Election vs. Selection: How Much Advice is Needed to Find the Largest Node in a Graph?

Avery Miller
University of Manitoba
avery@averymiller.ca

Andrzej Pelc
Université du Québec en Outaouais
andrzej.pelc@uqo.ca

First presented at SPAA 2016
Presentation - Damien Aymon, 08.05.2018
ETHZ - Seminar in Distributed Computing FS 2018
Application – Shared Resource

Microsoft Docs, Leader Election Pattern, 23.06.2017
https://docs.microsoft.com/en-us/azure/architecture/patterns/leader-election
Application – Shared Resource

Access to shared resource -> need coordinator

Failure resilience -> need new leader
Application – Radiocom

G. Le Lann, Distributed Systems - Towards a Formal Approach
Application – Radiocom
Application – Radiocom
Application – Radiocom
Application – Radiocom
Application – Radiocom
Application – Radiocom
Election vs. Selection: How Much Advice is Needed to Find the Largest Node in a Graph?

Avery Miller
University of Manitoba
avery@averymiller.ca

Andrzej Pelc*
Université du Québec en Outaouais
andrzej.pelc@uqo.ca
Election

Find a leader

Everyone knows its identity
Election

Find a leader

Everyone knows its identity
Election

Find a leader

Everyone knows its identity
Selection

Leader outputs 1

Every other node outputs 0
Selection

Leader outputs 1

Every other node outputs 0
Selection

Leader outputs 1

Every other node outputs 0
Election vs Selection
Synchronous Algorithms

In each round:

• **Send** messages to neighbours
• **Receive** messages from neighbours
• Do some **computation**

Time complexity is the **number of rounds**
Synchronous Algorithms

In each round:

- **Send** messages to neighbours
- **Receive** messages from neighbours
- Do some **computation**

Time complexity is the **number of rounds**
Synchronous Algorithms

In each **round**:

- **Send** messages to neighbours
- **Receive** messages from neighbours
- Do some **computation**

Time complexity is the **number of rounds**
Synchronous Algorithms

In each **round**:

- **Send** messages to neighbours
- **Receive** messages from neighbours
- Do some **computation**

Time complexity is the **number of rounds**
Algorithm with advice

Oracle with **full knowledge**

Gives same **advice** to each node

Goal: make computation **faster**
Algorithm with advice

Oracle with **full knowledge**

Gives same **advice** to each node

Goal: make computation **faster**
Algorithm with advice

Oracle with full knowledge

Gives same advice to each node

Goal: make computation faster
Algorithm with advice

Example: Election without advice

Notations:

\[ K(r, v) = \text{knowledge of } v \text{ after } r \text{ rounds} \]

\[ \Lambda(r, v) \text{ set of labels induced by } K(r, v) \]
Algorithm with advice

Example: Election **without** advice

Notations:

\[ K(r, \nu) = \text{knowledge of } \nu \text{ after } r \text{ rounds} \]

\[ \Lambda(r, \nu) \text{ set of labels induced by } K(r, \nu) \]

\[ K(0, 4) \]

\[ \Lambda(0, 4) = \{4\} \]
Algorithm with advice

Example: Election **without** advice

Notations:

\[ K(r, \nu) = \text{knowledge of } \nu \text{ after } r \text{ rounds} \]

\[ \Lambda(r, \nu) \text{ set of labels induced by } K(r, \nu) \]

\[ K(1, 4) \]

\[ \Lambda(1, 4) = \{4, 6\} \]
Algorithm with advice

Example: Election *without* advice

Notations:

\[ K(r, v) = \text{knowledge of } v \text{ after } r \text{ rounds} \]

\[ \Lambda(r, v) = \text{set of labels induced by } K(r, v) \]

\[ K(2, 4) \]

\[ \Lambda(2, 4) = \{3, 4, 6\} \]

