Coupled Bisection for Root-Ordering

Stephen N. Pallone
Cornell University ORIE
snp32@cornell.edu

Peter I. Frazier
Cornell University ORIE
pf98@cornell.edu

Shane G. Henderson
Cornell University ORIE
sgh9@cornell.edu

July 17, 2015
One-Dimensional Root-Finding
One-Dimensional Root-Finding

Problem Statement

Structural Results

Bisection Policy
One-Dimensional Root-Finding

Problem Statement

Structural Results

Bisection Policy

Coupled Bisection for Root-Ordering

Stephen N. Pallone
Coupled Root-Ordering
Coupled Root-Ordering
Coupled Root-Ordering

Problem Statement

Structural Results

Bisection Policy
• Let \( \mathcal{S} \) denote a set of elements (large but finite).

• Let \( f : \mathcal{S} \times [0,1) \rightarrow \mathbb{R} \), monotonic in its second argument.

• For every element \( s \in \mathcal{S} \), denote \( x^*(s) \) as the unique root of \( f(s, \cdot) \).

• Suppose when we evaluate \( f \) for a given \( \bar{x} \in [0,1) \), we find \( f(s, \bar{x}) \) for every \( s \in \mathcal{S} \).

• Goal: find the complete ordering of elements \( s \) by their roots \( x^*(s) \) with the least number of function evaluations as possible.
Applications

Multi-armed Bandit Problems

- Sorting states in state space according to their Gittens or Whittle indices.
- Ordering of states is all that is required to implement optimal policy.

Remote Sensing

- E.g., finding the boundary of a forest from an airborne laser scanner.
The Naive Approach
The Naive Approach

Problems with Discretizing the Interval

• Non-adaptive in nature; unclear how to determine subinterval length.
• May require more computational work than necessary.
The Naive Approach

Problems with Discretizing the Interval

- Non-adaptive in nature; unclear how to determine subinterval length.
- May require more computational work than necessary.
The Clever Approach
The Clever Approach
The Clever Approach

How can we adaptively choose where to query $f$?
The Clever Approach

Problem Statement

Structural Results

Bisection Policy
How can we adaptively choose where to query $f$?
Formal Definition

State Variables

$\begin{align*}
\text{At time epoch } j: \\
&\text{Let } X_j = \{(x(0), x(1)), \ldots, (x(j), x(j+1))\} \text{ denote the current partition of the interval.} \\
&\text{Let } N_j = (n(0), n(1), \ldots, n(j)) \text{ denote the number of roots in each subinterval.} \\
&\text{The pair } (X_j, N_j) \text{ denotes the computational state.}
\end{align*}$
Formal Definition

State Variables

\[ n^{(0)} \]
\[ 2 \]
\[ x^{(0)} \]

\[ n^{(1)} \]
\[ 2 \]
\[ x^{(1)} \]
Formal Definition

State Variables

\[ n^{(0)} \]

\[ n^{(1)} \]

\[ n^{(2)} \]

At time epoch \( j \):

- Let \( X_j = \{ [x^{(0)}, x^{(1)}], \ldots, [x^{(j)}, x^{(j+1)}] \} \) denote the current partition of the interval.
- Let \( N_j = (n^{(0)}, n^{(1)}, \ldots, n^{(j)}) \) denote the number of roots in each subinterval.
- The pair \( (X_j, N_j) \) denotes the computational state.
Formal Definition

State Variables

\[ n^{(0)} \]

\[ n^{(1)} \]

\[ n^{(2)} \]

\[ n^{(0)} \]

\[ n^{(1)} \]

\[ n^{(2)} \]

\[ x^{(0)} \]

\[ x^{(1)} \]

\[ x^{(2)} \]

\[ x^{(3)} \]

At time epoch \( j \):

- Let \( X_j = \{ [x^{(0)}, x^{(1)}], \ldots, [x^{(j)}, x^{(j+1)}] \} \) denote the current partition of the interval.
- Let \( N_j = (n^{(0)}, n^{(1)}, \ldots, n^{(j)}) \) denote the number of roots in each subinterval.
- The pair \( (X_j, N_j) \) denotes the computational state.
Formal Definition

Transition

- We assume i.i.d. priors on $x^*(s)$ for every element $s \in S$.
- Without loss of generality, $x^*(s) \sim \text{Uniform}[0, 1)$. 

\[ n_{j+1} = (n_0, \ldots, n_{\ell-1}, \bar{N}_{\ell}, n_{\ell}, \ldots, n_j) \]

where $\bar{N}_{\ell} \sim \text{Binomial}(n_\ell, \bar{x} - x_{\ell})$. 

