j^{j+1} r_j(\texttt{Alg}[t])\, dt$
  - Charge $\delta(s, s')$ to transition cost whenever $\texttt{Alg}$ switches from state s to state s'
- Generalizes discrete case

Continuous time algorithms are not more powerful

- Transform any continuous time MTS algorithm $\mathtt{Alg}$ to ordinary discrete time MTS algorithm $\mathtt{Alg}'$:
    - $\mathtt{Alg}'[j] = \arg\min_{s \in \{\mathtt{Alg}[t] : t \in [j, j+1)\}} r_j(s)$
- $\mathtt{Alg}'(\sigma) \leq \mathtt{Alg}(\sigma)$:
    - Processing cost: by definition of $\mathtt{Alg}'$
    - Transition cost: by triangle inequality

Periodic state traversals

- *Traversal algorithms* = family of cont. time online MTS algorithms
- `Alg` characterized by state sequence $s_0, \ldots, s_{k-1} \in [n]$
  - Repeated periodically to form infinite state sequence $s_0, s_1, \ldots$
    - $s_i = s_{i \bmod k}$
- Rule of `Alg` for $i = 0, 1, \ldots$:
  1. Serve incoming requests from state s_i as long as accumulated processing cost does not exceed threshold $\delta(s_i, s_{i+1})$
  2. Move to state s_{i+1}
- Observation: processing cost incurred by `Alg` in each period $\leq$

$$\sum_{i=0}^{k-1} \delta(s_i, s_{i+1})$$

  - $<$ if `Alg` encounters processing cost ∞ (state changes immediately)
- What is transition cost incurred by `Alg` in each period?
- Rule attempts to balance between two cost components
- Balancing between different cost components — common design pattern in many online algorithms
  - Recalling ski-rental

Warmup: a 2-State MTS

Observation

The traversal algorithm on a 2-point MS is 4-competitive.

- Why "the traversal algorithm"?
- In each period, total cost of `Alg` $\leq 4 \cdot \delta(1,2) = 4$
- Complete proof by showing that in each period, `Opt` pays ≥ 1
- If `Opt` moves during period, then `Opt`'s transition cost ≥ 1
- Otherwise, `Opt` stays in state 1 or 2 throughout entire period
- $\Longrightarrow$ `Opt`'s processing cost in period ≥ 1
  - `Alg` moved from there ■

MTS with n states

- Idea behind general traversal algorithm inspired by 2-point case:
  1. Partition MS into 2 components
  2. Traverse component 1 recursively until accumulated cost reaches crossing transition cost
  3. Cross to component 2
  4. Traverse component 2 recursively until accumulated cost reaches crossing transition cost
  5. Complete period by crossing back to component 1
- Main challenge: partition of MS
- To design periodic traversal, useful to think of $\mathcal{M}$ as weighted complete undirected graph $G = (V, E, w)$
  - Edge weights $w : E \to \mathbb{R}_{>0}$ obey triangle inequality
  - Rounded weight $w'(e) = 2^{\lceil \lg w(e) \rceil}$
    - Rounded weights do not necessarily obey triangle inequality
- Minimum spanning tree $T \subseteq E$ of G
- Period of traversal = cycle in G
  - Not necessarily simple

The traversal

- Define cycle $\mathcal{C}$ recursively
- If $T = \emptyset$, then $\mathcal{C}$ consists of one node and no edges
- If $T = \{(u, v)\}$, then $\mathcal{C} = (u, v, u)$.
- Assume $|T| > 1$
- Let $e = (u_1, u_2)$ be maximum weight edge in T with $w'(e) = 2^M$
- Removal of e breaks T into trees T_1 and T_2 with u_i in T_i
- $\mathcal{C}_i$ = cycle constructed recursively for T_i
- Maximum weight edge in T_i have w'-weight 2^{M_i}
- Cycle $\mathcal{C}$:
  1. Start at u_1
  2. Perform 2^{M-M_1} periods of $\mathcal{C}_1$
  3. Cross from u_1 to u_2
  4. Perform 2^{M-M_2} periods of $\mathcal{C}_2$
  5. Complete cycle by crossing from u_2 to u_1

An upper bound on the CR

Theorem

The MTS traversal algorithm on an n-point MS is $O(n)$-competitive.

- Identify $\mathcal{C}$, $\mathcal{C}_1$, $\mathcal{C}_2$ with single period

Lemma (Up-bound `Alg`)

The cost of the traversal algorithm in $\mathcal{C}$ is $O(n2^M)$.

Lemma (Low-bound `Opt`)

If M is defined (i.e., $|T| > 0$), then the cost of `Opt` in $\mathcal{C}$ is $\Omega(2^M)$.

- Is theorem (asymptotically) tight?
  - Recall: linear lower bound for deterministic paging algorithms established with $N = k + 1$ pages...

Up-bound cost of `Alg`

- Argue: edge $e' \in T$ with $w'(e') = 2^m$ traversed $O(2^{M-m})$ times in $\mathcal{C}$
  - $\implies$ Total transition cost over $e' = O(2^M)$
  - $\implies$ Total transition cost over $n-1$ edges in $T = O(n2^M)$
- Argument established by induction on $|T|$
- If $|T| = 0$, then holds vacuously
- Assume $|T| > 0$ and let $e = (u_1, u_2)$, T_i, $\mathcal{C}_i$, M_i be as in construction
- e traversed exactly $1 = 2^{M-M}$ time in each direction during $\mathcal{C}$
- Ind. hyp.: edge $e' \in T_i$ with $w'(e') = 2^m$ traversed $O(2^{M_i - m})$ times in $\mathcal{C}_i$
- $\mathcal{C}$ contains 2^{M-M_i} periods of $\mathcal{C}_i$, hence e' traversed

$$2^{M-M_i} \cdot O(2^{M_i-m}) = O(2^{M-m})$$

times ■

Low-bound cost of $\texttt{Opt}$

- Proof by induction on $|T|$
- If $|T| = 0$, then holds vacuously
- Assume $|T| > 0$ and let $e = (u_1, u_2)$, T_i, $\mathcal{C}_i$, M_i be as in construction
- If $\texttt{Opt}$ crosses between T_1 and T_2 during $\mathcal{C}$, then $\texttt{Opt}$ pays transition cost $\geq w(e) > w'(e)/2 = 2^{M-1}$
  - T is a minimum spanning tree
- Assume (WLOG) that $\texttt{Opt}$ stays in T_1 during entire period $\mathcal{C}$
- Case 1: $|T_1| = 0$ and u_1 is the only vertex in T_1
  - $\texttt{Opt}$ serves all requests in period from u_1
  - By definition of $\mathcal{C}$, processing cost from state u_1 of subset of period $\mathcal{C}$
    $$\geq \delta(u_1, u_2) = w(e) > w'(e)/2 = 2^{M-1}$$
- Case 2: $|T_1| > 0$
  - Ind. Hyp.: cost of $\texttt{Opt}$ for each period of $\mathcal{C}_1 = \Omega(2^{M_1})$
  - $\mathcal{C}$ contains 2^{M-M_1} periods of $\mathcal{C}_1$, hence cost of $\texttt{Opt}$ for $\mathcal{C}$
    $$\geq 2^{M-M_1} \cdot \Omega(2^{M_1}) = \Omega(2^M) \blacksquare$$

Aspect ratio and uniform MSs

- Finite MS $\mathcal{M} = (V, \delta)$
- *Aspect ratio* =

$$\frac{\max_{x,y \in V} \delta(x, y)}{\min_{x,y \in V, x \neq y} \delta(x, y)}$$

  - Assuming minimum (positive) distance is 1, aspect ratio = *diameter*
- MS with aspect ratio 1 is called *uniform*
- Algorithm `UnifMTS` =
  randomized online MTS algorithm for uniform $\mathcal{M}$ with CR $O(\log n)$
  - $\mathcal{M}$ is uniform, but tasks are general...