Round 2
Algorithm with advice

Example: Election without advice

Notations:

\[ K(r, v) = \text{knowledge of } v \text{ after } r \text{ rounds} \]

\[ \Lambda(r, v) = \text{set of labels induced by } K(r, v) \]

\[ K(3, 4) \]

\[ \Lambda(3, 4) = \{3, 4, 5, 6, 12\} \]
Algorithm with advice

Example: Election without advice

Notations:

\[ K(r, \nu) = \text{knowledge of } \nu \text{ after } r \text{ rounds} \]

\[ \Lambda(r, \nu) \text{ set of labels induced by } K(r, \nu) \]

\[ K(4, 4) \]

\[ \Lambda(4, 4) = \{1, 3, 4, 5, 6, 10, 12\} \]
Algorithm with advice

Example: Election \textbf{without} advice

\[ K(4, 4) \]

\[ \Lambda(4, 4) = \{1, 3, 4, 5, 6, 10, 12\} \]

Requires another round to terminate!

Round 4
Algorithm with advice

Example: Election **without** advice

\[ K(4, 4) \]

\[ \Lambda(4, 4) = \{1, 3, 4, 5, 6, 10, 12\} \]

Requires another round to terminate!

What if we give some **advice**?
Task – Measure of Difficulty

Time constraint for the execution

How much advice needed?

Upper and lower bound the size of advice
Tight Bounds on Advice

Tight bounds are given on the size of advice.
Tight Bounds on Advice

Tight bounds are given on the size of advice.

$$\Theta(f(x)) \iff \Omega(f(x)) \land O(f(x))$$
Tight Bounds on Advice

Tight bounds are given on the size of advice.

\[ \Theta(f(x)) \iff \Omega(f(x)) \land O(f(x)) \]

Lower bound \( l \)
Find a class of graphs for which at least \( l \) advice needed for any algorithm.

Upper bound \( u \)
Find an algorithm for which at most \( u \) advice needed on all graphs.
How is it helpful?

Can *rule out* entire classes of algorithms
How is it helpful?

Can **rule out** entire classes of algorithms

Given result: Task $T$ needs $\Theta(\log n)$ bits of advice
How is it helpful?

Can rule out entire classes of algorithms

Given result: Task $T$ needs $\Theta(\log n)$ bits of advice

Proposed algorithm: Needs linear upper bound on $n$ as advice.
How is it helpful?

Can **rule out** entire classes of algorithms

Given result: Task $T$ needs $\Theta(\log n)$ bits of advice

Proposed algorithm: Needs linear upper bound on $n$ as advice.

**Contradiction:** Advice can be given by $\lceil \log n \rceil$, using $\Theta(\log \log n)$ bits.
## Results - Election

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; diam</td>
<td>0</td>
</tr>
<tr>
<td>diam</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
<tr>
<td>&lt; diam</td>
<td>$\Theta(\log n)$</td>
</tr>
</tbody>
</table>
# Results - Election

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \text{diam}$</td>
<td>0</td>
</tr>
<tr>
<td>$\text{diam}$</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
<tr>
<td>$&lt; \text{diam}$</td>
<td>$\Theta(\log n)$</td>
</tr>
</tbody>
</table>
## Results - Election

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; diam</td>
<td>0</td>
</tr>
<tr>
<td>diam</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
<tr>
<td>&lt; diam</td>
<td>$\Theta(\log n)$</td>
</tr>
</tbody>
</table>

Provide the **diameter** of the graph.
## Results - Election

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \text{diam}$</td>
<td>0</td>
</tr>
<tr>
<td>$\text{diam}$</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
<tr>
<td>$&lt; \text{diam}$</td>
<td>$\Theta(\log n)$</td>
</tr>
</tbody>
</table>

No better advice than to give the solution.
Results - Election

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; diam</td>
<td>0</td>
</tr>
<tr>
<td>diam</td>
<td>(\Theta(\log diam))</td>
</tr>
<tr>
<td>&lt; diam</td>
<td>(\Theta(\log n))</td>
</tr>
</tbody>
</table>

Intra-task jumps
## Results - Selection

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; diam</td>
<td>0</td>
</tr>
<tr>
<td>$a \cdot diam$, $a \in (0, 1)$</td>
<td>$\Theta(\log \log diam)$</td>
</tr>
<tr>
<td>$diam^e$, $e &lt; 1$</td>
<td>$\Theta(\log diam)$</td>
</tr>
</tbody>
</table>
Results - Selection

