Advice Complexity of Online Coloring for Paths

Michal Forišek[1], Lucia Keller[2], and Monika Steinová[2]

[1]Comenius University, Bratislava, Slovakia

[2]ETH Zürich, Switzerland

LATA 2012, A Coruña, Spain

Definition

Online Problem

- Sequence of requests
- Satisfy each request before the next one arrives
- Minimize costs
- Examples: Ski rental, Paging, k-Server, various scheduling

Competitive Ratio

$$\text{comp}(\mathtt{A}(I)) = \frac{\text{Costs computed by online algorithm } \mathtt{A} \text{ on } I}{\text{Costs of an optimal solution for } I}$$

$$\text{comp}(\mathtt{A}) = \max\{\text{comp}(\mathtt{A}(I)) \mid \text{all possible } I\}$$

Definition

Online Problem

- Sequence of requests
- Satisfy each request before the next one arrives
- Minimize costs
- Examples: Ski rental, Paging, k-Server, various scheduling

Competitive Ratio

$$\text{comp}(\mathtt{A}(I)) = \frac{\text{Costs computed by online algorithm } \mathtt{A} \text{ on } I}{\text{Costs of an optimal solution for } I}$$

$$\text{comp}(\mathtt{A}) = \max\{\text{comp}(\mathtt{A}(I)) \mid \text{all possible } I\}$$

Advice Complexity

How much information are we missing. . .

- . . . to be optimal?
- . . . to achieve some competitive ratio?

A trivial example: Ski Rental

- No information about future ⇨ 2-competitive
- One bit of advice ⇨ optimal (1-competitive)

Motivation

- Theoretical interest: Measuring information loss
- Comparing with randomization
- Designing better approximation algorithms

Advice Complexity

How much information are we missing. . .

- . . . to be optimal?
- . . . to achieve some competitive ratio?

A trivial example: Ski Rental

- No information about future ⇨ 2-competitive
- One bit of advice ⇨ optimal (1-competitive)

Motivation

- Theoretical interest: Measuring information loss
- Comparing with randomization
- Designing better approximation algorithms

Advice Complexity

How much information are we missing. . .

- . . . to be optimal?
- . . . to achieve some competitive ratio?

A trivial example: Ski Rental

- No information about future ⇨ 2-competitive
- One bit of advice ⇨ optimal (1-competitive)

Motivation

- Theoretical interest: Measuring information loss
- Comparing with randomization
- Designing better approximation algorithms

Model: Details

Computation with Advice

Oracle with unlimited power:

1. Sees all requests
2. Prepares infinite tape

Algorithm starts:

3. Processes n requests one by one, can use advice tape
4. Advice: Total number of advice bits accessed

Analysis

- Solution: (oracle, algorithm)
- Correctness: the pair works correctly on all inputs
- Advice complexity $s(n)$: Maximal advice over all inputs of length $\leq n$

Model: Details

Computation with Advice

Oracle with unlimited power:

1. Sees all requests
2. Prepares infinite tape

Algorithm starts:

3. Processes n requests one by one, can use advice tape
4. Advice: Total number of advice bits accessed

Analysis

- Solution: (oracle, algorithm)
- Correctness: the pair works correctly on all inputs
- Advice complexity $s(n)$: Maximal advice over all inputs of length $\leq n$

Model: Details

Computation with Advice

Oracle with unlimited power:

1. Sees all requests
2. Prepares infinite tape

Algorithm starts:

3. Processes n requests one by one, can use advice tape
4. Advice: Total number of advice bits accessed

Analysis

- Solution: (oracle, algorithm)
- Correctness: the pair works correctly on all inputs
- Advice complexity $s(n)$: Maximal advice over all inputs of length $\leq n$

A less trivial example

Paging

Input: cache size n, sequence of page requests
Output: for each page fault: which cached page to replace?

A less trivial example

Paging

Input: cache size n, sequence of page requests
Output: for each page fault: which cached page to replace?

Offline solution

Optimal offline solution (MIN):
thrown-away page = next request is most distant

A less trivial example

Paging

Input: cache size n, sequence of page requests
Output: for each page fault: which cached page to replace?