Phases and saturated states

- `UnifMTS` works in continuous time
- Consider request sequence σ
- Partition time line $[1, |\sigma| + 1)$ into finite intervals called *phases*
  - Not to be confused with paging problem's phases
- t_i = start time of phase i
- State $s \in [n]$ is *saturated* (w.r.t. phase i) at time $t > t_i$ if accumulated processing cost of time interval $[t_i, t]$ from state s is ≥ 1
  - Cost we would have paid if served $[t_i, t]$ from s
- Phase i ends (phase $i + 1$ begins) when all states become saturated
- Partition into phases and saturation times depend only on σ
  - Independent of the algorithm

The policy of `UnifMTS`

- At beginning of phase, move to state chosen u.a.r. from $[n]$
  - u.a.r. abbreviates uniformly at random
- As long as phase did not end:
  when current state becomes saturated, move to new state chosen u.a.r. among unsaturated states

The CR of UnifMTS

Theorem

UnifMTS *is* $O(\log n)$*-competitive.*

- Claim: in each phase (other than last), cost of Opt ≥ 1
  - If Opt moves during phase, then its transition cost ≥ 1
  - Otherwise, Opt serves entire phase from some state $s \in [n]$
  - By phase definition, s is saturated when phase ends
  - $\Longrightarrow$ processing cost of Opt ≥ 1
- Assertion established by showing that expected cost of UnifMTS during phase $\leq 2H_n$

Up-bound cost of UnifMTS

- Fix some phase and let $s_1, \ldots, s_n$ be states in non-decreasing order of saturation time
  - Known only in hindsight
- Indicator r.v. X_j, $j \in [n]$: algorithm visits state s_j during phase
- #state transitions of algorithm in phase $= \sum_{j=1}^{n} X_j$
  - Including default transition at beginning of phase
- Fix some $j \in [n]$
- $t(j)$ = latest time in current phase at which algorithm moves and s_j still not saturated
  - Algorithm moves to state s
- By definition of $t(j)$ and s, we have $X_j = 1 \iff s = s_j$
- $\Pr(s = s_j) \leq \frac{1}{n-j+1}$
  - At time $t(j)$, none of states $s_j, s_{j+1}, \ldots, s_n$ is saturated

Proof continues

- So, expected transition cost of `UnifMTS` in phase is

$$\mathbb{E}\left[\sum_{j=1}^{n} X_j\right] = \sum_{j=1}^{n} \Pr(X_j = 1) \leq \sum_{j=1}^{n} \frac{1}{n-j+1} = \sum_{\ell=1}^{n} \frac{1}{\ell} = H_n$$

- Processing cost of `UnifMTS` in phase $\leq$ transition cost
  - Moving out of state s when it becomes saturated ■
- Is upper bound (asymptotically) tight?
  - Recall: logarithmic lower bound for randomized paging algorithms established with $N = k + 1$ pages...
    - Implies in particular uniform MS

Background

- Optimization problems often easier when restricted to special cases
- E.g., MTS admits fairly simple algorithm in uniform MSs
- Naturally, complexity of MTS increases drastically when moving to general MSs
  - Do not obey any special structure (other than triangle inequality)
- General technique allowing us to restrict attention to well structured family of MSs
  - Cost: slightly increased CR

Embedding, contraction, expansion, and distortion

- MSs $\mathcal{M} = (V, \delta)$, $\mathcal{M}' = (V', \delta')$
- Injection $f : V \to V'$ is called an *embedding* of $\mathcal{M}$ in $\mathcal{M}'$
- *Contraction* of f =

$$\sup_{x,y \in V, x \neq y} \frac{\delta(x,y)}{\delta'(f(x), f(y))}$$

- *Expansion* of f =

$$\sup_{x,y \in V, x \neq y} \frac{\delta'(f(x), f(y))}{\delta(x,y)}$$

- *Distortion* of f = product of contraction and expansion
  - Denoted by $\|f\|$
- Embedding is
  - *non-contracting* if contraction of $f \leq 1$
  - *non-expanding* if expansion of $f \leq 1$
  - *isometric* if $\|f\| = 1$
- Does isometric means non-contracting/expanding?

Discussion

- Distortion is always ≥ 1 since

$$\sup_{x,y \in V, x \neq y} \frac{\delta(x,y)}{\delta'(f(x),f(y))} \cdot \sup_{x,y \in V, x \neq y} \frac{\delta'(f(x),f(y))}{\delta(x,y)}$$
$$\geq \sup_{x,y \in V, x \neq y} \left[\frac{\delta(x,y)}{\delta'(f(x),f(y))} \cdot \frac{\delta'(f(x),f(y))}{\delta(x,y)} \right] = 1$$

- Distortion is invariant to scaling distances
- Intuitively, the lower the distortion, the better $\mathcal{M}'$ "approximates" $\mathcal{M}$
  - All distances are similar
- In what follows, restrict attention to finite MSs
- Algorithm designer typically interested in low distortion embeddings of less structured MS families in more structured ones
  - Often makes task of designing algorithms much easier

MSs with bounded aspect ratio

- Observation: n-point MS of aspect ratio ρ can be embedded in n-point uniform MS with distortion ρ
  - How would you construct the embedding?
- Corollary: randomized online MTS algorithm with CR $O(\rho \log n)$ for n-point MS $\mathcal{M}$ of aspect ratio ρ
  - Embed $\mathcal{M}$ in n-point uniform MS $\mathcal{U}$ using embedding f
    - Non-expanding
    - Contraction $\leq \rho$
  - Use f to translate incoming requests from $\mathcal{M}$ to $\mathcal{U}$
  - Run `UnifMTS` on $\mathcal{U}$ and use f^{-1} to translate state transition back to $\mathcal{M}$
  - Contraction of $f \leq \rho \implies \mathtt{UnifMTS}(\mathcal{M}) \leq \rho \cdot \mathtt{UnifMTS}(\mathcal{U})$
  - Guarantee of `UnifMTS` $\implies \mathtt{UnifMTS}(\mathcal{U}) \leq O(\log n) \cdot \mathtt{Opt}(\mathcal{U})$
  - Non-expanding $\implies \mathtt{Opt}(\mathcal{U}) \leq \mathtt{Opt}(\mathcal{M})$

General technique

- General technique for many online problems on MSs:
  1. Design c-competitive online algorithm `Alg` for family $\mathcal{F}$ of well structured MSs
  2. Embed input MS $\mathcal{M}$ in metric space $\mathcal{M}' \in \mathcal{F}$ with distortion d
  3. Translate request sequence from $\mathcal{M}$ to $\mathcal{M}'$ and run `Alg` to serve it
  4. Translate actions of `Alg` back to $\mathcal{M}$
- CR of resulting online algorithm is $c \cdot d$
- Applicable also for offline optimization problems on MSs
- General MSs can have large aspect ratio, so family of uniform metric spaces is not good enough in general
  - Embedding MS of aspect ratio ρ in uniform metric space incurs distortion $\geq \rho$

Tree MSs

- Every MS can be realized by distances in undirected weighted graph
- Natural attempt to "impose structure": restrict graph topology

Definition

MS $\mathcal{M} = (V, \delta)$ is said to be a *tree MS* if it can be embedded isometrically in a MS realized by the distances in some tree; that is, there exists some (edge-)weighted tree $T = (V_T, E_T)$ and an embedding $f : V \to V_T$ such that $\delta(x, y) = \delta_T(f(x), f(y))$ for every $x, y \in V$, where $\delta_T(\cdot, \cdot)$ is the distance function of T. The vertices in $V_T - f(V)$ are referred to as *Steiner vertices*.