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$diam$</td>
<td>0</td>
</tr>
<tr>
<td>$a \cdot diam$, $a \in (0, 1)$</td>
<td>$\Theta(\log \log diam)$</td>
</tr>
<tr>
<td>$diam^e$, $e &lt; 1$</td>
<td>$\Theta(\log diam)$</td>
</tr>
</tbody>
</table>

Only valid for rings
## Results - Selection

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \ diam$</td>
<td>0</td>
</tr>
<tr>
<td>$a \cdot diam, \ a \in (0,1)$</td>
<td>$\Theta(\log \log diam)$</td>
</tr>
<tr>
<td>$diam^e, \ e &lt; 1$</td>
<td>$\Theta(\log diam)$</td>
</tr>
</tbody>
</table>
## Results - Selection

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \text{diam}$</td>
<td>0</td>
</tr>
<tr>
<td>$a \cdot \text{diam}, a \in (0, 1)$</td>
<td>$\Theta(\log \log \text{diam})$</td>
</tr>
<tr>
<td>$\text{diam}^e, e &lt; 1$</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
</tbody>
</table>

We will go through the algorithm for the upper bound.
## Results - Selection

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \text{diam}$</td>
<td>0</td>
</tr>
<tr>
<td>$a \cdot \text{diam}$, $a \in (0, 1)$</td>
<td>$\Theta(\log \log \text{diam})$</td>
</tr>
<tr>
<td>$\text{diam}^e$, $e &lt; 1$</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
</tbody>
</table>

For rings $\Theta(\log \text{diam}) = \Theta(\log n)$
## Results - Selection

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>( &gt; \text{diam} )</td>
<td>0</td>
</tr>
<tr>
<td>( a \cdot \text{diam}, \ a \in (0, 1) )</td>
<td>( \Theta(\log \log \text{diam}) )</td>
</tr>
<tr>
<td>( \text{diam}^e, \ e &lt; 1 )</td>
<td>( \Theta(\log \text{diam}) )</td>
</tr>
</tbody>
</table>

Intra-task jumps
## Results

<table>
<thead>
<tr>
<th>Time Condition</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \text{diam}$</td>
<td>0</td>
</tr>
<tr>
<td>$\text{diam}$</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
<tr>
<td>$&lt; \text{diam}$</td>
<td>$\Theta(\log n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time Condition</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \text{diam}$</td>
<td>0</td>
</tr>
<tr>
<td>$a \cdot \text{diam}, \ a \in (0, 1)$</td>
<td>$\Theta(\log \log \text{diam})$</td>
</tr>
<tr>
<td>$\text{diam}^e, \ e &lt; 1$</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \text{diam}$</td>
<td>0</td>
</tr>
<tr>
<td>$\text{diam}$</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
<tr>
<td>$&lt; \text{diam}$</td>
<td>$\Theta(\log n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Advice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; \text{diam}$</td>
<td>0</td>
</tr>
<tr>
<td>$a \cdot \text{diam}$, $a \in (0, 1)$</td>
<td>$\Theta(\log \log \text{diam})$</td>
</tr>
<tr>
<td>$\text{diam}^e$, $e &lt; 1$</td>
<td>$\Theta(\log \text{diam})$</td>
</tr>
</tbody>
</table>

Inter-task jump
Selection - Example

Time constraint: \( t = a \cdot diam, a \in (0, 1) \)

Goal: Prove that size of advice is \( O(\log \log \text{diam}(R)) \) for any ring \( R \)

Simplification: \( a = 1 \), so \( t = \text{diam} \)
Selection - Example

Algorithm consists of two stages

1. Round-by-round discovery
2. Eliminate resulting nodes from first stage

Advice string split in two parts $A = A_1A_2$
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lfloor \log(diam(R)) \rfloor \]

On each node \( v \)
Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lfloor \log(\text{diam}(R)) \rfloor \]

