Preemptively Guessing the Center

International Symposium on Combinatorial Optimization 2018

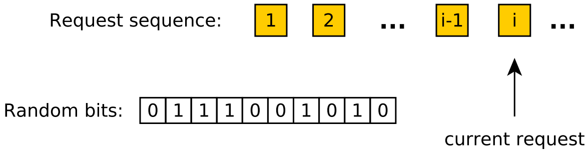
Christian Konrad and Tigran Tonoyan

University of BRISTOL

HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

13.04.2018

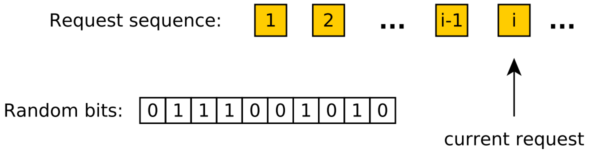
Online Algorithms



Online Algorithms:

- Each request: Irrevocable decision
- Unknown input length
- Randomized online algorithm: Access to uniform random bits

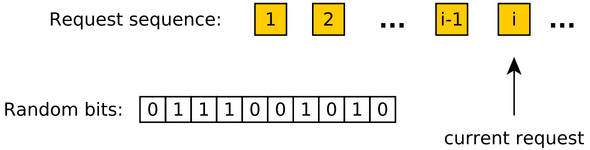
Online Algorithms



Online Algorithms:

- Each request: Irrevocable decision
- **Unknown input length**
- Randomized online algorithm: Access to uniform random bits

Online Algorithms



Online Algorithms:

- Each request: Irrevocable decision
- **Unknown input length**
- Randomized online algorithm: Access to uniform random bits

What is the impact of not knowing the input length?

Example: Rent/buy Problems

Ski Rental:

- Go skiing for unknown number of days
- Rent skis: 1 pound per day
- Buy skis: 10 pounds
- Should you rent or buy? When should you buy?
- **Trivial** if input length known in advance

Example: Rent/buy Problems

Ski Rental:

- Go skiing for unknown number of days
- Rent skis: 1 pound per day
- Buy skis: 10 pounds
- Should you rent or buy? When should you buy?
- **Trivial** if input length known in advance

General Difficulty: No orientation within the request sequence possible

- When have we seen half of the input?
- Are we close to the end of the request sequence?

Example: Rent/buy Problems

Ski Rental:

- Go skiing for unknown number of days
- Rent skis: 1 pound per day
- Buy skis: 10 pounds
- Should you rent or buy? When should you buy?
- **Trivial** if input length known in advance

General Difficulty: No orientation within the request sequence possible

- **When have we seen half of the input?**
- Are we close to the end of the request sequence?

Example: Rent/buy Problems

Ski Rental:

- Go skiing for unknown number of days
- Rent skis: 1 pound per day
- Buy skis: 10 pounds
- Should you rent or buy? When should you buy?
- **Trivial** if input length known in advance

General Difficulty: No orientation within the request sequence possible

- **When have we seen half of the input? Preemption!**
- Are we close to the end of the request sequence?

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \ldots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} |$)

Initially, $p = 0$

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

```
p ← 0 {initialization of our guess}
for each request j = 1, 2, ..., n do {n is unknown}
   if TODO: add condition here then {update guess}
      p ← j
return p
```

Example: $(p \hat{=} |)$

$| \ 1$

Either keep $p = 0$ $(| \ 1)$ or update $p = 1$ $(1 \ |)$

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

```
p ← 0 {initialization of our guess}
for each request j = 1, 2, ..., n do {n is unknown}
  if TODO: add condition here then {update guess}
    p ← j
return p
```

Example: ($p \hat{=} \,|$)

$| \; 1$

Either keep $p = 0$ ($| \; 1$) or update $p = 1$ ($1 \; |$)

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

```
p ← 0 {initialization of our guess}
for each request j = 1, 2, ..., n do {n is unknown}
   if TODO: add condition here then {update guess}
      p ← j
return p
```

Example: ($p \hat{=} |$)

$| \ 1 \ 1$

Either keep $p = 0$ ($| \ 1 \ 1$) or update $p = 2$ ($1 \ 1 \ |$)

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \ldots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} |$)

1 1 |

Either keep $p = 0$ (| 1 1) or update $p = 2$ (1 1 |)

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \ldots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} |$)

$$1\ 1 \mid 1$$

...

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \ldots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} |$)

$$1\ 1\ |\ 1\ 1$$

...

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \ldots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} \,|$)

$$1\ 1\ |\ 1\ 1\ 1$$

...

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \ldots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} |$)

1 1 1 1 1 |

...

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \ldots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} |$)

$$1\ 1\ 1\ 1\ 1\ |\ 1$$

...

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

```
p ← 0 {initialization of our guess}
for each request j = 1, 2, ..., n do {n is unknown}
   if TODO: add condition here then {update guess}
      p ← j
return p
```

Example: ($p \hat{=} |$)

$$1\ 1\ 1\ 1\ 1 \mid 1\ 1$$

...

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \dots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} |$)

1 1 1 1 1 | 1 1 1

...