- Suffices to consider trees in which internal vertices = Steiner vertices
  - $f(V)$ = leaves of T
  - Delete leaf not in $f(V)$
  - Connect leaf with edge of weight 0 to internal vertex in $f(V)$

Well separated trees

Definition

Consider a weighted tree $T = (V, E, w)$, $w : E \to \mathbb{R}_{>0}$, rooted at node r with leaf set $\mathcal{L}$. We say that T is a *hierarchically well separated tree* with parameter k, or *k-HST* for short, if

- the edge weights increase exponentially by factor $\geq k$ along any leaf-to-root path in T; and
- for every internal vertex $v \in V - \mathcal{L}$ and for every two leaves $x, y \in T_v \cap \mathcal{L}$, it holds that $\delta_T(v, x) = \delta_T(v, y)$.

When $k > 1$ is a constant, we may omit it and write simply *HST*.

- Distance between any two leaves in $\mathcal{L}$ fully determined by identity of least common ancestor
  - Often, k-HST represented as vertex-weighted tree
    - take weight of vertex v to be $\delta_T(v, x)$, where $x \in T_v \cap \mathcal{L}$
- Use term HST also for MS that can be isometrically embedded in leaves of HST

MTS on HSTs

- Tree MSs, in particular HSTs, are well structured and "easy to work with" from algorithmic perspective
  - Many graph theoretic problems become significantly easier on trees

Theorem ([BCLL2021])

There exists an online algorithm for MTS on HSTs with CR $O(\log n)$.

- This algorithm is beyond the scope of the course
- Can we embed any MS in HST with low distortion?

Embedding in tree MSs

Theorem ([RR1998])

Let $\mathcal{M}$ be the MS resulting from the distances in the (unweighted) n-cycle. If f is an embedding of $\mathcal{M}$ in a tree MS, then $\|f\| = \Omega(n)$.

- Trivial if attention restricted to spanning trees of n-cycle
  - Requires more work for general tree MSs
- So cannot hope to embed any MS in tree MS with low distortion
- But not all hope is lost. . .

Embedding in a random MS

Definition

Consider some MS $\mathcal{M} = (V, \delta)$. Let C be a collection of pairs of the form $\langle \mathcal{N}, f \rangle$, where $\mathcal{N} = (V_{\mathcal{N}}, \delta_{\mathcal{N}})$ is a MS and $f : V \to V_{\mathcal{N}}$ is a non-contracting embedding of $\mathcal{M}$ in $\mathcal{N}$. A distribution $\mathcal{D}$ over C is called a *probabilistic embedding* of $\mathcal{M}$ in C. The *distortion* of the probabilistic embedding $\mathcal{D}$ is the smallest d that satisfies

$$\mathbb{E}_{\langle \mathcal{N}, f \rangle \in \mathcal{D}} [\delta_{\mathcal{N}}(f(x), f(y))] \leq d \cdot \delta(x, y)$$

for every $x, y \in V$.

- Often, $V_{\mathcal{N}} = V$ and f is identity function for all $\langle \mathcal{N}, f \rangle \in C$
  - Omit f and refer to C as collection of MSs
  - If $\mathcal{N}$ is tree MS realized by weighted tree T, then $V_{\mathcal{N}} = V$ doesn't mean T cannot include Steiner vertices

Probabilistic embedding in HSTs

Theorem ([FRT2004])

Every n-point MS can be probabilistically embedded in the collection of n-point HSTs with distortion $O(\log n)$. Moreover, the distribution that yields this probabilistic embedding can be sampled efficiently.

- "Breaking" the lower bound of [RR1998]
- Can apply general technique as before with one difference: Instead of embedding $\mathcal{M}$ in deterministically chosen MS $\mathcal{M}'$, embed it in a randomly chosen MS $\mathcal{N}$
  - Called *probabilistically embedding* $\mathcal{M}$ in (random MS) $\mathcal{N}$
  - Resulting online algorithm is randomized even if `Alg` is deterministic

MTS on general MSs

Corollary

There exists a (randomized) online algorithm for MTS on general MSs with competitive ratio $O(\log^2 n)$.

- Arbitrary input MS $\mathcal{M} = (V, \delta)$
- Preprocessing stage: apply [FRT2004] to probabilistically embed $\mathcal{M}$ in random HST $\mathcal{M}' = (V, \delta')$
- Translate request sequence σ over $\mathcal{M}$ to σ' over $\mathcal{M}'$:
  - Task $r \in \sigma$ translated to $r' \in \sigma'$ so that $r'(x) = r(x)$ for every $x \in V$
- $\texttt{Alg}'$ = online MTS algorithm of [BCLL2021]
- Run $\texttt{Alg}'$ on σ' over $\mathcal{M}'$ and whenever $\texttt{Alg}'$ moves to x in $\mathcal{M}'$, move to x in $\mathcal{M}$
  - $\texttt{Alg}$ = resulting online algorithm (operating on σ over $\mathcal{M}$)

Proof continues

- Processing cost of $\texttt{Alg}$ on σ = processing cost of $\texttt{Alg}'$ on σ'
- Embedding of $\mathcal{M}$ in $\mathcal{M}'$ is non-contracting $\Longrightarrow$
  transition cost of $\texttt{Alg}$ on $\sigma \leq$ transition cost of $\texttt{Alg}'$ on σ'
- Thus, $\texttt{Alg}(\sigma) \leq \texttt{Alg}'(\sigma')$
- π = (time dependent) trajectory that realizes $\texttt{Opt}(\sigma)$
- $\texttt{Opt}'(\sigma')$ = cost of serving σ' over $\mathcal{M}'$ by trajectory π
  - $\texttt{Opt}'(\sigma') \geq \texttt{Opt}(\sigma')$
- Since processing cost component of $\texttt{Opt}'(\sigma')$ = that of $\texttt{Opt}(\sigma)$, [FRT2004] ensures

$$\mathbb{E}[\texttt{Opt}'(\sigma')] \leq O(\log n) \cdot \texttt{Opt}(\sigma)$$

  - Expectation over random choice of $\mathcal{M}'$
- Guarantee of $\texttt{Alg}'$: $\texttt{Alg}'(\sigma') \leq O(\log n) \cdot \texttt{Opt}(\sigma') + \beta$

Proof continues

- Combining the above,

$$\begin{aligned}\mathbb{E}[\texttt{Alg}(\sigma)] &\leq \mathbb{E}[\texttt{Alg}'(\sigma')] \\ &\leq O(\log n) \cdot \mathbb{E}[\texttt{Opt}(\sigma')] + \mathbb{E}[\beta] \\ &\leq O(\log n) \cdot \mathbb{E}[\texttt{Opt}'(\sigma')] + \mathbb{E}[\beta] \\ &\leq O(\log^2 n) \cdot \texttt{Opt}(\sigma) + \mathbb{E}[\beta]\end{aligned}$$

- Assertion follows since $\mathbb{E}[\beta]$ doesn't depend on σ
  - Charge to additive constant ∎
- Is upper bound tight?