On each node \( v \)
Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lceil \log(\text{diam}(R)) \rceil \]

On each node \( v \)
Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lfloor \log(diam(R)) \rfloor \]

[\log(5)] = 2

On each node \( v \)

Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \) \( r = 4 \)

Round 0
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lfloor \log(\text{diam}(R)) \rfloor \]

\[ \lfloor \log(5) \rfloor = 2 \]

On each node \( v \)

Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)

\( r = 4 \)

Round 1
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lceil \log(\text{diam}(R)) \rceil \quad \text{[log}(5)] = 2 \]

On each node \( v \)

Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)

\( r = 4 \)
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lfloor \log(diam(R)) \rfloor \]

\[ \lfloor \log(5) \rfloor = 2 \]

On each node \( v \)

Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)

\( r = 4 \)

![Diagram showing a network with nodes 14, 9, 4, 10, 2, 12, 7, 8, 5. The network is divided into two components, with one component having nodes 14, 9, 4, 10, and the other having nodes 2, 12, 7, 8, 5. The diagram indicates Round 3.]
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lfloor \log(\text{diam}(R)) \rfloor \]

\[ \lfloor \log(5) \rfloor = 2 \]

On each node \( v \)

Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)

\[ r = 4 \]

Round 4
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lceil \log(\text{diam}(R)) \rceil \]

[\log(5)] = 2

On each node \( v \)

Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)

If \( v \neq \max(\Lambda(r, v)) \) output 0

Round 4
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lfloor \log(\text{diam}(R)) \rfloor \]

\[ [\log(5)] = 2 \]

On each node \( v \)

- Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)
- If \( v \neq \max(\Lambda(r, v)) \) output 0

\( r = 4 \)
Selection – Example

Algorithm - Stage 1

\[ A_1 = \lfloor \log(\text{diam}(R)) \rfloor \]

\[ \lfloor \log(5) \rfloor = 2 \]

On each node \( v \)

Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)

If \( v \neq \max(\Lambda(r, v)) \) output 0

Round 4
Selection – Example

Algorithm - Stage 2: Advice construction

\[ C_R = \{ \gamma_0, \gamma_1, \ldots, \gamma_{|C_R|-1} \} = \text{set of resulting nodes where } \gamma_0 \text{ is largest} \]

Goal: eliminate all but \( \gamma_0 \)
Selection – Example

Algorithm - Stage 2: Advice construction

\[ C_R = \{\gamma_0, \gamma_1, \ldots, \gamma_{|C_R|-1}\} = \text{set of resulting nodes where } \gamma_0 \text{ is largest} \]

Goal: eliminate all but \( \gamma_0 \)

Solution: for each \( \gamma_j, j > 0 \), find difference with \( \gamma_0 \) and provide it as advice
Selection – Example

Algorithm - Stage 2: Advice construction

\[ C_R = \{\gamma_0, \gamma_1, \ldots, \gamma_{|C_R|-1}\} = \text{resulting set of nodes where } \gamma_0 \text{ is largest} \]

Goal: eliminate all but \( \gamma_0 \)

Solution: for each \( \gamma_j, j > 0 \), find difference with \( \gamma_0 \) and provide it as advice

\[ \gamma_0 = 100111 \]

\[ \gamma_1 = 100011 \]
Selection – Example

Algorithm - Stage 2: Advice construction

\[ C_R = \{\gamma_0, \gamma_1, \ldots, \gamma_{|C_R|-1}\} = \text{resulting set of nodes where } \gamma_0 \text{ is largest} \]

Goal: eliminate all but \( \gamma_0 \)

Solution: for each \( \gamma_j, j > 0 \), find difference with \( \gamma_0 \) and provide it as advice

\[ \gamma_0 = 100111 \]

\[ \gamma_2 = 100101 \]
Selection – Example

Algorithm - Stage 2: Advice construction

$A_2$ set of indices satisfying:

For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \emptyset$
Selection – Example

Algorithm - Stage 2: Advice construction

$A_2$ set of indices satisfying:

For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \emptyset$

$\gamma_0 = 100111$

$\gamma_1 = 100011$
Selection – Example

Algorithm - Stage 2: Advice construction

$A_2$ set of indices satisfying:

For all $\gamma_{j, j>0}$, there exists $i \in A_2$ such that $\gamma_{j[i]} = 0$ and $\gamma_{0[i]} = 1$

Construction Example

$A_2 = \{3\}$

$\gamma_0 = 100111$

$\gamma_1 = 100011$
Selection – Example

Algorithm - Stage 2: Advice construction

$A_2$ set of indices satisfying:

For all $\gamma_j, j > 0$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \{3\}$

$\gamma_0 = 100111$

$\gamma_2 = 100101$
Selection – Example

Algorithm - Stage 2: Advice construction

$A_2$ set of indices satisfying:

For all $\gamma_j, j > 0$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \{3, 4\}$

$\gamma_0 = 100111$

$\gamma_2 = 100101$
Selection – Example

Algorithm - Stage 2: Advice construction

\( A_2 \) set of indices satisfying:

\[
\text{For all } \gamma_j, j > 0, \text{ there exists } i \in A_2 \text{ such that } \gamma_j[i] = 0 \text{ and } \gamma_0[i] = 1
\]

Construction Example
\( A_2 = \{3, 4\} \)

\[
\gamma_0 = 100111
\]

\[
\gamma_3 = 100001
\]
Selection – Example

Algorithm - Stage 2: Algorithm

On each node $\gamma$ in $C_R$
  If there exists $i \in A_2$ such that $\gamma[i] = 0$
    Output 0
  Else
    Output 1
Selection – Example

Algorithm - Stage 2: Algorithm

On each node $\gamma$ in $C_R$
   If there exists $i \in A_2$ such that $\gamma[i] = 0$
      Output 0
   Else
      Output 1

Algorithm at $\gamma_2$

$A_2 = \{3, 4\}$

$\gamma_2 = 100101$
Selection – Example

Algorithm - Stage 2: Algorithm

On each node $\gamma$ in $C_R$
   If there exists $i \in A_2$ such that $\gamma[i] = 0$
      Output 0
   Else
      Output 1

Algorithm at $\gamma_2$

$A_2 = \{3, 4\}$

$\gamma_2 = 100101$

$\Rightarrow$ Output 0
Selection – Example

Algorithm - Recap

\[ A = A_1 A_2 \]

\[ A_1 = \lfloor \log(\text{diam}(R)) \rfloor \]

\[ A_2 : \text{For all } \gamma_{j,j>0}, \text{ there exists } i \in A_2 \text{ such that } \gamma_j[i] = 0 \text{ and } \gamma_0[i] = 1 \]

On each node \( v \)

- Run \( r = 2^{A_1} \) rounds to learn \( \Lambda(r, v) \)
- If \( v \neq \max(\Lambda(r, v)) \) output 0

On each node \( \gamma \) in \( C_R \)

- If there exists \( i \in A_2 \) such that \( \gamma[i] = 0 \)
  - Output 0
- Else
  - Output 1
Selection – Example

Algorithm - Size of Advice

\[ A_1 = \lfloor \log \text{diam}(R) \rfloor \]

Size of \( A_1 \) is \( O(\log \log \text{diam}(R)) \)

\[ A_2 = \text{is a set of at most } |C_R| \text{ indices} \]

Size of each index is \( O(\log \log \text{diam}(R)) \)

\[ |C_R| \text{ is a constant} \]

Size of \( A_2 \) is \( O(|C_R| \cdot \log \log \text{diam}(R)) = O(\log \log \text{diam}(R)) \)

Size of advice \( A \) is \( O(\log \log \text{diam}(R)) \)

Note that the time constraint is also respected.
Summary

• Election vs Selection

• Algorithm with advice

• Measure of difficulty – size of advice

• Results

• Algorithm overview for selection in time linear in the diameter (upper bound)
Questions ?
Related Work

• Message complexity

• Non-unique labels

• Non-labelled graphs

• Election of arbitrary node

• Algorithms with advice for other problems

• Different advice for each node

• Different kind of difficulty measurement.