\[ x_{\ell+1} - x_{\ell} \]
Formal Definition

Transition

• We assume i.i.d. priors on \( x^*(s) \) for every element \( s \in S \).
• Without loss of generality, \( x^*(s) \sim \text{Uniform}[0, 1] \).
Formal Definition

Transition

- We assume i.i.d. priors on $x^*(s)$ for every element $s \in S$.
- Without loss of generality, $x^*(s) \sim \text{Uniform}[0, 1)$.
Formal Definition

Transition

- We assume i.i.d. priors on $x^*(s)$ for every element $s \in S$.
- Without loss of generality, $x^*(s) \sim \text{Uniform}[0, 1]$.

```
2
```

```
2
```

```
× × ×
```

```
× × ×
```
Formal Definition

Transition

- We assume i.i.d. priors on $x^*(s)$ for every element $s \in S$.
- Without loss of generality, $x^*(s) \sim \text{Uniform}[0, 1]$.

\[
\begin{align*}
N_{j+1} &= \left( n^{(0)}, \ldots, n^{(\ell-1)}, \bar{N}^{(\ell)}, n^{(\ell)} - \bar{N}^{(\ell)}, n^{(\ell+1)}, \ldots, n^{(j)} \right),
\end{align*}
\]

where

\[
\bar{N}^{(\ell)} \sim \text{Binomial} \left( n^{(\ell)}, \frac{\bar{x} - x^{(\ell)}}{x^{(\ell+1)} - x^{(\ell)}} \right).
\]
Formal Definition

Objective Function

Let $\tau$ denote the number of function evaluations to sort all elements, i.e.,

$$\tau = \inf \{ j \in \mathbb{N} : n(i, j) \leq 1 \forall i \}.$$
Formal Definition

Objective Function

Let $\tau$ denote the number of function evaluations to sort all elements, i.e.,
$$\tau = \inf \{ j \in \mathbb{N} : n(i)^j \leq 1 \ \forall \ i \}.$$
Formal Definition

Objective Function

Let $\tau$ denote the number of function evaluations to sort all elements, i.e., $\tau = \inf \{ j \in \mathbb{N} : n(i)^j \leq 1 \quad \forall i \}$.

Problem Statement

Structural Results

Bisection Policy
Formal Definition

Objective Function

Let $\tau$ denote the number of function evaluations to sort all elements, i.e.,

$$\tau = \inf\{j \in \mathbb{N} : n_j^{(i)} \leq 1 \quad \forall i\}.$$
Formal Definition

Defining the Optimal Policy
Formal Definition

Defining the Optimal Policy

- A policy $\pi$ is a mapping from computational states $(X, N)$ to a refined partition $\bar{X}$ that adds one additional subinterval.
Formal Definition

Defining the Optimal Policy

- A policy $\pi$ is a mapping from computational states $(X, N)$ to a refined partition $\tilde{X}$ that adds one additional subinterval.
- Let $W^\pi(X, N)$ denote the expected number of function evaluations required to sort elements, where we start at state $(X, N)$, i.e.,

$$W^\pi(X, N) = \mathbb{E}^\pi [\tau | (X_0, N_0) = (X, N)].$$
Formal Definition

Defining the Optimal Policy

- A policy $\pi$ is a mapping from computational states $(X, N)$ to a refined partition $\bar{X}$ that adds one additional subinterval.
- Let $W^\pi(X, N)$ denote the expected number of function evaluations required to sort elements, where we start at state $(X, N)$, i.e.,

\[ W^\pi(X, N) = \mathbb{E}^\pi [\tau | (X_0, N_0) = (X, N)] . \]

- Let $W(X, N) = \inf_\pi W^\pi(X, N)$ denote the expected number of evaluations required under the optimal policy.
By the principle of optimality, we can iteratively take the best partition $\bar{X}$, giving us a recursion.
By the principle of optimality, we can iteratively take the best partition $\bar{X}$, giving us a recursion.

**Theorem (Basic Recursion)**

*For a given computational state $(X, N)$, we have*

$$W(X, N) = 1 + \inf_{\bar{X}} \mathbb{E} \left[ W(\bar{X}, \bar{N}) \mid (X_0, N_0) = (X, N) \right],$$

*with stopping conditions $W(X, N) = 0$ if $n^{(i)} \leq 1$ for all $i$.***
Theorem (Decomposition and Interval Invariance)

Under the optimal policy, the value function $W$ has the following two properties.