Theorem ([BCR2023])

There exist n-point MSs for which the CR of any (randomized) MTS algorithm is $\Omega(\log^2 n)$.

Online Steienr Tree

Algorithms in Uncertain Scenarios (097280)

Yuval Emek

Spring 2024/5

Offline Steiner tree

- Input:
  - Connected undirected graph G with edge cost function $c : E(G) \to \mathbb{R}_{>0}$
  - A set $U \subseteq V(G)$ of *terminals*
- Output: subgraph H of G that satisfies
  - H is connected
  - $V(H) \supseteq U$
- Objective: minimize $c(H) = c(E(H))$
- $V(H)$ may be a strict superset of U
  - Vertices in $V(H) - U$ are called *Steiner vertices*

Discussion

- Can H contain cycles?
  - Remove edges from cycles without affecting connectivity
  - H is a Steiner tree
  - Problem equivalent to minimum spanning tree if $U = V$
- Problem is known to be NP-hard
  - Decision version is one of Karp's 21 NP-complete problems
  - APX-complete (no PTAS unless P = NP) [Bern, Plassmann 1989]
  - State of the art approximation ratio is $\ln 4 + \epsilon < 1.39$ [Byrka, Grandoni, Rothvoss, and Laura Sanita 2010]
- Can H include *non-metric edges*?
  - Edge $e \in E(G)$ is metric if $c(e) = \delta_G(e)$
  - Replace non-metric edge (x, y) in H by path realizing $\delta_G(x, y)$

Online Steiner tree

- Graph G and edge cost function c are known in advance
- Terminal set U is provided online
  - Terminal u_t is reported at time t for $t = 1, \ldots, k$
- Online algorithm `Alg` grows Steiner tree in online fashion
  - Adding (but not removing) vertices and edges gradually
- At time $1 \leq t \leq k$, `Alg` constructs subgraph H_t of G that satisfies:
  - H_t is connected
  - $V(H_t) \supseteq \{u_1, \ldots, u_t\}$
  - H_t is a supergraph of H_{t-1}
- Objective: minimize $c(H_k)$

Discussion

- W.l.o.g. H_t is a tree
  - Remove edges from cycles without affecting connectivity
- Alternative view:
  - `Alg` maintains (growing) Steiner tree H
  - At time t, `Alg` augments H with a path P_t connecting it to u_t
    - Lazy algorithm: no point in adding vertices/edges ahead of time
  - $c(P_t)$ = cost paid by `Alg` at time t
- Restrict attention to competitiveness without additive constant
  - n is inherent upper bound on length of request sequence
  - Additive constant can swallow `Alg`'s whole cost

Preprocessing

- Develop randomized online Steiner tree algorithm `Alg`
- `Alg` also relies on low distortion prob. embedding in HST, but design (and analysis) follow different approach
- In preprocessing stage, employ FRT to prob. embed G, c in (random) HST T with edge weight function $w : E(T) \to \mathbb{R}_{>0}$
  - leaf set of T is $V(G)$
  - $\delta_T(x, y) \geq \delta_G(x, y)$ for every $x, y \in V(G)$
    - $\delta_T(\cdot, \cdot)$ *dominates* $\delta_G(\cdot, \cdot)$
  - $\mathbb{E}\left(\delta_T(x, y)\right) \leq O(\log n) \cdot \delta_G(x, y)$ for every $x, y \in V(G)$
- T doesn't even have to be HST:
  suffices that edge weights increase exponentially along leaf-root paths

Preliminaries

- For vertex $v \in V(T)$, let $x(v)$ be leaf in T_v closest to v
  - Break ties arbitrarily but consistently
- For edge $e = (u, v) \in E(T)$, let $M(e)$ be shortest $(x(u), x(v))$-path in G
  - Break ties arbitrarily but consistently
  - Extend definition to edge subsets $F \subseteq E(T)$: $M(F) = \bigcup_{e \in F} M(e)$
    - Union in graphical sense (vertices and edges)
  - Refer to $M(e)$ and $M(F)$ as the *mirrors* of e and F, respectively, in G
- For vertices $x, y \in V(G)$, let $T(x, y)$ be unique (x, y)-path in T
  - Recall that x, y are leaves in T
  - Extend definition to vertex subsets $W \subseteq V(G)$: $T(W) = \bigcup_{x,y \in W} T(x, y)$
    - $=$ unique minimal subtree of T that spans W

Online algorithm

- Subgraph H_t of G defined to be mirror of $E(T(\{u_1, \ldots, u_t\}))$ in G, breaking cycles arbitrarily but consistently
- Alternative description:
  `Alg` maintains Steiner tree H in G and upon arrival of terminal u_t:
  - Let Q be unique path in T that connects u_t to $T(\{u_1, \ldots, u_{t-1}\})$
  - Augment H with $M(E(Q))$, breaking cycles arbitrarily
- `Alg` is randomized although online component is deterministic
  - Due to random construction of T

Proof's outline

Theorem

$\texttt{Alg}$ *is* $O(\log n)$*-competitive.*

- Let $T^* = T(\{u_1, \ldots, u_k\})$
  - Recall: $E(H) \subseteq M(E(T^*))$

Lemma (Low-bound $\texttt{Opt}$)

$\mathbb{E}(w(T^*)) \leq O(\log n) \cdot c(\texttt{Opt})$.

Lemma (Up-bound $\texttt{Alg}$)

$c(H) \leq O(1) \cdot w(T^*)$.

- Theorem established by combining two lemmas:

$$\mathbb{E}(c(\texttt{Alg})) = \mathbb{E}(c(H)) \leq O(1) \cdot \mathbb{E}(w(T^*)) \leq O(\log n) \cdot c(\texttt{Opt})$$

Low-bound $\texttt{Opt}$

Lemma

$\mathbb{E}(w(T^*)) \le O(\log n) \cdot c(\texttt{Opt})$.