Online approximation

Very poor!
LRU, FIFO: both have competitive ratio n.

A less trivial example

Paging

Input: cache size n, sequence of page requests

Output: for each page fault: which cached page to replace?

Advice complexity

Expected: $\Theta(\log n)$ bits per request (advice = page id)

Reality: 1 bit per request (will it be used?)

A less trivial example

Paging

Input: cache size n, sequence of page requests
Output: for each page fault: which cached page to replace?

Advice complexity

Expected: $\Theta(\log n)$ bits per request (advice = page id)
Reality: 1 bit per request (will it be used?)

Online Graph Coloring

Many practical applications; usually very hard to approximate.

Informal definition

A graph is uncovered one vertex at a time (w/incident edges).
Each time a new vertex appear, assign it a positive integer (color).
Requirement: adjacent vertices $\rightarrow$ different integers.
Goal: minimize largest integer used.

Subproblems and variations

- Subclasses of graphs (paths, trees, bipartite, planar, etc.)
- Presentation order (dfs, bfs, connected, arbitrary)
- Partial coloring (online-offline tradeoff)

Online Graph Coloring

Many practical applications; usually very hard to approximate.

Informal definition

A graph is uncovered one vertex at a time (w/incident edges).
Each time a new vertex appear, assign it a positive integer (color).
Requirement: adjacent vertices $\rightarrow$ different integers.
Goal: minimize largest integer used.

Subproblems and variations

- Subclasses of graphs (paths, trees, bipartite, planar, etc.)
- Presentation order (dfs, bfs, connected, arbitrary)
- Partial coloring (online-offline tradeoff)

Simpler results

Graph subclass: paths

Optimal: use 2 colors.
Trivial online coloring with 3 colors.
Only open question: optimality.

Result

For arbitrary presentation order:
Exactly $\lceil n/2 \rceil - 1$ bits of advice needed in worst case.

Proof sketch

- Only needs advice when given an isolated vertex.
- Hardest instances: most isolated vertices.
- $\lceil n/2 \rceil - 1$ bits of advice sufficient:
  pick any color for the first isolated vertex
  ask advice for colors of all following ones
- Main idea for even n:

  $2^{n/2}$ such instances, indistinguishable, w/different colorings
- Odd n: $+1$ bit achieved by a careful consideration of algorithm behavior for special instances.

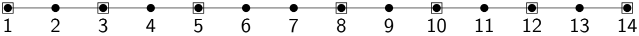
Proof sketch

- Only needs advice when given an isolated vertex.
- Hardest instances: most isolated vertices.
- $\lceil n/2 \rceil - 1$ bits of advice sufficient:
  pick any color for the first isolated vertex
  ask advice for colors of all following ones
- Main idea for even n:

  $2^{n/2}$ such instances, indistinguishable, w/different colorings
- Odd n: $+1$ bit achieved by a careful consideration of algorithm behavior for special instances.

Proof sketch

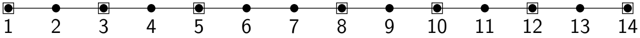
- Only needs advice when given an isolated vertex.
- Hardest instances: most isolated vertices.
- $\lceil n/2 \rceil - 1$ bits of advice sufficient:
  pick any color for the first isolated vertex
  ask advice for colors of all following ones
- Main idea for even n:



  $2^{n/2}$ such instances, indistinguishable, w/different colorings
- Odd n: $+1$ bit achieved by a careful consideration of algorithm behavior for special instances.

Proof sketch

- Only needs advice when given an isolated vertex.
- Hardest instances: most isolated vertices.
- $\lceil n/2 \rceil - 1$ bits of advice sufficient:
  pick any color for the first isolated vertex
  ask advice for colors of all following ones
- Main idea for even n:

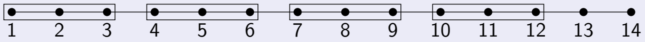
  $2^{n/2}$ such instances, indistinguishable, w/different colorings
- Odd n: $+1$ bit achieved by a careful consideration of algorithm behavior for special instances.