Preemptively Guessing the Center

Preemptively Guessing the Center:

- **Input:** Sequence of 1s of unknown length n (even)
- **Output:** Guess p for position $n/2$ s.t. *deviation* $|p - n/2|$ minimized
- **Constraint:** Guess can only be updated at the current position

$p \leftarrow 0$ {initialization of our guess}
for each request $j = 1, 2, \ldots, n$ **do** {n is unknown}
  if *TODO: add condition here* **then** {update guess}
    $p \leftarrow j$
return p

Example: ($p \hat{=} |$)

$$1\ 1\ 1\ 1\ 1\ |\ 1\ 1\ 1$$

Sequence ends ($n = 8$). Deviation $= |5 - \frac{8}{2}| = 1$.

Weighted Version and Applications

Weighted Version:

- **Input:** Sequence X of integers of unknown length n that can be split into two parts of equal weight
- **Output:** Guess p such that $|\sum_{i=1}^{p} X_i - \frac{1}{2}\sum X|$ is minimized
- **Constraint:** Guess can only be updated at current position (as before)

3 11 2 8 37 18 4 3 19 5 6 6

Weighted Version and Applications

Weighted Version:

- **Input:** Sequence X of integers of unknown length n that can be split into two parts of equal weight
- **Output:** Guess p such that $|\sum_{i=1}^{p} X_i - \frac{1}{2}\sum X|$ is minimized
- **Constraint:** Guess can only be updated at current position (as before)

$$\underbrace{3\ 11\ 2\ 8\ 37}_{\sum=61} \mid \underbrace{18\ 4\ 3\ 19\ 5\ 6\ 6}_{\sum=61}$$

Weighted Version and Applications

Weighted Version:

- **Input:** Sequence X of integers of unknown length n that can be split into two parts of equal weight
- **Output:** Guess p such that $|\sum_{i=1}^{p} X_i - \frac{1}{2}\sum X|$ is minimized
- **Constraint:** Guess can only be updated at current position (as before)

3 11 2 8 37 | 18 | 4 3 19 5 6 6
deviation 18

Weighted Version and Applications

Weighted Version:

- **Input:** Sequence X of integers of unknown length n that can be split into two parts of equal weight
- **Output:** Guess p such that $|\sum_{i=1}^{p} X_i - \frac{1}{2}\sum X|$ is minimized
- **Constraint:** Guess can only be updated at current position (as before)

3 11 2 8 37 | 18 | 4 3 19 5 6 6
deviation 18

Applications/Relation to other Problems:

- Special case of the problem of partitioning integer sequences
- Special case of the online checkpointing problem

Our Results

Unweighted Sequences:

- **Upper Bound:** There is a randomized preemptive online algorithm with expected deviation $0.172n$.
- **Upper Bound:** Using a single random bit, an expected deviation of $0.25n$ can be achieved.
- **Lower Bound:** Every randomized preemptive online algorithm has expected deviation $0.172n$.

Our Results

Unweighted Sequences:

- **Upper Bound:** There is a randomized preemptive online algorithm with expected deviation $0.172n$.
- **Upper Bound:** Using a single random bit, an expected deviation of $0.25n$ can be achieved.
- **Lower Bound:** Every randomized preemptive online algorithm has expected deviation $0.172n$.

Weighted Sequences: $W = \sum_{i=1}^{n} X_i$

- **Upper Bound:** There is a randomized preemptive online algorithm with expected deviation $0.313W$.
- **Lower Bound:** Every (randomized) preemptive online algorithm has expected deviation $0.25W$.

Our Results

Unweighted Sequences:

- **Upper Bound:** There is a randomized preemptive online algorithm with expected deviation $0.172n$.
- **Upper Bound:** Using a single random bit, an expected deviation of $0.25n$ can be achieved.
- **Lower Bound:** Every randomized preemptive online algorithm has expected deviation $0.172n$.

Weighted Sequences: $W = \sum_{i=1}^{n} X_i$

- **Upper Bound:** There is a randomized preemptive online algorithm with expected deviation $0.313W$.
- **Lower Bound:** Every (randomized) preemptive online algorithm has expected deviation $0.25W$.

Open Question: Close gap for weighted sequences?

Upper Bounds

Upper Bound

Doubling Method with Random Seed:

1. **Chose random seed:** Choose $\delta \in (0,1)$ uniformly at random
2. **Select base:** $x = 3.052$ (unweighted) or $x = 5.357$ (weighted)
3. **Update rule: for** each request i **do**

   $p \leftarrow i$ iff $i = \lceil x^{j+\delta} \rceil$, for some $j \in \mathbb{N}$ (Unweighted)
   $p \leftarrow i$ iff $\sum_{j=1}^{i} X_j \geq \lceil x^{j+\delta} \rceil$ and $\sum_{j=1}^{i-1} X_j < \lceil x^{j+\delta} \rceil$, for $j \in \mathbb{N}$ (Weighted)

Expected Deviation: $0.172n$ (unweighted), $0.313W$ (weighted)

Remarks:

- Observe that x substantially larger than 2
- Penalty when updating is larger with weights: (update at weight 10)

$$\begin{array}{ll}
1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \mid 1 & 1\,1\,1\,1\,1\,1\,1\,1\,1\,{\color{red}2} \mid \\
1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \mid 1\,1 & 1\,1\,1\,1\,1\,1\,1\,1\,1\,{\color{red}3} \mid \\
1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \mid 1\,1\,1 & 1\,1\,1\,1\,1\,1\,1\,1\,1\,{\color{red}4} \mid
\end{array}$$

Analysis of Single Bit Algorithm

Algorithm using single random bit:

1. **Flip a coin**
2. **If coin = 'tails':** Update at positions $2^0, 2^2, 2^4, 2^6, \dots$
3. **If coin = 'heads':** Update at positions $2^1, 2^3, 2^5, 2^7, \dots$

Analysis:

- Let $n = 2^{i+\epsilon}$, for an integer i and $0 \leq \epsilon < 1$
- Algorithm either outputs 2^i or 2^{i-1}
- Thus, the expected deviation is:

$$\frac{1}{2}(2^i - 2^{i+\epsilon-1}) + \frac{1}{2}(2^{i+\epsilon-1} - 2^{i-1}) = 2^{i-2} \leq \frac{n}{4} \ .$$

Lower Bounds

Randomized versus Deterministic Algorithms

Yao's Minimax Principle:

Randomized Algorithm $\mathcal{A}_r$		**Deterministic Algorithm $\mathcal{A}_d$**
Expected deviation C on any input length	$\Rightarrow$	Expected deviation C over any input length distribution σ
$\forall n : \mathbb{E}\,\mathcal{A}_r(n) \leq C$		$\mathop{\mathbb{E}}_{n \sim \sigma} \mathcal{A}_d(n) \leq C$

Deterministic Algorithm

Uniquely specified update positions $J = \{j_1, j_2, j_3, \ldots\}$

Proof Outline:

- Define distribution σ over input lengths
- Show that for any set of update positions J, the average deviation over σ is at least C

Lower Bound Proof

Hard Input Distribution: σ

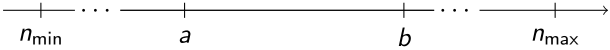
- Let $n_{\min} < n_{\max}$ be integers
- Input is of length $n \in [n_{\min}, n_{\max}]$ with probability proportional to $\frac{1}{n}$

Deterministic Algorithm: $\mathcal{A}$

Let J denote the update positions between $n_{\min}$ and $n_{\max}$

Idea:

- Consider every pair of consecutive positions $n_{\min} \leq a < b \leq n_{\max}$



- Consider input lengths $n \in [a, b]$
- Prove that expected deviation on these inputs is at least C

First Case

Standardized performance measure:

- Let R_n be the ratio between larger half and optimal split $\frac{n}{2}$
- Deviation $= R_n \cdot \frac{n}{2} - \frac{n}{2} = \frac{n}{2}(R_n - 1)$

Case: $b \leq 2a$ (assume that $b = 2a$ (worst case))

- Observe that for $a \leq n \leq 2a$, we have $R_n = 2a/n$
- Let $S = \sum_{n=a}^{2a} \frac{1}{n} \approx \ln(2a) - \ln(a) = \ln 2$
- Then, expected value R_n for input lengths n with $a \leq n \leq 2a$ is:

$$\sum_{n=a}^{b} \frac{1}{nS} \frac{2a}{n} \quad = \quad \frac{2a}{S} \sum_{n=a}^{b} \frac{1}{n^2} \approx a \frac{2}{\ln(2)} \left(\frac{1}{a} - \frac{1}{b} \right) = \frac{1}{\ln(2)} \; .$$

- Deviation:

$$\frac{n}{2}(R_n - 1) = \frac{n}{2} \left(\frac{1}{\ln(2)} - 1 \right) \approx 0.2213n > 0.172n \; .$$

Completing the Proof

Case: $b > 2a$

- Requires more work
- We prove: deviation $\geq 0.172n$ in this case
- Worst case: $b/a \approx 3.052$
- Observe: This is the same as the base in our upper bound

Completing the Proof

Case: $b > 2a$

- Requires more work
- We prove: deviation $\geq 0.172n$ in this case
- Worst case: $b/a \approx 3.052$
- Observe: This is the same as the base in our upper bound

Theorem Every randomized preemptive online algorithm for Guessing the Center on unweighted sequences has expected deviation $0.172n$.

Summary

Our Results:

	Unweighted sequences	Weighted sequences
Upper Bound:	$0.172n$	$0.313n$
Lower Bound:	$0.172n$	$0.25n$

Open Questions:

- Close gap for weighted sequences?
- Guessing $1/3$, $1/4$ of the input length?
- Randomized algorithms for online checkpointing (i.e., splitting in more than 2 parts)

Summary

Our Results:

	Unweighted sequences	Weighted sequences
Upper Bound:	$0.172n$	$0.313n$
Lower Bound:	$0.172n$	$0.25n$

Open Questions:

- Close gap for weighted sequences?
- Guessing $1/3$, $1/4$ of the input length?
- Randomized algorithms for online checkpointing (i.e., splitting in more than 2 parts)

Thank you