- **Decomposition:**
  $$W(X, N) = \sum_{i=0}^{\lfloor N \rfloor - 1} W \left( \{ [x^{(i)}, x^{(i+1)}) \} , n^{(i)} \right)$$

- **Interval Invariance:**
  $$W \left( \{ [x^{(i)}, x^{(i+1)}) \} , n^{(i)} \right) = W \left( \{ [0, 1) \} , n^{(i)} \right).$$
Decomposition

**Ex.** Expected remaining number of iterations required to finish sorting 7 elements:

\[
W \left( \left\{ [x^{(0)}, x^{(1)}), [x^{(1)}, x^{(2)}), [x^{(2)}, x^{(3)}] \right\}, (2, 2, 3) \right)
\]
**Decomposition**

**Ex.** Expected remaining number of iterations required to finish sorting 7 elements:

\[ W \left( \left\{ [x^{(0)}, x^{(1)}], [x^{(1)}, x^{(2)}], [x^{(2)}, x^{(3)}] \right\}, (2, 2, 3) \right) \]

\[ \begin{align*} &2 \quad 2 \quad 3 \\ &\text{sort 2 elements} \quad \text{sort 2 elements} \quad \text{sort 3 elements} \end{align*} \]
**Decomposition**

**Ex.** Expected remaining number of iterations required to finish sorting 7 elements:

\[
W \left( \left\{ [x^{(0)}, x^{(1)}], [x^{(1)}, x^{(2)}], [x^{(2)}, x^{(3)}] \right\}, (2, 2, 3) \right)
\]

\[
\begin{align*}
\text{sort 2 elements} & \quad \text{sort 2 elements} & \quad \text{sort 3 elements} \\
\begin{array}{c}
\boxed{2} \\
\boxed{2} \\
\boxed{3}
\end{array} & \quad \begin{array}{c}
\times \quad \times \\
\times \quad \times \\
\times \quad \times \\
\end{array} & \quad \begin{array}{c}
\times \quad \times \\
\times \quad \times \\
\times \quad \times \\
\end{array}
\end{align*}
\]

\[
= W \left( \{ [x^{(0)}, x^{(1)}] \}, 2 \right) + W \left( \{ [x^{(1)}, x^{(2)}] \}, 2 \right) + W \left( \{ [x^{(2)}, x^{(3)}] \}, 3 \right)
\]
Interval Invariance
Interval Invariance

Problem Statement

Structural Results

Bisection Policy
Interval Invariance

Problem Statement

Structural Results

Bisection Policy
Interval Invariance

Problem Statement

Structural Results

Bisection Policy
Interval Invariance

![Diagram of intervals and numbers]

Re-scaling

Problem Statement

Structural Results

Bisection Policy
Interval Invariance

Problem Statement

Structural Results

Bisection Policy
Corollary (Simple Recursion)

$$W(n) = 1 + \min_{x \in (0, 1)} E\left[ W(N_x) + W(n - N_x) \right],$$

where $N_x \sim \text{Binomial}(n, x)$, and $W(0) = W(1) = 0$. 

Problem Statement

Structural Results

Bisection Policy
A Simpler Recursion

\[ n \]

\[ 0 \quad x \quad x \quad 1 \]

\[ N_x \sim \text{Binomial}(n, x) \]

\[ n - N_x \]

\[ 0 \quad x \quad x \quad 1 \]

Corollary (Simple Recursion)

\[ W(n) = 1 + \min_{x \in (0, 1)} \mathbb{E}[W(N_x) + W(n - N_x)], \]

where \( N_x \sim \text{Binomial}(n, x) \), and \( W(0) = W(1) = 0 \).
A Simpler Recursion

\[ W ([0, 1], n) \]

\[ N_x \sim \text{Binomial}(n, x) \]

\[ W ([0, x], N_x) + W ([x, 1], n - N_x) \]
A Simpler Recursion

\[ \begin{align*}
N_x & \sim \text{Binomial}(n, x) \\
0 & \quad W(\{[0, 1]\}, n) \\
& \quad \downarrow \\
N_x & \quad n - N_x \\
0 & \quad 1 \quad 0 \quad 1
\end{align*} \]

\[ W(\{[0, 1]\}, N_x) + W(\{[0, 1]\}, n - N_x) \]
A Simpler Recursion

\[ W(n) = 1 + \min_{x \in (0, 1)} \mathbb{E}[W(N_x) + W(n - N_x)] , \]
where \( N_x \sim \text{Binomial}(n, x) \), and \( W(0) = W(1) = 0 \).
For notation purposes, let

\[ W(n) := W([0, 1], n). \]
A Simpler Recursion

For notation purposes, let

\[ W(n) := W([0, 1], n). \]