Paths, partial coloring, sequential presentation

Informally: Drive along a path, stop in some vertices, color them. At the end, the coloring must be a subset of an optimal coloring.

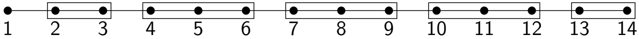
Trivial bounds

upper bound: $\lceil n/2 \rceil$-1 bits sufficient (as before)
lower bound: $\lfloor n/3 \rfloor$-1 bits necessary



A less redundant set of instances

Distance 2 or 3 between each pair of queries.
Can be visualized as dividing the path into pieces of lengths 2, 3.



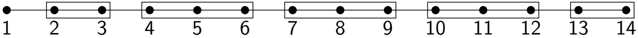
Above instance: queries 1, 3, 6, 9, 12, 14.

Cheap advice:
In $O(\log n)$ bits we can announce # of queries, # of 3-vertex steps

Goal: Maximize the number of different instances.
First guess: Same # of 2-vertex and 3-vertex steps?

A less redundant set of instances

Distance 2 or 3 between each pair of queries.
Can be visualized as dividing the path into pieces of lengths 2, 3.



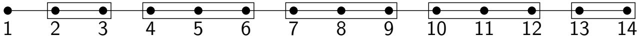
Above instance: queries 1, 3, 6, 9, 12, 14.

Cheap advice:
In $O(\log n)$ bits we can announce # of queries, # of 3-vertex steps

Goal: Maximize the number of different instances.
First guess: Same # of 2-vertex and 3-vertex steps?

A less redundant set of instances

Distance 2 or 3 between each pair of queries.
Can be visualized as dividing the path into pieces of lengths 2, 3.



Above instance: queries 1, 3, 6, 9, 12, 14.

Cheap advice:
In $O(\log n)$ bits we can announce # of queries, # of 3-vertex steps

Goal: Maximize the number of different instances.
First guess: Same # of 2-vertex and 3-vertex steps?

Nearly-optimal lower bound

Maximizing $\binom{k+l}{l}$ given that $n = 2k + 3l + 1$.
Optimum is off-center: slightly larger k is better.

A wild math formula appears! :)

$$\lg \max_{0<\alpha\leq 1/5} f(\alpha) \geq \lg \frac{1}{2x} - \lg \min_{0<\alpha\leq 1/5} (1-3\alpha) + x \cdot \lg \max_{0<\alpha\leq 1/5} \underbrace{\left(\frac{(1-\alpha)^{(1-\alpha)/2}}{(1-3\alpha)^{(1-3\alpha)/2} \cdot (2\alpha)^{\alpha}} \right)}_{g(\alpha)}$$

Nearly-optimal lower bound

Maximizing $\binom{k+l}{l}$ given that $n = 2k + 3l + 1$.
Optimum is off-center: slightly larger k is better.

A wild math formula appears! :)

$$\lg \max_{0<\alpha\le 1/5} f(\alpha) \ge \lg \frac{1}{2x} - \lg \min_{0<\alpha\le 1/5} (1-3\alpha) + x \cdot \lg \max_{0<\alpha\le 1/5} \underbrace{\left(\frac{(1-\alpha)^{(1-\alpha)/2}}{(1-3\alpha)^{(1-3\alpha)/2} \cdot (2\alpha)^{\alpha}} \right)}_{g(\alpha)}$$

Results

Lower bound

$\beta n - \lg n + O(1)$ bits of advice necessary.

Upper bound

$\beta n + 2 \lg n + \lg \lg n + O(1)$ bits of advice sufficient.

The common value β

plastic constant P: the real root of $x^3 - x - 1$; $\beta = \lg P$
closed form: $\beta = \lg \left(\sqrt[3]{9 - \sqrt{69}} + \sqrt[3]{9 + \sqrt{69}} \right) - \lg \sqrt[3]{18}$
approximate value: $\beta \approx 0.405685$.

Conclusions and future work

One newer result: online coloring for bipartite graphs (submitted to COCOON '12).

Advice complexity:
Lots of open problems, including much of graph coloring.

A more precise analysis:
tradeoff between advice and competitive ratio.

Lots of room to have fun! :)

Thank you for your attention!