- $\texttt{Opt}_T = \bigcup_{e \in E(\texttt{Opt})} T(e)$
  - Connected since $\texttt{Opt}$ is connected
  - Spans $\{u_1, \ldots, u_k\}$
    - Since $\texttt{Opt}$ spans $\{u_1, \ldots, u_k\}$ in G
- $\implies E(\texttt{Opt}_T) \supseteq E(T^*) \implies w(\texttt{Opt}_T) \ge w(T^*)$
- Construction of T ensures that

$$\mathbb{E}(w(T(e))) = \mathbb{E}(\delta_T(e)) \le O(\log n) \cdot c(e)$$

  for every edge $e \in E(G)$
  - Holds in particular for edges $e \in E(\texttt{Opt})$

Proof continues

- Therefore,

$$\begin{aligned}
\mathbb{E}(w(T^*)) &\leq \mathbb{E}(w(\mathtt{Opt}_T)) \\
&= \mathbb{E}\left(w\left(\bigcup_{e \in E(\mathtt{Opt})} T(e)\right)\right) \\
&\leq \mathbb{E}\left(\sum_{e \in E(\mathtt{Opt})} w(T(e))\right) \\
&= \sum_{e \in E(\mathtt{Opt})} \mathbb{E}\left(w(T(e))\right) \\
&= \sum_{e \in E(\mathtt{Opt})} \mathbb{E}\left(\delta_T(e)\right) \\
&\leq O(\log n) \cdot \sum_{e \in E(\mathtt{Opt})} c(e) \\
&= O(\log n) \cdot c(\mathtt{Opt}) \blacksquare
\end{aligned}$$

Up-bound `Alg` — auxiliary claim

Claim

$\delta_T(x(u), x(v)) \leq O(1) \cdot w(e)$ for every edge $e = (u, v) \in E(T)$.

- W.l.o.g. $u = p(v)$
- $\delta_T(x(u), u) \leq \delta_T(x(v), u)$
  - By definition of $x(\cdot)$
- $\delta_T(x(v), u) = \delta_T(x(v), v) + w(e) \leq O(1) \cdot w(e)$
  - By exponential growth of edge weights along $(x(v), u)$-path in T
    - Due to convergence of geometric sums
- Combine together:

$$\begin{aligned} \delta_T(x(u), x(v)) &\leq \delta_T(x(u), u) + \delta_T(x(v), u) \\ &\leq 2 \cdot \delta_T(x(v), u) \leq O(1) \cdot w(e) \blacksquare \end{aligned}$$

- Distance is 0 if $x(u) = x(v)$

Up-bound Alg

Lemma

$c(H) \leq O(1) \cdot w(T^*)$.

- Since $\delta_T(\cdot,\cdot)$ dominates $\delta_G(\cdot,\cdot)$, it follows that

$$c(M(e)) = \delta_G(x(u), x(v)) \leq \delta_T(x(u), x(v)) \leq O(1) \cdot w(e)$$

  for every edge $e = (u, v) \in E(T)$
    - Last transition holds by claim
    - Holds in particular for edges $e = (u, v) \in E(T^*)$
- $E(H) \subseteq M(E(T^*))$, thus

$$c(H) \leq c(M(E(T^*))) = c\left(\bigcup_{e \in E(T^*)} M(e)\right)$$

$$\leq \sum_{e \in E(T^*)} c(M(e)) \leq O(1) \cdot \sum_{e \in E(T^*)} w(e) = O(1) \cdot w(T^*) \blacksquare$$

The two player zero sum game view

- So far, "tailor made" lower bounds for randomized online algorithms
- Today, a more generic approach called Yao's principle
- Consider some online minimization problem $\mathcal{P}$
  - Can be applied to maximization problems as well
- Fix (arbitrarily large) upper bound n on length of request sequence
- $\mathcal{I}$ = collection of request sequences for $\mathcal{P}$ (length $\leq n$)
- $\mathcal{A}$ = collection of deterministic online algorithms for $\mathcal{I}$
- Assume $\mathcal{I}$ and $\mathcal{A}$ are finite
  - (Can be made) true for most reasonable online problems
- Two player zero sum game:
  - (Algorithm) designer chooses $\mathtt{Alg} \in \mathcal{A}$
  - Adversary chooses $\sigma \in \mathcal{I}$
  - Designer pays $\chi(\mathtt{Alg}, \sigma) = c(\mathtt{Alg}(\sigma))/c(\mathtt{Opt}(\sigma))$ to adversary
- (Deterministic) $\mathtt{Alg}$ is α-competitive iff $\chi(\mathtt{Alg}, \sigma) \leq \alpha$ for any $\sigma \in \mathcal{I}$
  - Ignore additive constant for now

Randomized algorithms

- Key observation: randomized algorithm = mixed strategy of designer
  - Probability distribution $p \in \Delta(\mathcal{A})$
- p is α-competitive iff $\chi(p, \sigma) \leq \alpha$ for any $\sigma \in \mathcal{I}$
  - $\chi(p, \sigma) = \sum_{\texttt{Alg} \in \mathcal{A}} p(\texttt{Alg}) \cdot \chi(\texttt{Alg}, \sigma)$
  - $\chi(\texttt{Alg}, q) = \sum_{\sigma \in \mathcal{I}} q(\sigma) \cdot \chi(\texttt{Alg}, \sigma)$
- Minimax theorem [von Neumann 1928]: game has value v:

$$\exists p \in \Delta(\mathcal{A}) \text{ s.t. } \forall \sigma \in \mathcal{I}, \quad \chi(p, \sigma) \leq v$$
$$\exists q \in \Delta(\mathcal{I}) \text{ s.t. } \forall \texttt{Alg} \in \mathcal{A}, \quad \chi(\texttt{Alg}, q) \geq v$$

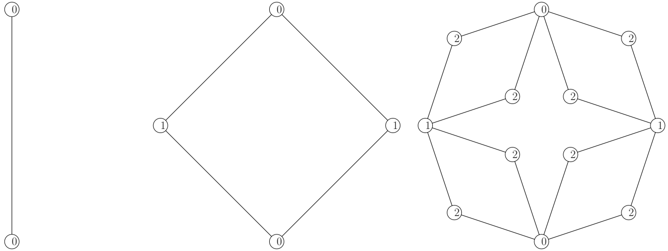
  - With mixed strategies, each player can guarantee game's value!
    - Doesn't matter who plays first
- [Yao 1977]: If we can find $q \in \Delta(\mathcal{I})$ such that $\chi(\texttt{Alg}, q) \geq \tilde{v}$ for every deterministic Alg, then c.r. of any randomized algorithm $\geq \tilde{v}$
  - Challenge: design bad $q \in \Delta(\mathcal{I})$
    - $q = q(n)$ for arbitrarily large n

Discussion

- What about the additive constant?
  - Cannot "help" algorithm designer as n can be arbitrarily large
- Yao's principle developed originally for randomized RAM algorithms
  - Applied to online algorithms by [Borodin, Linial, Saks 1992]

The diamond graph

- [Imase, Waxman 1991]: lower bound for online Steiner tree based on Yao's principle
- Family $\{D_j\}_{j \geq 0}$ of *diamond graphs* defined inductively:
  - D_0 = graph consisting of 2 vertices and 1 edge
    - Vertices of D_0 are called *poles*
  - D_{j+1} obtained from D_j by replacing each edge $(x, y) \in E(D_j)$ by paths (x, z_e^1, y) and (x, z_e^2, y)
    - z_e^1 and z_e^2 are fresh vertices
    - x and y are called *parents* of z_e^1 and z_e^2



The diamond graph — cont.