**Corollary (Simple Recursion)**

\[ W(n) = 1 + \min_{x \in (0, 1)} \mathbb{E}[W(N_x) + W(n - N_x)], \]

where \( N_x \sim \text{Binomial}(n, x) \), and \( W(0) = W(1) = 0 \).
Computing the Optimal Policy

\[ W(n) = 1 + \min_{x \in (0,1)} \mathbb{E}[W(N_x) + W(n - N_x)] \]
Computing the Optimal Policy

\[ W(n) = 1 + \min_{x \in (0,1)} \mathbb{E}[W(N_x) + W(n - N_x)] \]

- Using the stopping conditions \( W(0) = W(1) = 0 \), we can iteratively compute \( W(\cdot) \) for arbitrary \( n \).
Computing the Optimal Policy

\[ W(n) = 1 + \min_{x \in (0,1)} \mathbb{E}[W(N_x) + W(n - N_x)] \]

- Using the stopping conditions \( W(0) = W(1) = 0 \), we can iteratively compute \( W(\cdot) \) for arbitrary \( n \).
- Problem: \( N_x \in \{0, n\} \) with positive probability.
Computing the Optimal Policy

\[ W(n) = 1 + \min_{x \in (0,1)} \mathbb{E} \left[ W(N_x) + W(n - N_x) \right] \]

- Using the stopping conditions \( W(0) = W(1) = 0 \), we can iteratively compute \( W(\cdot) \) for arbitrary \( n \).
- Problem: \( N_x \in \{0, n\} \) with positive probability.
- This is an implicit definition of \( W \).
Computing the Optimal Policy

Theorem (Computational Recursion)

For all $n \geq 2$, we have

$$W(n) = \min_{x \in (0,1)} \left\{ \frac{1}{1 - x^n - (1 - x)^n} \right. \\
+ \mathbb{E} \left[ W(N_x) + W(n - N_x) \bigg| 1 \leq N_x \leq n - 1 \right] \left\} \right.,$$

where $N_x \sim \text{Binomial}(n, x)$, and $W(0) = W(1) = 0$. 

Problem Statement

Structural Results

Bisection Policy
Bisection Policy

- Suppose we queried the midpoint of the interval, regardless of the number of elements to sort, i.e., choose $x = 1/2$ for all $n$.
- We call this the *Bisection Policy*.
- Denote $W_\beta(n)$ the expected number of function evaluations to sort $n$ elements by their roots, under the bisection policy.
Theorem (Bisection Policy Recursion)

For all \( n \geq 2 \), we have

\[
W^\beta(n) = 1 + 2 \mathbb{E} \left[ W^\beta(N_{1/2}) \right],
\]

where \( N_{1/2} \sim \text{Binomial}(n, 1/2) \), and \( W^\beta(0) = W^\beta(1) = 0 \).
Bisection vs. Optimal Policy

![Graph showing expected function evaluations for Bisection Policy and Optimal Policy with respect to n. The graph indicates that the expected function evaluations for the Optimal Policy are significantly lower compared to the Bisection Policy for larger values of n.](image-url)
Is Bisection Optimal?

No. Consider the case $n = 6$. 

Problem Statement
Structural Results
Bisection Policy
Is Bisection Optimal?

No. Consider the case $n = 6$. 
Is Bisection Optimal?

The case $n = 6$
Is Bisection Optimal?

Linear Growth Rate

Problem Statement
Structural Results
Bisection Policy
Theorem (Lower Bound)

For all \( n \geq 0 \), we have

\[
W(n) \geq n - 1.
\]
Theorem (Lower Bound)

For all $n \geq 0$, we have

$$W(n) \geq n - 1.$$ 

Proof: If we want to sort $n$ elements, we must perform at least $n - 1$ function evaluations of $f$ to separate them.
Theorem (Lower Bound)

For all \( n \geq 0 \), we have

\[ W(n) \geq n - 1. \]

**Proof**: If we want to sort \( n \) elements, we must perform at least \( n - 1 \) function evaluations of \( f \) to separate them.

Can we find upper bounds on \( W^\beta \)?
What’s the difference?

For notation purposes, define $\Delta h(n) = h(n + 1) - h(n)$.

**Theorem (Recursion for $\Delta W^\beta$)**

For all $n \geq 2$, we have

$$\Delta W^\beta(n) = \mathbb{E} \left[ \Delta W^\beta(N_{1/2}) \right],$$

where $N_{1/2} \sim \text{Binomial}(n, 1/2)$, and stopping conditions $\Delta W^\beta(0) = 0$ and $\Delta W^\beta(1) = 2$. 
Bounding the Bisection Policy

Goal: show for some $m, \gamma > 0$ that $\Delta W^\beta(n) \leq \gamma$ for all $n \geq m$. 
Goal: show for some $m, \gamma > 0$ that $\Delta W^\beta(n) \leq \gamma$ for all $n \geq m$.