- $|E(D_j)| = 4 \cdot |E(D_{j-1})| = 4^j$
- Length of any pole-to-pole path in D_j is 2^j
- Vertices introduced at inductive construction of D_i are of *level* i
  - Parents of level i vertex are of levels $i-1$ and $\leq i-1$
    - $\lneqq i-1$ if $i > 1$
- Fix some level $1 \leq i \leq j$
  - #level i vertices in any pole-to-pole path in D_j is 2^{i-1}
    - $=$ length of pole-to-pole path in D_{i-1}
  - #level i vertices in D_j is $2 \cdot |E(D_{i-1})| = 2 \cdot 4^{i-1}$
- $|V(D_j)| = 2 + \sum_{i=1}^{j} 2 \cdot 4^{i-1} = 2 + \frac{2}{3}(4^j - 1) = \Theta(4^j)$
- Assign costs to edges: $c(e) = 2^{-j}$ for every edge $e \in E(D_j)$
  - Cost of each pole-to-pole path in D_j is 1
  - Distance between level i vertex and any of its parents is 2^{-i}
    - Direct edge (of cost 2^{-i}) in D_i
    - Distance remains unchanged in D_j for $j > i$

A bad probability distribution

- Bad prob. distribution over online Steiner tree instances in D_j:
  - Pick a pole-to-pole path P u.a.r.
  - Instance σ: expose vertices in P by non-decreasing order of levels
    - 2 poles first
    - then 1 level 1 vertex
    - then 2 level 2 vertices
    - then 4 level 3 vertices
    - ...
- By definition of D_j and cost function $c(\cdot)$, $\texttt{Opt}(\sigma) = 1$
- Prove:

$$\mathbb{E}(\texttt{Alg}(\sigma)) \geq \Omega(j) = \Omega(\log |V(D_j)|)$$

  for any deterministic online algorithm Alg
  - Yao's principle:
    c.r. of any randomized online Steiner tree algorithm $\geq \Omega(\log n)$

Low-bounding expected payment of $\texttt{Alg}$

- H = Steiner tree of D_j maintained by $\texttt{Alg}$
- Vertices x and x' of levels i and i' are *well-connected* in H if all internal vertices of unique (x, x')-path in H are of levels $> \max\{i, i'\}$
  - Must be a shortest (x, x')-path in D_j
- Fix some $1 \leq i \leq j$
- Consider arrival of some level i vertex z with parents x and y
- If x and y are not well-connected in H, then $\texttt{Alg}$ well-connects z to one of them, paying 2^{-i}
- Assume: when z arrives, x and y are well-connected by path Q
- Key observation: $\mathbb{P}(z \in V(Q)) = 1/2$
  - If $z \notin V(Q)$, then $\texttt{Alg}$ well-connects z to x or y, paying 2^{-i}
- In both cases, expected payment of $\texttt{Alg}$ when z arrives is $\Omega(2^{-i})$
- Summing over all levels $1 \leq i \leq j$:

$$\mathbb{E}(\texttt{Alg}(\sigma)) \geq \sum_{i=1}^{j} 2^{i-1} \cdot \Omega(2^{-i}) = \Omega(j) \blacksquare$$

Online Algorithms with Machine Learned Predictions

Algorithms in Uncertain Scenarios (097280)

Yuval Emek

Spring 2024/5

Beyond worst case analysis

- Worst case nature of competitive analysis may be too harsh
- In practice, many online algs outperform theoretical guarantees
  - Rarely encounter pitfalls that realize competitiveness lower bounds
- Extensive research on "beyond worst case" analysis of online algs
  - **Restrict power of adversary**
    - random arrival order
    - locality of reference
    - access graph
    - smoothed analysis
    - diffused adversaries
  - **Relax definition of competitive analysis**
    - resource augmentation
    - loose competitiveness
  - **Modify computational model**
    - lookahead
    - small advice
- **This lecture:** augment online algs with machine learned predictions [Lykouris, Vassilvitskii 2021]

Machine learned predictions

- ML successful in many application domains
- Good on average assuming test distribution = train distribution
  - Almost no worst case guarantees
- Can ML help online algs?
- **Idea:** provide online alg with ML predictions on future requests
  - Likely to encounter prediction errors
- **Goal (informally):**
  - Small competitive ratio when predictions are correct
  - Competitive ratio deteriorates slowly as prediction error increases

Machine learning preliminaries

- Prediction model defined over
  - *feature space* $\mathcal{X}$
  - *label space* $\mathcal{Y}$
- *Example* = pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- **Assumption:** if $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$ is training/test distribution, then $(x, y), (x, y') \in \text{support}(\mathcal{D})$ implies $y = y'$
- *Predictor* (a.k.a. hypothesis) = mapping $h : \mathcal{X} \to \mathcal{Y}$
- Performance of predictor measured w.r.t. *loss function*
$$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$$
  - **This lecture:** restrict attention to *absolute loss* $\ell(y, y') = \|y - y'\|_1$
    - $\mathcal{Y}$ assumed to be vector space
- **Goal**: if h trained on $\mathcal{D}$, then $\mathbb{E}_{(x,y)\sim\mathcal{D}}(\ell(h(x), y))$ is small
  - Does not provide guarantees for all $(x, y) \in \text{support}(\mathcal{D})$
  - Does not provide any guarantees if h trained on $\mathcal{D}' \neq \mathcal{D}$

Online Algorithms with predictions

- Online (minimization) problem P defined over universe $\mathcal{Z}$ of elements
- Prediction model defined over feature space $\mathcal{X}$ and label space $\mathcal{Y}$
- Request sequence $\sigma = (\sigma_1, \dots, \sigma_{|\sigma|})$
- Each request σ_t associated with
  - element $z(\sigma_t) \in \mathcal{Z}$
    - paging: requested page
  - feature $x(\sigma_t) \in \mathcal{X}$
    - captures info available to predictor
    - paging: e.g., process from which σ_t arrives and its state
  - label $y(\sigma_t) \in \mathcal{Y}$
    - captures info sufficient to solve P (almost) optimally
    - paging: next arrival time of $z(\sigma_t)$
- Online alg has (oracle) access to predictor $h : \mathcal{X} \to \mathcal{Y}$
  - Provides *predicted label* $h(\sigma_t) = h(x(\sigma_t))$ for each request σ_t
- $\mathtt{Alg}_h(\sigma)$ = (expected) cost of $\mathtt{Alg}$ on σ using h's predictions

Prediction error

- Define *error* of predictor h on request sequence σ (w.r.t. ℓ) as
$$\eta(h, \sigma) = \sum_{t=1}^{|\sigma|} \ell\left(h(\sigma_t), y(\sigma_t)\right)$$
- Predictor h is *χ-accurate* for P if
$$\eta(h, \sigma) \leq \chi \cdot \texttt{Opt}(\sigma)$$
for every request sequence σ
  - Bounding error relatively to $\texttt{Opt}(\sigma)$
- Relatively to $\texttt{Opt}(\sigma)$ rather than $|\sigma|$?
  - $|\sigma|$ can be extended "artificially"
  - $\texttt{Alg}(\sigma)$ is also bounded w.r.t. $\texttt{Opt}(\sigma)$
- $\mathcal{H}_P(\chi)$ = class of χ-accurate predictors for P
- Online alg is *χ-assisted* if $\texttt{Alg}$'s predictor $\in \mathcal{H}_P(\chi)$
  - $\texttt{Alg}$'s designer typically oblivious to χ