- Each subsequent term of $\Delta W^\beta$ is a weighted average of the previous terms in the sequence.
- Idea: Induction!
Bounding the Bisection Policy

Goal: show for some $m, \gamma > 0$ that $\Delta W^\beta(n) \leq \gamma$ for all $n \geq m$.

$$\mathbb{E} [\Delta W^\beta(N_{1/2})]$$

- Bound above using $\mathbb{E} [\Delta W^\beta(N_{1/2}) \mid N_{1/2} \leq m - 1]$ ?
- Bound above using $\mathbb{E} [\Delta W^\beta(N_{1/2}) \mid N_{1/2} \geq m]$ Induction Hypthesis
Computational Upper Bounds

Growth Rate of Expected Evaluations Required to Sort

PMF of Truncated Binomial Distribution

Problem Statement

Structural Results

Bisection Policy
Computational Upper Bounds

Growth Rate of Expected Evaluations Required to Sort

PMF of Truncated Binomial Distribution

Problem Statement

Structural Results

Bisection Policy
Computational Upper Bounds

Growth Rate of Expected Evaluations Required to Sort

PMF of Truncated Binomial Distribution
Computational Upper Bounds

Growth Rate of Expected Evaluations Required to Sort

PMF of Truncated Binomial Distribution
A Useful Property of Binomial RVs
A Useful Property of Binomial RVs
A Useful Property of Binomial RVs
A Useful Property of Binomial RVs
Computational Upper Bounds

Growth Rate of Expected Evaluations Required to Sort

PMF of Truncated Binomial Distribution
Computational Upper Bounds

Growth Rate of Expected Evaluations Required to Sort

PMF of Truncated Binomial Distribution

Problem Statement

Structural Results

Bisection Policy
Theorem (Upper Bound on Bisection Policy)

For some non-negative integer \( m \), we define 
\[
g_m(\ell) = \max_{k \in [\ell, m-1]} \Delta W^\beta(k),
\]
and define \( h \) as
\[
h_m(n) = \mathbb{E} \left[ g_m \left( \frac{N_{1/2}}{2} \right) \mid N_{1/2} \leq m - 1 \right],
\]
where \( N_{1/2} \sim \text{Binomial}(n, 1/2) \). Suppose we have \( \gamma_m > 0 \) so that the following condition holds:
\[
\max \left\{ \Delta W^\beta(m), h_m(m) \right\} \leq \gamma_m.
\]

Then for all \( n \geq m \), it must be that \( \Delta W^\beta(n) \leq \gamma_m \).
Computing Upper Bounds

- Use the computational recursion to compute $\Delta W^\beta(k)$ for $k = \{1, 2, \ldots, m\}$ for some positive integer $m$.
- Compute $\gamma_m = \max\{\Delta W^\beta(m), h_m(m)\}$. 
Computing Upper Bounds

- Use the computational recursion to compute $\Delta W^\beta(k)$ for $k = \{1, 2, \ldots, m\}$ for some positive integer $m$.
- Compute $\gamma_m = \max\{\Delta W^\beta(m), h_m(m)\}$.

Corollary (Bounds for Bisection Policy)

For $n \geq 15$, we have that

$$n - 1 \leq W(n) \leq W^\beta(n) \leq \gamma_{15} \cdot (n - 15) + W^\beta(15),$$

where $\gamma_{15} = 1.444$ and $W^\beta(15) = 22.081$. 
Further Questions

• If we were allowed to evaluate the function at multiple points in parallel, where would we query the function?

• Which subintervals do we refine at each time epoch?

• How much computing power do we allocate to each subinterval?

• What if we were only interested in sorting the rankings of the top $k$ zeros?

• What would a bisection policy look like?

• Can we effectively compute the optimal policy if the i.i.d. assumption does not hold?
Further Questions

- If we were allowed to evaluate the function at multiple points in parallel, where would we query the function?
  - Which subintervals do we refine at each time epoch?
  - How much computing power do we allocate to each subinterval?

- What if we were only interested in sorting the rankings of the top $k$ zeros?
- What would a bisection policy look like?
- Can we effectively compute the optimal policy if the i.i.d. assumption does not hold?
Further Questions

- If we were allowed to evaluate the function at multiple points in parallel, where would we query the function?
  - Which subintervals do we refine at each time epoch?
  - How much computing power do we allocate to each subinterval?

- What if we were only interested in sorting the rankings of the top \( k \) zeros?
  - What would a bisection policy look like?
  - Can we effectively compute the optimal policy if the i.i.d. assumption does not hold?