The competitive ratio of χ-assisted algorithms

- Define
$$\texttt{Alg}_{\chi}(\sigma) = \sup_{h \in \mathcal{H}_P(\chi)} \texttt{Alg}_h(\sigma)$$
- χ-assisted $\texttt{Alg}$ is *c-competitive* if
$$\texttt{Alg}_{\chi}(\sigma) \leq c \cdot \texttt{Opt}(\sigma) + \alpha$$
for any request sequence σ
  - $\alpha \in \mathbb{R}_{\geq 0}$ is additive constant
    - independent of σ and χ
- CR of χ-assisted $\texttt{Alg}$ as function of χ = *robustness* (function) of $\texttt{Alg}$
  - Denote by $\varrho(\chi)$
- **Holy grail:**
  - $\varrho(\chi)$ grows slowly
  - $\varrho(0)$ is small constant
    - a.k.a. *consistency*
  - $\limsup_{\chi \to \infty} \varrho(\chi)$ not (asymptotically) larger than best CR of P
    - a.k.a. *robust* $\texttt{Alg}$

The setting

- *Paging* over N pages with cache size k
- Request sequence $\sigma = (\sigma_1, \ldots, \sigma_{|\sigma|})$
- $z(\sigma_t)$ = page associated with σ_t
- For $1 \leq t \leq |\sigma|$ and $p \in [N]$, define
$$\mathrm{NAT}(p, t) = \min\left(\{t < t' \leq |\sigma| : z(\sigma_{t'}) = p\} \cup \{|\sigma| + 1\}\right)$$
- $y(\sigma_t)$ = label associated with $\sigma_t = \mathrm{NAT}(z(\sigma_t), t)$
- $h(\sigma_t) = h(x(\sigma_t))$ = predicted label associated with σ_t
- C_t = cache configuration of `Alg` at time t
- **Simplifying assumption:** $C_1 = \emptyset$
  - `Alg` holds prediction for each page in cache
- For $1 \leq t \leq |\sigma|$ and $p \in [N]$, define
$$\mathrm{NAT}_h(p, t) = h(\sigma_{t'}),$$
where $t' \leq t$ is largest index satisfying $z(\sigma_{t'}) = p$
  - If no such t' exists, then $\mathrm{NAT}_h(p, t) = \perp$
  - Notice: $\mathrm{NAT}_h(p, t) \neq \perp$ for every $1 \leq t \leq |\sigma|$ and $p \in C_t$
  - Prediction errors may cause $\mathrm{NAT}_h(p, t) \leq t$
- Restrict attention to *lazy algs* (act only on PFs)

Following the predictions blindly

- **Belady rule:** on PF at time t, evict $\text{argmax}_{p \in C_t} \text{NAT}_h(p, t)$
- If $\eta(h, \sigma) = 0$, then following Belady rule is optimal [Belady 1966]
- Should we "blindly follow" Belady rule if $\eta(h, \sigma)$ may be positive?
  - Variants depending on treatment of $\text{NAT}_h(p, t) \leq t$
- Simple examples demonstrate that robustness is $\Omega(1 + \chi)$
  - Unattractive dependency on χ (see later)
  - Not robust (take $\chi \gg \log k$)

Recalling marking algorithm

- Request sequence σ (inductively) partitioned into *phases*
  - each phase is maximal subsequence that includes $\leq k$ distinct pages
- All pages are *unmarked* at beginning of phase
- Page is *marked* upon request
- `Alg` never evicts marked page
- $C_1 = \emptyset$ implies trivial first phase (no PFs)

Stale vs. fresh vs. fading

- Consider page $p \in [N]$ and phase $\phi > 1$
  - p is *stale* if p requested in (both) phases $\phi - 1$ and ϕ
  - p is *fresh* if p not requested in phase $\phi - 1$ and requested in phase ϕ
  - p is *fading* if p requested in phase $\phi - 1$ and not requested in phase ϕ
  - Cf. definitions in paging presentation
  - Does not depend on `Alg`
- PF request σ_t in phase $\phi > 1$ is *stale/fresh* if $z(\sigma_t)$ is stale/fresh
  - Fading pages not requested at all
  - Does depend on `Alg`
- In each phase $\phi > 1$
  - $\#$stale pages $+$ $\#$fresh pages $= k$
  - $\#$fresh pages $=$ $\#$fading pages

Blame chains

- Consider marking `Alg` and phase $\phi > 1$
- During ϕ, maintain *blame chains* that record evicted pages
  - Blame chains of ϕ, deleted when ϕ ends
  - Page evicted exactly once, hence assigned to unique blame chain
- **Invariant:**
  throughout ϕ, evicted pages not requested yet = tails of blame chains
- Upon arrival of fresh (PF) request σ_t
  1. create (empty) blame chain $B = B(\sigma_t)$
  2. $B_1 \leftarrow$ page evicted in σ_t
- Upon arrival of stale (PF) request σ_t
  1. Let B be blame chain such that $B_{|B|} = z(\sigma_t)$
     - well defined by invariant as σ_t is stale
  2. $B_{|B|+1} \leftarrow$ page evicted in σ_t
- When B is completed
  - B_j is stale for every $1 \leq j < |B|$
  - $B_{|B|}$ is fading
- **Important:** blame chains can be maintained online

The χ-assisted online algorithm

- Marking alg `PredMark`
  - M_t = set of marked pages at time t
  - Maintaining blame chains implicitly

1. let σ_t be PF request
2. if σ_t is fresh, then evict $\text{argmax}_{p \in C_t - M_t} \text{NAT}_h(p, t)$
3. if σ_t is stale, then
   1. let B be blame chain such that $B_{|B|} = z(\sigma_t)$
      - well defined by invariant
   2. if $|B| < H_k$, then evict $\text{argmax}_{p \in C_t - M_t} \text{NAT}_h(p, t)$
      - H_k = kth harmonic number
   3. else, evict $p \in_R C_t - M_t$

Theorem

The robustness of `PredMark` *is (up-bounded by)*
$\varrho(\chi) = O\left(\min\left\{1 + \sqrt{\chi}, \log k\right\}\right)$.

Global view

- F = #fresh requests throughout σ
  - PredMark has exactly F blame chains

Lemma

$F/2 \leq \texttt{Opt}(\sigma) \leq F$.

- 1st inequality: established in analysis of RandMark
- 2nd inequality: offline marking alg that evicts only fading pages ■
- $\texttt{PredMark}_h(\sigma)$ = expected total length of blame chains throughout σ
- **Goal:** bound $\texttt{PredMark}_h(\sigma)$ w.r.t. F

Blame chain's prefix and suffix

- Consider blame chain B
- Partition B into
  - *prefix* B^p of length $L^p = \min\{|B|, \lceil H_k \rceil\}$
  - (possibly empty) *suffix* B^s of length $L^s = |B| - L^p$
    - $L^s = \max\{|B| - \lceil H_k \rceil, 0\}$
- **Intuitively:** deterministic B^p vs. probabilistic B^s
- Request σ_t *affects* B^p if $q = z(\sigma_t)$ appended to B^p at time $t^* > t$ and $z(\sigma_{t'}) \neq q$ for all $t < t' \leq t^*$
  - $\text{NAT}_h(q, t^*) = h(\sigma_t)$
  - σ_t necessarily belongs to earlier phase
- $\pi(B)$ = total error of h on requests that affect B^p

Outline

Proposition (prefix' length)

If $\pi(B) \leq \pi$, *then* $L^p \leq 1 + \sqrt{5\pi}$.

Proposition (expected suffix' length)

$\mathbb{E}(L^s \mid \pi(B) \leq \pi) \leq \lceil H_k \rceil$.[a]

[a] Bound on $\mathbb{E}(L^s)$ holds regardless of $\pi(B)$.

Corollary

$\mathbb{E}(L^p + L^s \mid \pi(B) \leq \pi) \leq 2 \cdot \min \left\{1 + \sqrt{5\pi}, \lceil H_k \rceil \right\}$.

- If $1 + \sqrt{5\pi} < \lceil H_k \rceil$, then follows from 1st proposition
  - Implies $L^p < \lceil H_k \rceil \implies L^s = 0$
- If $1 + \sqrt{5\pi} \geq \lceil H_k \rceil$, then follows from 2nd proposition
  - Recall $L^p \leq \lceil H_k \rceil$ ∎

Outline — cont.

Corollary

If the total error of h on σ is η, then
$\texttt{PredMark}_h(\sigma) \leq 2F \cdot \min\left\{1 + \sqrt{5\eta/F}, \lceil H_k \rceil\right\}$.

- $g(\pi) = 2 \cdot \min\left\{1 + \sqrt{5\pi}, \lceil H_k \rceil\right\}$ is concave, hence $\texttt{PredMark}_h(\sigma)$ is maximized by dividing η equally over F blame chains ■
- Theorem follows by plugging $2 \cdot \texttt{Opt} \rightarrow F$ and $1/\texttt{Opt} \rightarrow 1/F$

1. Augmenting Online Algorithms with Unreliable Predictions

2. Online Paging with Predictions
   - A χ-Assisted Online Paging Algorithm
   - Analysis of `PredMark`
     - Bounding Expected Suffix' Length
     - Bounding Prefix' Length

Proving the proposition

- Consider phase $\phi > 1$ and blame chain B of `PredMark` in ϕ
  - prefix B^p of length L^p; suffix B^s of length L^s
- **Show:** $\mathbb{E}(L^s \mid \pi(B) \leq \pi) \leq \lceil H_k \rceil$
- For stale page q, let $\tau(q)$ be 1st arrival time of q in ϕ
- $f = \#$fresh pages of ϕ
- $D =$ set of fading pages of ϕ
- If $B^p \cap D \neq \emptyset$, then $L^s = 0$ w.p. 1
- Assume $B^p \cap D = \emptyset$ and let $t_0 = \tau(B_{\lceil H_k \rceil})$
  - time of first random choice made for B
- $\{q_1, \ldots, q_w\} =$ set of stale pages $q_i \in C_{t_0}$ with $\tau(q_i) > t_0$
  - ordered so that $\tau(q_i) < \tau(q_{i+1})$
  - stale pages candidates for inclusion in B^s
  - $w \leq k - f$

Key lemma

Lemma

$\mathbb{P}(q_i \in B^s \mid \pi(B) \leq \pi) \leq \frac{1}{w+2-i}$ *for each* $1 \leq i \leq w$.

- Consider page q_i for $1 \leq i \leq w$
- Let $t \geq t_0$ be the latest time $< \tau(q_i)$ such that $z(\sigma_t) = B_{|B|}$
  - If q_i not evicted in σ_t, then $q_i \notin B^s$
- Any page $q \in \{q_j \mid j \geq i\} \cup D$ may be evicted in σ_t as long as $q \in C_t$
  - one such page picked u.a.r.
- **Observation:** $|(\{q_j \mid j \geq i\} \cup D) - C_t| \leq f - 1$
  - Each blame chain $B' \neq B$ "steals" ≤ 1 page from $\{q_j \mid j \geq i\} \cup D$
  - $f - 1$ such blame chains
- $|D| = f$, hence
  $|(\{q_j \mid j \geq i\} \cup D) \cap C_t| \geq w + 1 - i + f - (f - 1) = w + 2 - i$ ■
- Prop. follows as
  $\mathbb{E}(L^s \mid \pi(B) \leq \pi) \leq 1 + \sum_{i=1}^{w} \frac{1}{w+2-i} = H_{w+1} \leq \lceil H_k \rceil$

1. Augmenting Online Algorithms with Unreliable Predictions

2. Online Paging with Predictions
   - A χ-Assisted Online Paging Algorithm
   - Analysis of `PredMark`
     - Bounding Expected Suffix' Length
     - Bounding Prefix' Length

Proving the proposition

- Consider phase $\phi > 1$ and blame chain B of `PredMark` in ϕ
  - prefix B^p of length L^p with error $\pi(B)$
- **Show:** if $\pi(B) \leq \pi$, then $L^p \leq 1 + \sqrt{5\pi}$

Lemma (spread of absolute loss [Lykouris, Vassilvitskii 2021])

Let

$$\mathcal{R}_n = \{(r_1, \ldots, r_n) \in \mathbb{Z}_{>0} \mid r_1 < \cdots < r_n\}$$

and

$$\mathcal{S}_n = \{(s_1, \ldots, s_n) \in \mathbb{R}_{>0} \mid s_1 \geq \cdots \geq s_n\}.$$

For any $\gamma \geq 0$ *and* $n > 1 + \sqrt{5\gamma}$, *if* $(r_1, \ldots, r_n) \in \mathcal{R}_n$ *and* $(s_1, \ldots, s_n) \in \mathcal{S}_n$, *then*

$$\sum_{j=1}^{n} |r_j - s_j| > \gamma.$$

Proving the proposition — cont.

- Let $q_j = B_j$ for $1 \leq j \leq L^p$
- q_0 = fresh page whose arrival responsible for creating B
- t_ϕ = time right before ϕ starts
- For $0 \leq j \leq L^p$, define
  - $\psi(j) = \text{NAT}(q_j, t_\phi)$
    - first arrival time of page q_j in ϕ for $0 \leq j < L^p$
    - page q_j may be fading (not arriving during ϕ) for $j = L^p$
  - $\psi_h(j) = \text{NAT}_h(q_j, t_\phi)$
- **Observation 1:**
  $\psi(j) < \psi(j+1)$ for every $0 \leq j < L^p$
  - q_{j+1} evicted by marking algorithm in $\sigma_{\psi(j)}$
- **Observation 2:**
  $\text{NAT}_h(q_{j+1}, \psi(j)) \geq \text{NAT}_h(q_i, \psi(j))$ for every $0 \leq j < i \leq L^p$
  - q_{j+1} selected for eviction by `PredMark`$_h(\sigma)$ in $\sigma_{\psi(j)}$
- **Observation 3:**
  $\text{NAT}_h(q_i, \psi(j)) = \psi_h(i)$ for every $0 \leq j < i \leq L^p$
  - prediction on q_i does not change during $(t_\phi, \psi(j)]$

Proving the proposition — cont.

- So,
  - $\psi(1) < \cdots < \psi(L^p)$
  - $\psi_h(1) \geq \cdots \geq \psi_h(L^p)$
- $\sum_{j=1}^{L^p} |\psi(j) - \psi_h(j)| = \pi(B) \leq \pi$
  - total error of h on requests affecting B^p
- Spread of absolute loss lemma: $L^p \leq 1 + \sqrt{5\pi}$ ∎